HDEM: A Hierarchical Dynamic Ensemble Model for Accurate Runtime Prediction on High-Performance Computing Platforms

Thanh Hoang Le Hai [✉][1][0000−0001−7821−9158], Huy Nguyen Tuan, Bao Tran Dang[1][0009−0004−0894−7212], Bao Vo Thuong[1][0009−0008−1039−8936], Khoi Phan Tran Dinh, and Nam Thoai [✉][1][0000−0003−0499−8640]

High Performance Computing Laboratory, Faculty of Computer Science and Engineering,
Advanced Institute of Interdisciplinary Science and Technology,
Ho Chi Minh City University of Technology (HCMUT),
268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam
Vietnam National University Ho Chi Minh City,
Linh Trung Ward, Thu Duc City, Ho Chi Minh City, Vietnam
`{thanhhoang,huy.nguyentuank22,bao.trandang,bao.vo1092004, khoi.phantrandinh25082005,namthoai}@hcmut.edu.vn`

Abstract. Accurate job runtime prediction for High Performance Computing (HPC) workloads is crucial for efficient job scheduling, resource allocation, and system performance optimization. However, the dynamic nature of HPC workloads, including fluctuations in job size, priority, and system conditions, poses significant challenges for traditional prediction models. To address these challenges, this study proposes a novel Hierarchical Dynamic Ensemble Model (HDEM), which adapts to workload variations by dynamically combining predictions from multiple base learners. HDEM employs a hierarchical structure where base models generate initial predictions, while a dynamic ensemble mechanism adjusts their contributions. Using historical workload data from production HPC systems, the proposed method demonstrates superior performance compared to conventional models and providing more reliable runtime estimates. This approach not only enhances prediction accuracy but also ensures adaptability to evolving workload patterns in HPC environments.

Keywords: High Performance Computing · Hierarchical Dynamic Ensemble Model · Ensemble Learning · Machine Learning · Runtime Prediction

1 Introduction

The rapid development of new technologies is profoundly transforming various aspects of life, from healthcare and education to transportation and entertainment. Behind these advancements are High Performance Computing (HPC) systems, which process and analyze massive amounts of data at incredible speeds.

[✉]: Corresponding authors

HPC systems can perform trillions of calculations per second, providing the foundation for groundbreaking progress in artificial intelligence (AI), big data, and scientific simulations. Many scientific and technological organizations have heavily invested in supercomputing systems dedicated to AI, aiming to accelerate the development of intelligent applications and services. However, the emergence of increasingly large intelligent models, such as generative AI, is placing growing pressure on HPC systems. These models require substantial computational power for training and operation, significantly increasing resource consumption and leading to greater pressure to share resources efficiently [1]. Therefore, developing an optimal solution for resource allocation is crucial, given the limited computational resources and the inability to meet the escalating demand.

To effectively leverage this powerful computing resource, selecting an appropriate job scheduling strategy plays a crucial role in optimizing resource allocation and minimizing waiting times for tasks in the execution queue on HPC systems. Advanced approaches, such as reinforcement learning agents [2] and practical scheduling algorithms [3], have shown significant potential in enhancing scheduling efficiency. One of the most critical aspects of job scheduling in HPC environments is the accuracy of job runtime prediction. Accurate predictions not only enable better estimation of job completion times but also facilitate more effective planning and resource management, ultimately improving the overall system performance.

Predicting job runtime on HPC systems is complex due to the dynamic nature of user demands, job types, resource requirements, job complexities, and system availability. These variations stem from factors such as user behavior, system maintenance, hardware upgrades, and unexpected technical issues. HPC systems often serve diverse users, including researchers, engineers, and students, each running different workloads, from quick simulations to prolonged AI tasks, with varying input parameters further complicating predictions. Although many approaches, ranging from simple heuristic techniques to advanced machine learning models [4, 5], have been explored, they still struggle to overcome the aforementioned challenges. As a result, a single predictive model often fails to adapt to these fluctuations, leading to inefficient resource allocation, increased waiting times, and reduced system performance.

Ensemble learning has gained significant attention in recent years as an effective approach to solving complex prediction problems across various fields [6]. By combining the outputs of multiple models, these methods enhance accuracy and robustness, making them particularly suitable for dynamic and intricate environments like HPC systems. As a result, ensemble models have been widely applied to predicting HPC job runtimes, demonstrating their potential in improving prediction precision [7–9]. However, most models prioritize maximizing accuracy while failing to adapt effectively to temporal shifts in workload patterns.

Recognizing this challenge, we propose a novel approach called HDEM (Hierarchical Dynamic Ensemble Model) for predicting job runtime in HPC systems. Our method is structured into multiple levels. At Level 1, multiple base models are utilized and organized into distinct groups to maximize their predictive power. The predictions from these groups are then aggregated at Level 2, where base model weights are dynamically adjusted based on performance, serving as inputs for the meta-learner. Finally, at Level 3, a meta-model combines the refined outputs to generate the final prediction. This hierarchical structure enhances robustness and flexibility, making HDEM more effective in handling the complexities of HPC job.

The main contributions of this paper are as follows:

1. We introduced a novel Hierarchical Dynamic Ensemble Mode specifically designed for job runtime prediction on HPC systems.
2. We evaluated our approach with HPC workloads collected from real HPC systems, demonstrating the effectiveness of HDEM across diverse HPC environments.
3. Our approach has significantly reduced prediction error compared to both traditional single predictive models and conventional ensemble learning methods.

The rest of this paper is organized as follows: Section 2 provides an overview of related work in HPC job runtime prediction and ensemble learning. Section 3 formally defines the problem and outlines the challenges specific to HPC environments. Section 4 describes our proposed HDEM approach in detail. Section 5 presents our experimental setup, including datasets and evaluation metrics. Section 6 discusses the results and implications of our work. Finally, Section 7 concludes the paper and suggests directions for future research.

2 Related Works

Numerous studies have been conducted to address the challenge of predicting job runtime on HPC systems. In particular, the issue of inaccurate runtime prediction has been recognized as a critical factor affecting the efficient utilization of these systems [?]. A simple approach, such as using the average of the two most recent actual runtimes from a user's job history [10], has shown significant improvements in HPC system performance. Modern techniques, including genetic algorithms [11], artificial neural networks [12], Gate Recurrent Units (GRU) [13], and Transformers [14], have also been explored for job runtime prediction in HPC environments. Researchers at the HPC Lab of HCMUT have contributed to this field by introducing a user-aligned prediction method based on kNN [15], providing a straightforward and reliable approach to enhance job scheduling efficiency.

A common limitation of these approaches is their reliance on a single prediction method, despite the fact that job resource parameters in HPC systems are influenced by numerous complex factors and do not follow a fixed pattern. Therefore, combining multiple algorithms for prediction emerges as a promising solution for resource estimation in HPC applications. Recently, some studies have applied ensemble learning techniques for job runtime prediction [16, 17], achieving higher accuracy and faster computation compared to single algorithms. However, these models lack the ability to self-adjust in response to operational fluctuations.

To address the challenge of self-adaptation in ensemble learning models for highly dynamic data over time, a method called AWEE has been employed to tackle data imbalance and concept drift, which are commonly observed in network traffic classification. In this approach, base models are divided into two groups, referred to as sub-ensembles, each focusing on different aspects of the classification task. Additionally, a sliding window-based validation mechanism is used to adapt to changes in data distribution over time [18].

Building on these principles, we have adopted a similar approach for regression tasks in HPC workloads and propose the HDEM framework as an effective solution. This approach enhances both adaptability and robustness in dynamic HPC environments.

3 Background

3.1 Runtime Prediction on HPC systems

HPC systems are defined by their high-speed processing power, high-performance networking, and large-memory capacity, enabling them to execute massive parallel computations efficiently. Supercomputers, a specialized type of HPC system, offer exceptional computational power and speed, making them essential for high-performance computing applications [19]. With the increasing demand for HPC resources, efficiently managing these large-scale systems has become essential. Effective job scheduling is crucial for optimizing resource allocation and reducing queue wait times for tasks. Scheduling decisions rely on the actual runtime of jobs; for instance, short-running jobs can be prioritized at the front of the queue, while long-running jobs might be placed at the end (Shorted Job First) or split into smaller tasks for optimal scheduling. However, the actual runtime of a job is unknown in advance and must be predicted. Therefore, the more accurate the prediction, the more optimized the scheduling becomes.

There are several challenges to address when predicting job runtimes in HPC systems.

- **Workload type:** HPC workloads are highly variable and dynamic, differing in computational intensity, memory usage, and I/O operations.
- **System Configuration:** HPC systems often consist of nodes with varying architectures (e.g., CPU, GPU, FPGA). A workload running on different nodes or a mix of nodes can have widely varying runtimes.
- **User Estimate:** Users have to provide their walltime estimation before submitting a new job to the scheduler. User-provided estimates are often inaccurate due to overestimation (to avoid premature termination) or underestimation (causing job failures and resubmissions). These inaccuracies lead to inefficient scheduling, longer wait times, and reduced cluster efficiency [20].

- **Hardware limitations:** Due to heterogeneous components like CPUs, GPUs, and FPGAs, along with performance variability from memory bottlenecks, I/O contention, and network constraints.

In the context of HPC systems with diverse and constantly changing workloads, we cannot rely solely on manual estimates or individual machine learning models. Instead, ensemble learning emerges as a promising approach to improve resource management performance. By combining multiple prediction models, ensemble models can provide more accurate, generalized, and reliable predictions of job resources, particularly runtime, even under rapidly changing computing infrastructure conditions.

3.2 Ensemble Learning

Ensemble learning is a machine learning approach where multiple models are trained to solve the same problem, and their predictions are combined to enhance accuracy and generalization [21]. This method is particularly useful for handling complex problems while maintaining reasonable resource consumption [22]. In high-performance computing (HPC) environments, where workloads are highly dynamic and diverse, ensemble learning offers a promising strategy for improving resource management efficiency. By integrating multiple predictors, ensemble models can provide more precise and reliable estimates of job runtime, even in rapidly changing computing infrastructures.

There are three primary ensemble techniques:

- **Bagging:** This method partitions the training dataset into multiple subsets, each used to train a different base model [23]. The final prediction is obtained by aggregating or voting on the results of these models. Bagging helps reduce variance and mitigates overfitting.
- **Boosting:** Unlike bagging, boosting trains models sequentially, where each model builds on the errors of the previous one to improve accuracy step by step [24].
- **Stacking:** The stacking method combines the outputs of multiple models using a meta-learner [25], which determines the final prediction. This approach leverages the strengths of different models, making it particularly effective for runtime prediction in HPC systems, where job characteristics vary widely and scheduling conditions change over time.

By leveraging techniques such as Bagging, Boosting, or Stacking, ensemble learning can minimize errors caused by noisy data or the imperfections of a single model. Specifically, individual models often have moderate prediction accuracy and performance, with high bias, high variance, and larger errors, making them less stable. In contrast, ensemble learning excels by combining multiple models to reduce bias and variance, resulting in more accurate predictions, better performance, and fewer errors. This method significantly enhances the overall stability of the model, making ensemble learning an optimal choice for complex problems where individual models struggle to capture the full complexity of the data [6].

Ensemble learning, while powerful, comes with certain limitations and trade-offs. The increased complexity of combining multiple models can make interpretation and debugging more challenging. Additionally, training multiple models requires more computational resources, which may not be feasible for applications with limited computational capacity. There is also a risk of correlated errors if the base models are too similar, reducing the effectiveness of the ensemble. Consequently, ensemble learning may not be suitable for latency-sensitive or resource-constrained applications. Therefore, it is crucial to evaluate the trade-off between the accuracy gains achieved and the additional resources required to ensure the method aligns with the specific requirements of the application [26].

4 Proposed Method

4.1 Overview of HDEM

Hierarchical Dynamic Ensemble Model (HDEM) is an advanced machine learning framework designed to enhance predictive accuracy by dynamically adjusting the contributions of multiple base models (Fig 1). Given the inherently dynamic nature of high-performance computing (HPC) environments—characterized by fluctuating workload patterns, resource availability, and system performance metrics—HDEM continuously updates model weights based on real-time validation and training feedback. By dynamically optimizing its predictive models, it minimizes errors, improves resource allocation efficiency, and ensures seamless adaptation to evolving computational demands, making it particularly well-suited for the rapid fluctuations of HPC systems.

The HDEM framework follows a structured five-stage process to enable adaptive and efficient predictive modeling in HPC environments.It begins with data preprocessing, where irrelevant information is removed, and key features are selected to provide structured input. The dataset is then split into training, validation, and testing subsets. Next, sub-model integration groups diverse base model into specialized sub-ensembles, enhancing predictive robustness. These sub-ensembles are further combined in hierarchical aggregation, forming an ensemble-of-ensembles that improves resilience to workload variations. Performance tracking continuously evaluates predictive accuracy, identifying potential degradation. Finally, adaptive weighting dynamically adjusts model contributions, prioritizing high-performing base model while reducing the influence of weaker ones. This structured approach ensures HDEM remains precise, efficient, and responsive to changing computational demands. Algorithm 1 presents a detailed pseudo-code formulation of our proposed HDEM methodology, outlining the systematic procedure for model implementation.

```
Algorithm 1: HDEM - Hierarchical Dynamic Ensemble Model
1. Input: HPC workload dataset $D$, base models $M1 - M9$, initial weights $W1 - W9$,
   learning rate $lr$, Discount factor $Df$
2. Step 1: Preprocessing D
3. Remove outliers, normalize features with StandardScaler, fill missing data D
4. Split D into train set (70%), validation set (15%), test set (15%)
5. Step 2: Train Sub-ensemble
6. Initialize sub-ensembles $S1 = \{M1, M2, M3\}, S2 = \{M4, M5, M6\}, S3 = \{M7, M8, M9\}$
7. Train base models on train set; compute initial weights $W_i = 1/3$ per sub-ensemble
8. Aggregate base-model predictions via soft regressor
9. Step 3: Train Meta model
10. Train meta-model (Gradient Boosting) on sub-ensemble outputs
11. Update weights $W_i$ using $Eq.(2)$ if $R^2$ drops below threshold
12. Output: Trained HDEM
```

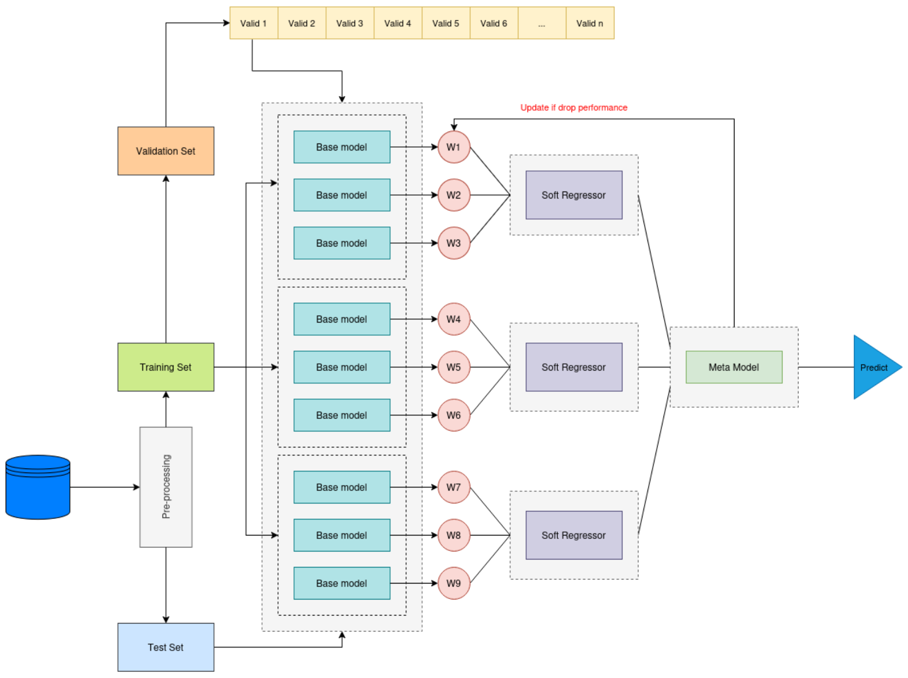

Fig. 1. The architecture of HDEM

4.2 Hierarchical Structure of HDEM

Sub-ensembles form the core of HDEM's predictive framework, enhancing both accuracy and adaptability. Instead of treating all base models as a single unit like traditional stacking, HDEM organizes them into three specialized sub-ensembles, each designed to capture different workload characteristics. This structured division mitigates bias and variance, ensuring the model remains robust under changing computational conditions.

The first sub-ensemble leverages tree-based ensemble methods to enhance robustness, generalization, and predictive accuracy. Extra Trees introduces extreme randomization in tree splits, reducing variance and improving performance on complex dependencies in HPC runtime prediction. Random Forest further strengthens stability by averaging multiple decision trees, mitigating overfitting. XGBoost, a gradient boosting algorithm, refines predictions iteratively, making it highly effective at capturing intricate workload dependencies. This combination excels in handling high-dimensional datasets and detecting subtle patterns in HPC runtime prediction.

The second sub-ensemble integrates tree-based and neural network-based approaches for a balance of interpretability and deep learning capabilities. Random Forest ensures stability and reliable feature selection, making it suitable for dynamic workloads. MLP provides the ability to learn complex, non-linear dependencies that traditional models might overlook, which is crucial for fluctuating runtime patterns. Gradient Boosting refines predictions by optimizing weak learners in sequence, allowing for improved accuracy in predicting HPC execution times. This sub-ensemble is well-suited for real-time adaptability and complex dependency modeling.

The third sub-ensemble prioritizes feature selection, scalability, and variance reduction. LASSO efficiently eliminates redundant variables, improving computational efficiency for large-scale HPC runtime prediction tasks. XGBoost enhances predictive performance by employing gradient boosting to fine-tune model weights, making it highly effective at capturing workload dependencies. Extra Trees further stabilizes the ensemble by reducing overfitting and improving robustness against noisy data. This sub-ensemble is particularly useful for interpretable and scalable HPC runtime forecasting, ensuring both accuracy and computational efficiency.

The decision to structure HDEM with three sub-ensembles was not arbitrary but was based on a careful balance between predictive performance, model diversity, and computational efficiency. Initial experiments were conducted to evaluate different configurations, including architectures with two and four sub-ensembles, to assess their impact on accuracy and computational cost. When using only two sub-ensembles, the model maintained relatively high accuracy but suffered from limited generalization due to insufficient diversity among the grouped base models. This lack of diversity led to biases in predictions, particularly for HPC workloads with highly dynamic characteristics. On the other hand, increasing the number of sub-ensembles to four resulted in a slight improvement in accuracy, but at a significant computational cost. Training and optimizing a larger number of models increased processing overhead, making the system less practical for real-world HPC deployments, where computational resources are often constrained. Through these experiments, the selection of three sub-ensembles emerged as the optimal balance between model diversity, generalization ability, and computational feasibility. This structure allows HDEM to leverage multiple learning paradigms while maintaining efficiency. Specifically, each sub-ensemble plays a distinct role: one integrates tree-based ensemble methods and adaptive learning, another balances deep learning, instance-based learning, and interpretability, and the last focuses on feature selection, boosting techniques, and variance reduction.

4.3 Initial Weights, Soft Regressor and Meta Model

The initialization of base model weights represents a fundamental component in constructing and optimizing our (HDEM). We implement an equitable weight initialization strategy that ensures unbiased contribution from each base model, establishing a robust foundation for diverse dataset applications. The architecture incorporates three distinct sub-ensembles, where the weights within each sub-ensemble sum to unity. We employ a soft regressor mechanism to aggregate individual predictions into consolidated sub-ensemble outputs. The validation process employs R-squared metrics to evaluate performance, with higher weights assigned to superior-performing base model. The fomula for weight normalization follows:

$$W_i = \frac{W_i}{\sum_{j=1}^{n} W_j} \tag{1}$$

where W_i represents the normalized weight for base model i, W_i denotes the validation weight for base model i, and $\sum_{j=1}^{n} W_j$ indicates the cumulative validation weights within the sub-ensemble. This normalization methodology ensures balanced contribution across all base models, prevention of single-model dominance, and optimal integration of diverse prediction capabilities. The meta-model layer utilizes the outputs from all three sub-ensembles as input features, creating a hierarchical structure that leverages the strengths of each component while maintaining system stability and prediction accuracy. This comprehensive approach ensures robust performance across varying datasets while preventing over-reliance on any individual base model, ultimately enhancing the model's generalization capabilities and prediction accuracy.

About meta-model, Gradient Boosting is chosen because it effectively integrates the strengths of each sub-ensemble while mitigating their weaknesses, leading to improved generalization and prediction accuracy in HPC runtime estimation. The first sub-ensemble, consisting of tree-based models (ExtraTrees, RandomForest, and XGBoost), captures nonlinear patterns well but is prone to overfitting. The second sub-ensemble, which includes RandomForest, MLP, and GradientBoosting, leverages MLP's ability to model complex relationships, but MLP requires extensive tuning and large datasets. The third sub-ensemble, combining Lasso, XGBoost, and ExtraTrees, benefits from Lasso's feature selection but struggles with highly nonlinear relationships, limiting its predictive power. Gradient Boosting is particularly well-suited as the meta-model because it iteratively refines predictions by learning from residual errors, effectively balancing bias and variance to prevent both overfitting and underfitting. Unlike simple averaging, it adapts by assigning higher importance to more accurate base models while correcting weaker predictions. This adaptive learning process enhances robustness, making Gradient Boosting a reliable final predictor that improves HPC runtime estimation and optimizes scheduling performance.

4.4 Base Model Weight Update Mechanism

In this study, we introduce an innovative weight adaptation mechanism within a hierarchical ensemble framework, purpose-built for the demands of High-Performance Computing (HPC) environments. This mechanism combines gradient-based optimization with adaptive techniques to improve both model accuracy and robustness. Central to our approach is the dynamic adjustment of base model weights, which is triggered when a performance decline surpasses a predefined threshold. This adjustment adheres to the formula:

$$W = W_o - lr * (R_o - R_c) * Df \tag{2}$$

where W_c represents the current weight configuration, W_o denotes the prior weight state, R_o signifies the historical R-squared metric, R_c indicates the current R-squared evaluation, Df is the discount factor parameter, and lr is the learning rate. The learning rate acts as a crucial hyperparameter that manages the rate at which weights are modified, influencing the system's sensitivity to performance variations and striking a balance between rapid adaptation and stability. The discount factor, conversely, functions as a modulation mechanism, regulating the update intensity for individual base model, sustaining ensemble robustness, and determining the impact of performance differentials. To ensure system integrity, we implement a strict normalization protocol, scaling weights to sum to unity, preventing the dominance of any single base model, and promoting a balanced contribution across the ensemble. This overall mechanism provides advanced adaptive properties, enabling real-time responses to evolving data patterns, dynamic adjustments to performance variations, optimization of the influence of high-performing base model, and mitigation of the impact of underperforming components. The result is an ensemble that maintains resilience and responsiveness in complex, real-world HPC environments, maximizing prediction accuracy and computational efficiency.

5 Experiments and Results

5.1 Evaluation Metrics

To thoroughly evaluate the performance of the proposed solution, we use popular evaluation metrics that are widely used in related studies, including:

– **Mean Absolute Error (MAE)**: MAE is a common regression metric that measures the average absolute difference between predicted and actual values. It is calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_{\text{pred},i} - y_{\text{true},i}| \tag{3}$$

where $y_{\text{pred},i}$ is the predicted runtime, $y_{\text{true},i}$ is the actual runtime, and n is the number of scheduled jobs.

– **Root Mean Square Error (RMSE)**: RMSE is a widely used metric for regression problems, providing a measure of the average magnitude of prediction errors. It is calculated as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\text{pred},i} - y_{\text{true},i})^2} \tag{4}$$

where $y_{\text{pred},i}$ is the predicted runtime, $y_{\text{true},i}$ is the actual runtime, and n is the number of scheduled jobs.

– **R^2 Score**: The R^2 score, also known as the coefficient of determination, is a statistical measure that indicates how well the predictions correlate with the actual value:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the actual values. The R^2 score ranges from negative to 1, where a higher score represents a better fit of the model to the training data.

5.2 Datasets

To evaluate the practical application of our proposed solution, we analyzed data from multiple sources. Initially, we utilized two established datasets from the Parallel Workload Archive [27], which are frequently cited resources in HPC performance analysis literature. Complementing these, we incorporated scheduling logs collected from the SuperNode-XP HPC system at HCMUT, VNU-HCM [28]. This methodological diversity in data selection provides comprehensive insights into the effectiveness of the HDEM method across varied operational environments, particularly highlighting its capability to integrate multiple predictors within a unified ensemble model framework.

Before conducting analysis and training the HDEM model, we first needed to load and preprocess the datasets to ensure their suitability for subsequent tasks. We selected features that align with the input data characteristics of jobs executed in real-world HPC environments. Table 1 provides information on the dataset dimensions and the selected features. To address issues of incomplete data, inconsistent records, and missing downtime information, we implemented a comprehensive data preprocessing pipeline. Data within HPC systems typically contains discrepancies, lacks critical values, or includes negative values recorded during the data collection process. Effective data preprocessing significantly enhances model performance. Our preprocessing workflow involves identifying and rectifying these issues.

Table 1. HPC Workloads Logs Inormation

Workload Log	#Jobs	#Nodes
ANL-Intrepid-2009	68,936	640
PIK-IPLEX-2009	742,964	320
SuperNode-XP-2017	19,124	24

Table 2. Feature Selection Across Datasets

Features	ANL	SuperNode	PIK
`job_id`	x		
`wait_time`	x		
`run_time`	x	x	x
`requested_processors`	x	x	
`requested_time`	x		
`num_allocated_processors`		x	
`avg_cpu_time_used`		x	x
`used_memory`		x	x
`user_id`	x	x	x
`group_id`		x	x
`executable_id`		x	x
`queue_id`	x	x	x

Data preprocessing represents a critical aspect of machine learning methodology. We employed a thorough preprocessing strategy before utilizing the data to train our HDEM model [27]. Additionally, we performed data normalization using StandardScaler to ensure feature uniformity across the dataset. This normalization step is particularly important when features exhibit significantly different value ranges, as it prevents features with larger magnitudes from dominating the learning process and ensures all features contribute appropriately to the model training.

5.3 Simulation Setup

To implement the HDEM model, we used the scikit-learn library [29]. Base predictors were configured with specific parameters, which were chosen empirically. These parameters are summarized in Table 3:

Table 3. Parameters for base predictors and Eq. 2 of HDEM

Method	**Parameters**
ExtraTrees	`n_estimators: 100, criterion: "squared_error", min_samples_split: 2, random_state: 42`
RandomForest	`n_estimators: 100, criterion: "squared_error", min_samples_split: 2`
XGBoost	`n_estimators: 100, learning_rate: 0.1, max_depth: 3, verbosity: 1`
MLP	`hidden_layer_sizes: (100,), activation: "relu", solver: "adam", max_iter: 500`
GradientBoosting	`loss: "squared_error", learning_rate: 0.1, n_estimators: 100, subsample: 1.0`
Lasso	`alpha: 1.0, fit_intercept: True, max_iter: 1000, tol: 0.0001, random_state: 42`
All models	`learning_rate: 0.1, discount_factor: 0.6, window_size: 200, drift_threshold: 0.1`

5.4 Experimental Result

This section presents a comparison of the accuracy of the proposed method, evaluating its performance across three different datasets.

On the ANL dataset, the HDEM method achieved an R^2 score of 0.87, demonstrating its ability to accurately predict task execution times in an HPC environment. The RMSE and MAE values, 893.10 and 409.15, respectively, indicate a low error rate, improving forecasting performance compared to traditional methods.

Table 4. Comparison of HDEM and base models on ANL-Intrepid-2009.

Configuration	*RMSE*	*MAE*	R^2
HDEM	**893.10**	**409.15**	**0.8711**
RANDOMFOREST	966.96	434.99	0.8465
EXTRATREES	999.32	434.43	0.8361
TRADITIONAL STACKING	1076.40	649.47	0.8128
KNN	1116.84	512.81	0.7953
CATBOOST	1152.99	692.26	0.7818
DECISIONTREE	1220.37	463.06	0.7556
GRADIENTBOOSTING	1234.22	762.83	0.7499
XGBOOST	1238.23	759.13	0.7483
MLP	1590.93	1065.40	0.5846
LASSO	1806.43	1321.84	0.4644
LINEARREG	1810.30	1323.72	0.4621

Next, on the HCMUT dataset, the HDEM method continued to show outstanding performance with an $\mathbf{R^2}$ score of 0.967, indicating a high level of agreement between actual and predicted values. The RMSE and MAE values of 26335.75 and 5776.01, respectively, highlight the model's ability to capture variations in execution times in complex HPC environments with high accuracy.

Table 5. Comparison of HDEM and base models on HCMUT-SuperNodeXP-2017.

Configuration	*RMSE*	*MAE*	R^2
HDEM	**26335.75**	**5776.01**	**0.9622**
EXTRATREES	37184.96	5892.40	0.9246
TRADITIONAL STACKING	38876.12	12407.07	0.9177
GRADIENTBOOSTING	41324.49	9380.34	0.9069
RANDOMFOREST	43258.05	7697.75	0.8979
CATBOOST	44205.85	9585.62	0.8935
XGBOOST	45059.60	10436.27	0.8893
DECISIONTREE	49735.52	8414.64	0.8651
KNN	56263.42	13532.27	0.8274
LASSO	125684.44	53421.23	0.1387
MLP	125711.33	53366.00	0.1384
LINEARREG	168593.95	32419.05	-0.5497

Finally, on the PIK-IPLEX dataset, the HDEM method achieved an R^2 score of 0.8316, with an RMSE of 85999.43 and an MAE of 15507.37. Despite the high variability of this dataset, HDEM still demonstrated reliable predictive capabilities, effectively adapting to the characteristics of HPC workloads.

Table 6. Comparison of HDEM and base models on PIK-IPLEX-2009.

Configuration	*RMSE*	*MAE*	R^2
HDEM	**85999.43**	15507.37	**0.8316**
RANDOMFOREST	93130.90	**14858.77**	0.7819
CATBOOST	95838.40	20227.72	0.7690
GRADIENTBOOSTING	97186.48	18879.03	0.7625
XGBOOST	101306.04	20988.64	0.7419
TRADITIONAL STACKING	107922.83	19309.64	0.7348
EXTRATREES	105452.72	14949.49	0.7203
DECISIONTREE	111151.95	15210.64	0.6893
LASSO	162252.18	41077.94	0.3380
LINEARREG	162325.43	40226.45	0.3374
KNN	164076.58	27038.13	0.3229
MLP	164531.34	26863.78	0.3192

Overall, the experimental results confirm that the HDEM method not only improves execution time prediction accuracy but also adapts well to various workload types in HPC environments, ultimately optimizing scheduling and resource management efficiency.

5.5 Discussion

The experimental results demonstrate that the Hierarchical Dynamic Ensemble Model (HDEM) consistently outperforms traditional and other ensemble methods across all evaluated datasets. However, notable performance differences arise between the datasets, which can be primarily attributed to variations in workload characteristics, feature composition, data quality, and system-specific factors.

One key factor is the inherent diversity of the datasets. For instance, the ANL-Intrepid-2009 dataset, with its moderate job count and relatively homogeneous workload characteristics, yielded an R^2 of 0.87. In contrast, the HCMUT-SuperNodeXP-2017 dataset, despite having fewer jobs, offered a richer set of features—such as requested processors and used memory—which likely contributed to more consistent runtime patterns and an improved R^2 of 0.96. This suggests that when the available features more effectively capture the nuances of job execution, the model is able to better adjust its ensemble weights and enhance prediction accuracy.

On the other hand, the PIK-IPLEX-2009 dataset exhibited the lowest R^2 (0.83) among the three. This dataset is characterized by a much larger volume of jobs and higher variability in runtime and resource requirements. The increased complexity and potential data imbalances in the PIK dataset introduce additional noise, which challenges the model's ability to capture underlying patterns accurately. Thus, the performance drop on this dataset underscores the impact of dataset size and heterogeneity on prediction outcomes.

Moreover, the differences in feature sets across the datasets play a significant role in model performance. The ANL dataset includes basic features such as job_id, wait_time, and run_time, whereas the HCMUT and PIK datasets incorporate additional parameters like requested_processors, used_memory, and user or group identifiers. This variation in feature richness can affect how well the model learns the underlying workload behaviors, with richer feature sets enabling the ensemble to more effectively balance its base models and adapt to workload-specific patterns.

Lastly, variations in the HPC environments and scheduling policies from which the datasets were collected may also contribute to performance discrepancies. Differences in system architectures, job scheduling strategies, and operational policies influence the distribution of job runtimes and resource usage patterns, thereby impacting the predictive performance of models like HDEM.

In summary, the observed performance differences across datasets highlight the importance of considering dataset-specific characteristics when applying predictive models in HPC environments. While HDEM's dynamic weight adaptation mechanism helps mitigate some of these variations, future work should focus on tailoring feature engineering and model calibration techniques to better accommodate the unique properties of each dataset, ultimately enhancing prediction accuracy and resource management efficiency.

6 Conclusion

The Hierarchical Dynamic Ensemble Model (HDEM), presented in this paper, represents a novel and robust solution for job runtime prediction within High Performance Computing (HPC) environments. HDEM leverages dynamic weight adaptation through sliding window analysis and intelligent prediction aggregation via soft regressor techniques, which act as inputs for a meta-model layer. Experimental evaluations demonstrate HDEM's superior predictive capabilities compared to traditional machine learning approaches, achieving notable improvements in runtime prediction accuracy across diverse HPC workloads. While HDEM demonstrates exceptional performance, its computational complexity relative to simpler models necessitates ongoing optimization efforts. Future research will focus on streamlining the ensemble structure, implementing efficient weight update mechanisms, and exploring dimensionality reduction to enhance HDEM's practical applicability while preserving its superior predictive capabilities. This research positions HDEM as a significant advancement in HPC workload prediction, with potential for further optimization and enhancement in future iterations.

Acknowledgments. This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2021-20-16.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Geist, A., Reed, D.A.: A survey of high-performance computing scaling challenges. The International Journal of High Performance Computing Applications 31(1), 104–113 (2017), https://doi.org/10.1177/1094342015597083
2. Hai, T.H.L., Dinh, L.L., Tien, D.N., Tien, D.B.H., Thoai, N.: Irls: An improved reinforcement learning scheduler for high performance computing systems. In: 2023 International Conference on System Science and Engineering (ICSSE). pp. 587–592 (2023)
3. Fan, Y.: Job scheduling in high performance computing (2021), https://arxiv.org/abs/2109.09269
4. Zrigui, S., de Camargo, R.Y., Legrand, A., Trystram, D.: Improving the performance of batch schedulers using online job runtime classification. Journal of Parallel and Distributed Computing 164, 83–95 (2022), https://www.sciencedirect.com/science/article/pii/S0743731522000090
5. Ramachandran, S., Jayalal, M., Vasudevan, M., Das, S., Jehadeesan, R.: Combining machine learning techniques and genetic algorithm for predicting run times of high performance computing jobs. Applied Soft Computing 165, 112053 (2024), https://www.sciencedirect.com/science/article/pii/S1568494624008275
6. Mienye, I.D., Sun, Y.: A survey of ensemble learning: Concepts, algorithms, applications, and prospects. IEEE Access 10, 99129–99149 (2022)
7. Tanash, M., Yang, H., Andresen, D., Hsu, W.: Ensemble prediction of job resources to improve system performance for slurm-based hpc systems. In: Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions. PEARC '21, Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3437359.3465574
8. Taiwo, R., Yussif, A.M., Adegoke, A.H., Zayed, T.: Prediction and deployment of compressive strength of high-performance concrete using ensemble learning techniques. Construction and Building Materials 451, 138808 (2024), https://www.sciencedirect.com/science/article/pii/S0950061824039503
9. Hai, T.H.L., Nguyen, M.T., Nguyen, Q.H., Thoai, N.: Jrep - a job runtime ensemble predictor for improving scheduling performance on high performance computing systems. In: Dang, T.K., Küng, J., Chung, T.M. (eds.) Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications. pp. 144–157. Springer Nature Singapore, Singapore (2024)
10. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel Distrib. Syst. 18(6), 789–803 (Jun 2007), https://doi.org/10.1109/TPDS.2007.70606
11. Minh, T.N., Wolters, L.: Using historical data to predict application runtimes on backfilling parallel systems. In: 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing. pp. 246–252 (2010)

12. Park, J.W., Kim, E.: Runtime prediction of parallel applications with workload-aware clustering. The Journal of Supercomputing 73, 4635–4651 (2017), https://api.semanticscholar.org/CorpusID:36708462
13. Wang, Q., Zhang, H., Li, J., Shen, Y., Liu, X.: Predicting job finish time based on parameter features and running logs in supercomputing system. J. Supercomput. 78(17), 18551–18577 (Nov 2022), https://doi.org/10.1007/s11227-022-04582-5
14. Chen, F.: Job runtime prediction of hpc cluster based on pc-transformer. The Journal of Supercomputing 79(17), 20208–20234 (2023), https://doi.org/10.1007/s11227-023-05470-2
15. Le Hai, T.H., La Hoang, L., Thoai, N.: Potential of applying knn with soft walltime to improve scheduling performance. In: 2021 International Conference on Computing, Computational Modelling and Applications (ICCMA). pp. 1–8 (2021)
16. Bai, Y., Guo, Y., Zhang, H., Wang, J., Chen, J.: An ensemble learning-based hpc multi-resource demand prediction model for hybrid clusters. In: 2022 3rd International Conference on Computer Science and Management Technology (ICCSMT). pp. 413–420 (2022)
17. Chen, X., Zhang, H., Bai, H., Yang, C., Zhao, X., Li, B.: Runtime prediction of high-performance computing jobs based on ensemble learning. In: Proceedings of the 2020 4th International Conference on High Performance Compilation, Computing and Communications. p. 56–62. HP3C 2020, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3407947.3407968
18. Abbasi, M., Florez, S., Shahraki, A., Taherkordi, A., Prieto, J., Corchado Rodríguez, J.: Class imbalance in network traffic classification: An adaptive weight ensemble-of-ensemble learning method. IEEE Access PP, 1–1 (01 2025)
19. Hager, G., Wellein, G.: Introduction to high performance computing for scientists and engineers. CRC Press (2010)
20. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U., Sevcik, K.C., Wong, P.: Theory and practice in parallel job scheduling. In: JSSPP (1997)
21. Dasarathy, B., Sheela, B.: A composite classifier system design: Concepts and methodology. Proceedings of the IEEE 67(5), 708–713 (1979)
22. Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M.: Ensemble learning for data stream analysis: A survey. Information Fusion 37, 132–156 (2017), https://www.sciencedirect.com/science/article/pii/S1566253516302329
23. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996), https://api.semanticscholar.org/CorpusID:47328136
24. Schapire, R.E.: The strength of weak learnability. Machine Learning 5, 197–227 (1989), https://api.semanticscholar.org/CorpusID:6207294
25. Wolpert, D.: Stacked generalization. Neural Networks 5, 241–259 (12 1992)
26. Omar, R., Bogner, J., Muccini, H., Lago, P., Martínez-Fernández, S., Franch, X.: The more the merrier? navigating accuracy vs. energy efficiency design trade-offs in ensemble learning systems (2024), https://arxiv.org/abs/2407.02914
27. Feitelson, D.G., Tsafrir, D., Krakov, D.: Experience with using the parallel workloads archive. Journal of Parallel and Distributed Computing 74(10), 2967–2982 (2014), https://www.sciencedirect.com/science/article/pii/S0743731514001154
28. Hoang Le Hai, T., Duy, K., Manh, T., Mai Hoang, D., Thoai, N.: Deviation backfilling: A robust backfilling scheme for improving the efficiency of job scheduling on high performance computing systems. pp. 32–37 (11 2023)
29. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)