An Explainable ML-based Approach to Predicting Virtual Machine Resource Demand for High Resource Utilization in Clouds

The 2025 International Conference on Computational Science and Computational Intelligence (CSCI)
(December 3-5, 2025, Las Vegas, USA)

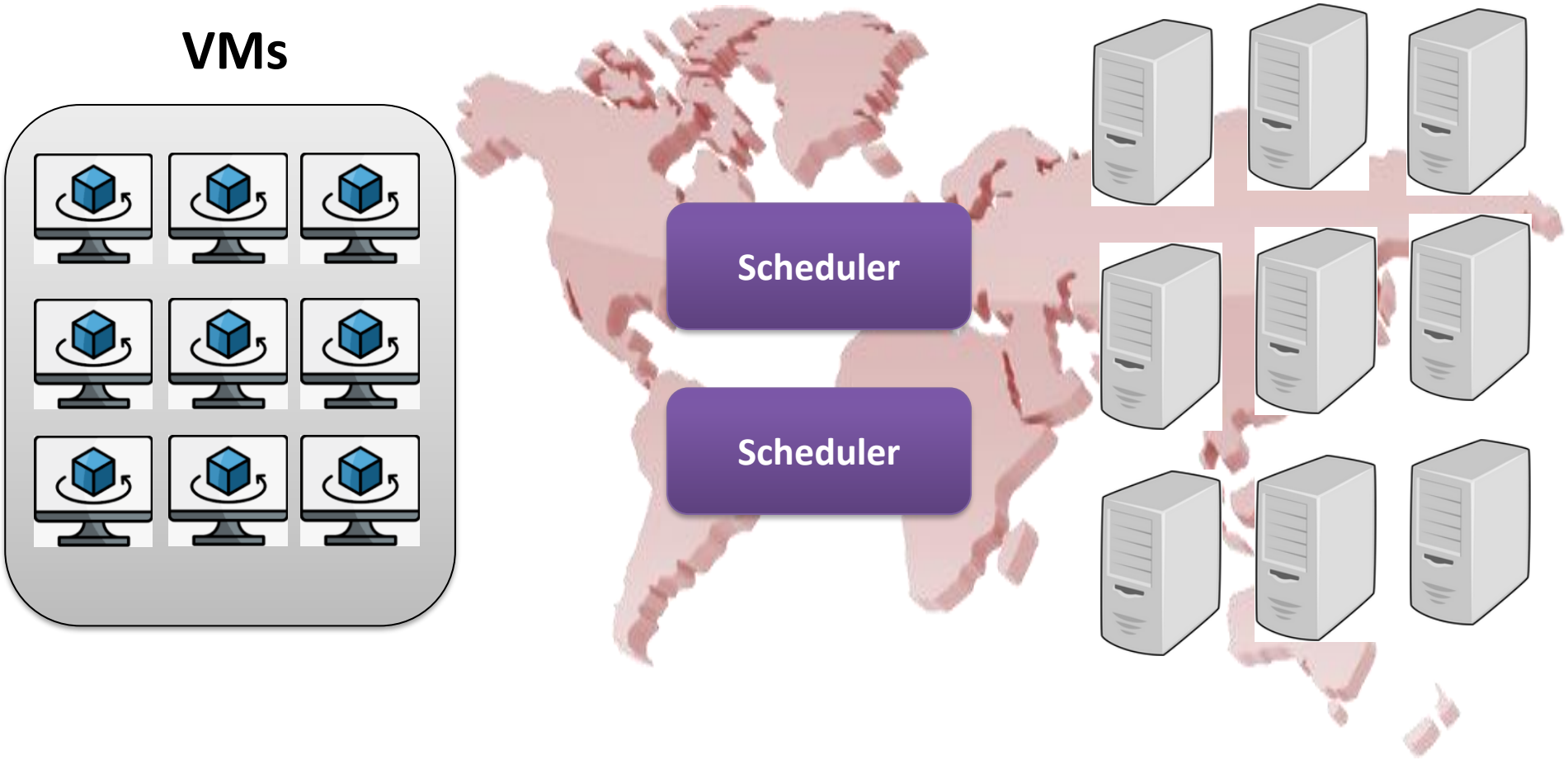
Jinwei Liu*, Kalab M. Kiros*, Rui Gong†, Clement G. Yedjou‡

*Dept. of Computer and Information Sciences, Florida A&M University, Tallahassee, FL, USA
†Dept. of Informatics and Mathematics, Mercer University, Macon, GA, USA
‡Dept. of Biological Sciences, Florida A&M University, Tallahassee, FL, USA

Talk Outline

- **Introduction**
- Problem Statement
- Proposed Approach
- System Design
- Performance Evaluation
- Conclusions and Future Work

Introduction



Motivation

- **Limitations of Traditional Scheduling**
  - Classic heuristics like FCFS, Round Robin, and Min-Min are static and rule-based.
  - They fail to adapt to dynamic workloads and ignore real-time fluctuations in resource demand.
  - Result → underutilized CPUs, longer queue times, and higher operational costs.

Motivation (cont.)

Prior Work and Research Progress

- Earlier studies applied metaheuristics (e.g., genetic or particle swarm optimization) and machine learning models for prediction
- ML models (e.g., gradient boosting, neural nets) improved accuracy in CPU and memory usage forecasting
- Explainable AI (XAI) methods like SHAP and LIME recently emerged to make ML models interpretable

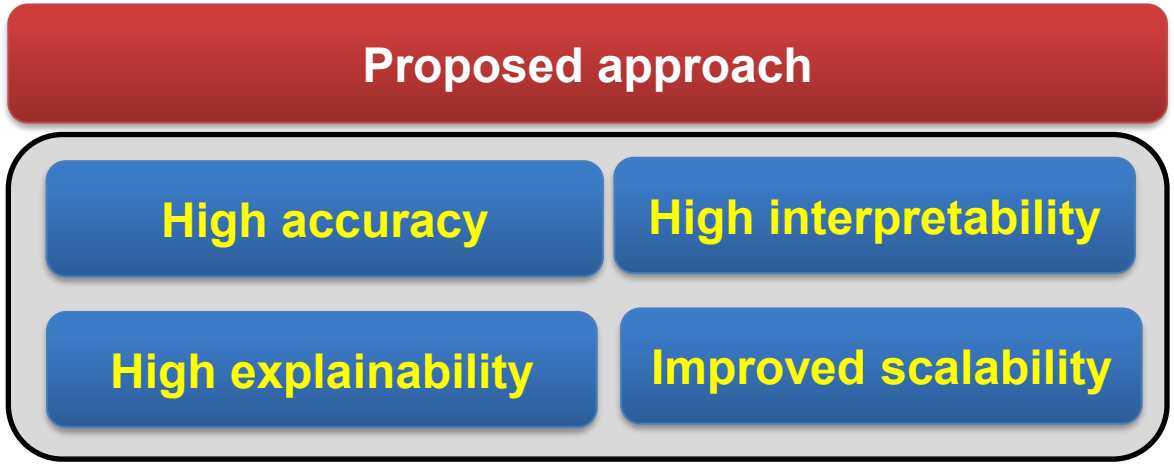
Remaining Challenges

- Most ML-based frameworks rely on synthetic or small datasets, limiting real-world applicability
- Trade-off remains between accuracy and interpretability — models are often treated as black boxes
- Cloud administrators still lack trustworthy, transparent AI-driven schedulers

This Research

An explainable machine learning (ML)-based approach to predict VM resource demand to enhance placement decisions and achieve high resource utilization in clouds

- **Features of the proposed approach**
  - High accuracy
  - High interpretability
  - High explainability
  - Improved scalability



Framework of proposed approach

Talk Outline

- Introduction
- **Problem Statement**
- Proposed Approach
- System Design
- Performance Evaluation
- Conclusions and Future Work

Problem Statement

- **Problem statement:** How can machine learning algorithms be effectively utilized to predict the resource demands of virtual machines in dynamic cloud environments, enabling proactive and optimal VM placement decisions that maximize resource utilization and minimize SLA violations?

Talk Outline

- Introduction
- Problem Statement
- **Proposed Approach**
- System Design
- Performance Evaluation
- Conclusions and Future Work

Proposed Approach

- **Concept Overview**
  - Develop an Explainable ML-based Predictive Scheduling Framework that combines:
    - XGBoost → for accurate per task CPU usage prediction
    - SHAP → for transparency and interpretability
  - Integrates directly into a cloud task scheduling pipeline to guide real-time resource allocation.

Data Description, Preprocessing and Analysis

- **Data Ingestion**
  - Extract workload traces from Google Cluster Trace Dataset (task events, job events, resource usage).

google/**cluster-data**

- **Data Preprocessing**
  - The data preprocessing stage was crucial for transforming the raw Google Cluster Workload Trace into a clean, structured, and ML-ready dataset.
    - **Data Integration:** Combined data from multiple tables — *task_events*, *task_usage*, and *job_events* — to link user-level, scheduling, and runtime performance information.
    - **Cleaning & Filtering:** Removed corrupted entries (e.g., negative or missing CPU usage values) and retained only valid workload samples.

Data Preprocessing

- **Handling Missing Values:** Applied **median imputation** for missing numerical attributes to preserve data distribution and prevent bias.
- **Normalization & Log-Scaling:** Applied logarithmic transformations to cpu_request, mem_request, and disk_space_request to reduce skewness and stabilize feature variance.

$$\begin{aligned} \log\_cpu\_request &= \log(1 + cpu\_request) \\ \log\_mem\_request &= \log(1 + mem\_request) \\ \log\_disk\_request &= \log(1 + disk\_space\_request) \end{aligned} \tag{1}$$

- **Batch Processing Strategy:** Due to the dataset's gigabyte-scale size, files were processed in batches of 100–200 to prevent memory overflow and ensure reproducibility.

Feature Engineering

- Feature engineering aimed to capture the key behavioral patterns of workloads and improve model sensitivity to important predictors.
  - **Selected Key Attributes:**
    cpu_request, mem_request, disk_space_request, user, priority, and scheduling_class were chosen as core features based on prior literature and exploratory data analysis.
  - **Engineered Features:**
    Log-transformed features (**log_cpu_request, log_mem_request, log_disk_request**) were created to manage extreme values(Skewness) and capture nonlinear relationships.
  - **Derived Indicators:**
    A binary feature **different_machines** was included to represent whether a job spans multiple machines, a critical proxy for potential resource contention.
  - **Feature-Target Definition:**
    The target variable (avg_cpu_rate) represented the observed average CPU utilization per task, establishing a regression-based prediction framework.

Model Development

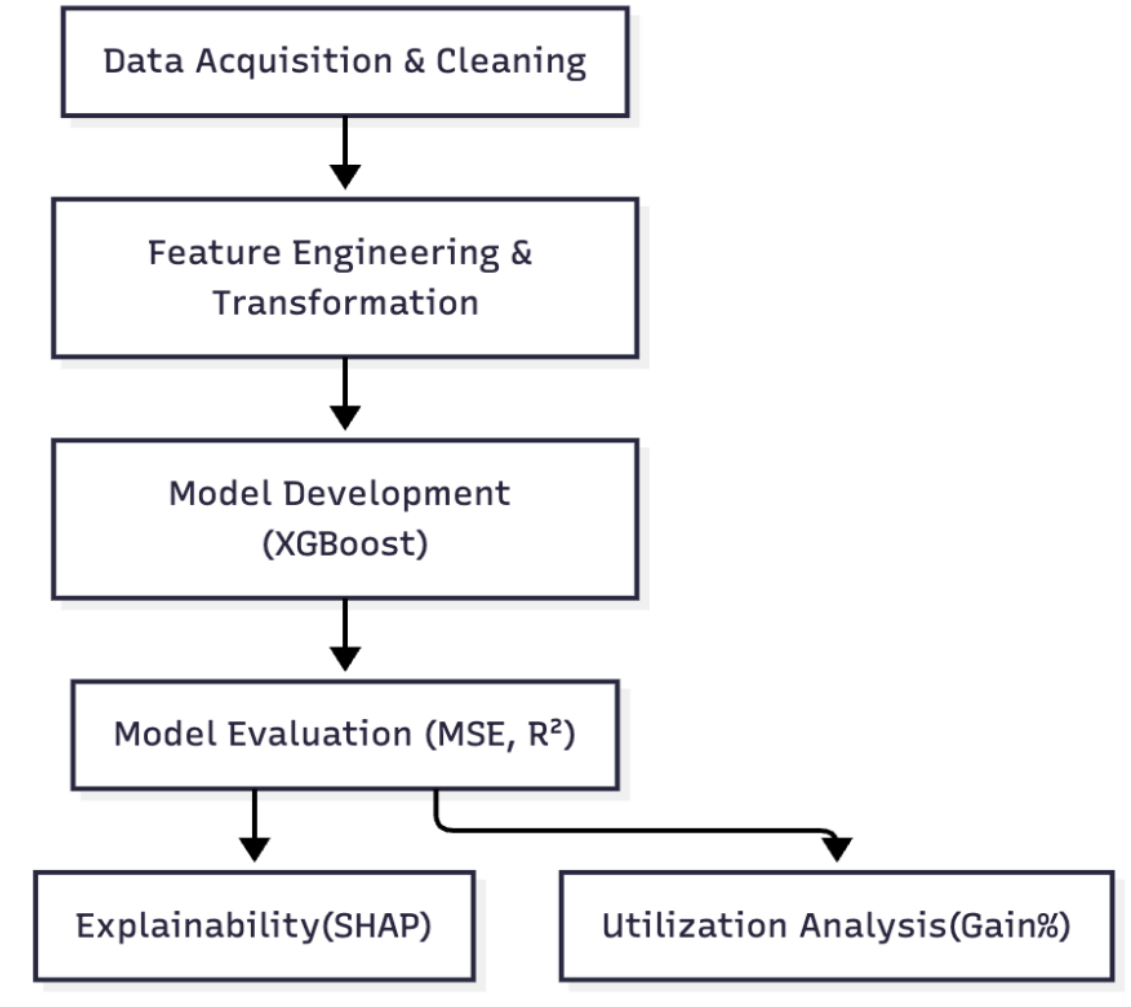
- **Model Development (XGBoost)**
  - The modeling phase focuses on balancing **predictive accuracy** and **interpretability**.
  - **Model Selection:** Adopt **Extreme Gradient Boosting (XGBoost)** for its scalability, ability to handle heterogeneous data, and compatibility with explainable AI tools.
  - **Training Configuration:**
    - **Train-Test Split:** 80–20 ratio for generalization testing.
    - **Hyperparameters:**
      - n_estimators = 500
      - max_depth = 15
      - learning_rate = 0.01

Talk Outline

- Introduction
- Problem Statement
- Proposed Approach
- **System Design**
- Performance Evaluation
- Conclusions and Future Work

System Design

- **System Architecture Workflow**



Architecture of the proposed explainable ML-based approach

Talk Outline

- Introduction
- Problem Statement
- Proposed Approach
- System Design
- **Performance Evaluation**
- Conclusions and Future Work

Performance Evaluation

- **Evaluation Metrics:**
  - **Mean Squared Error (MSE):** Measures squared prediction error magnitude.
  - **Mean Absolute Error (MAE):** Reflects average deviation from actual CPU usage.
  - **R^2 (Coefficient of Determination):** Quantifies explained variance and model fit.
- **Explainability Module (SHAP(SHapley Additive exPlanations):** Integrated SHapley Additive exPlanations (SHAP) to interpret both individual (local) and overall (global) prediction contributions. This allows stakeholders to understand *why* certain workload features influence CPU demand predictions.
- **Evaluation & Insights**
  - Compare results with **traditional algorithms** (FCFS, Round Robin).
  - Assess **efficiency gains (≈12–14%)**,model transparency.

Trace Data

- **Trace data**

Google Cluster Workload Traces

- **Google Cluster Workload Trace [1]**
  - The Google Cluster trace is a large-scale, publicly available collection of traces from production Google data centers. It records resource usage on a cluster of about 11,000 machines from May 2011 for 29 days. The trace also describes job submission, scheduling decision, task priority, tasks' resource request, and tasks' resource usage.

[1] Google trace. https://code.google.com/p/googleclusterdata/

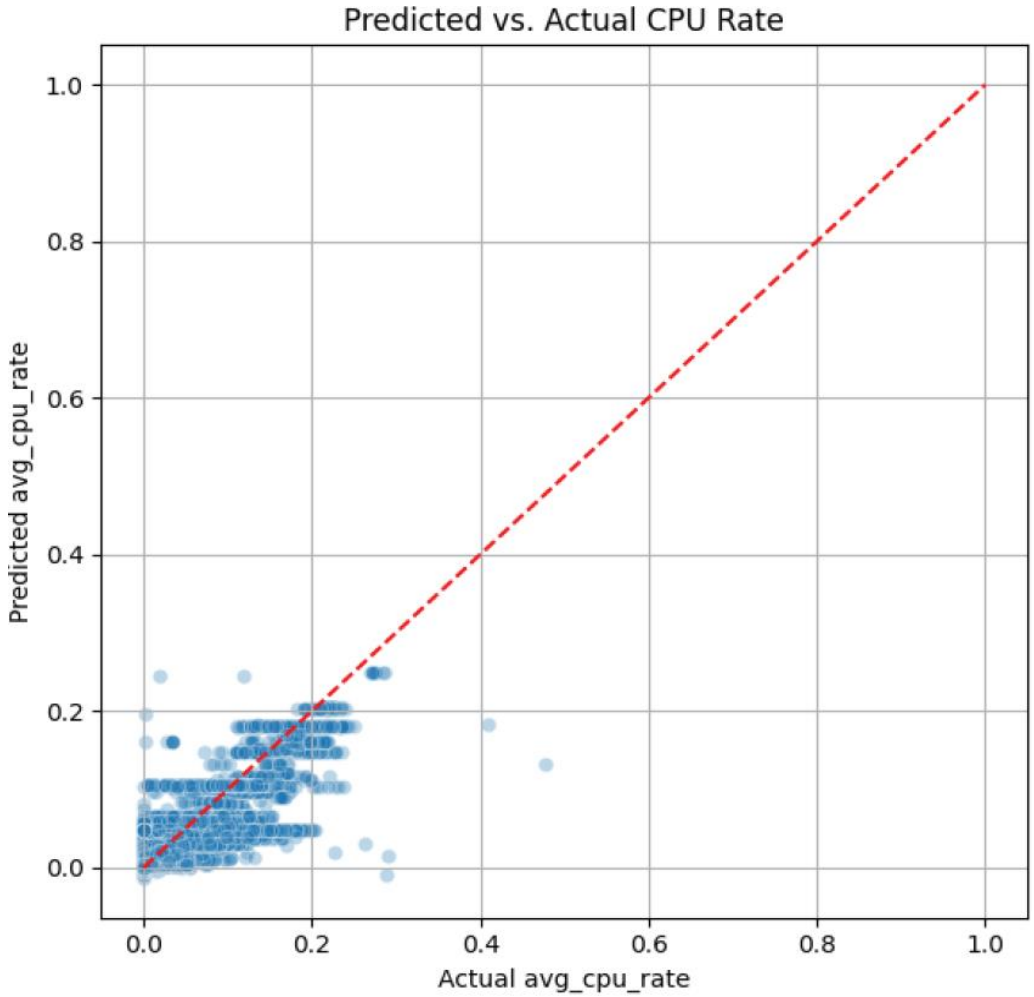
Evaluation

- **Model Accuracy**

Metric	Value	Interpretation
Mean Squared Error (MSE)	0.0001	Low average squared difference between predicted and actual CPU usage, indicating high accuracy.
Root Mean Squared Error (RMSE)	0.0100	Confirms strong predictive precision, penalizing large deviations effectively.
Mean Absolute Error (MAE)	0.0065	Small average prediction error, demonstrating consistent model performance across workloads.
R^2 (Coefficient of Determination)	0.7249	Model explains over 72% of the variance in CPU usage across diverse workloads, confirming strong explanatory power.

Evaluation (cont.)

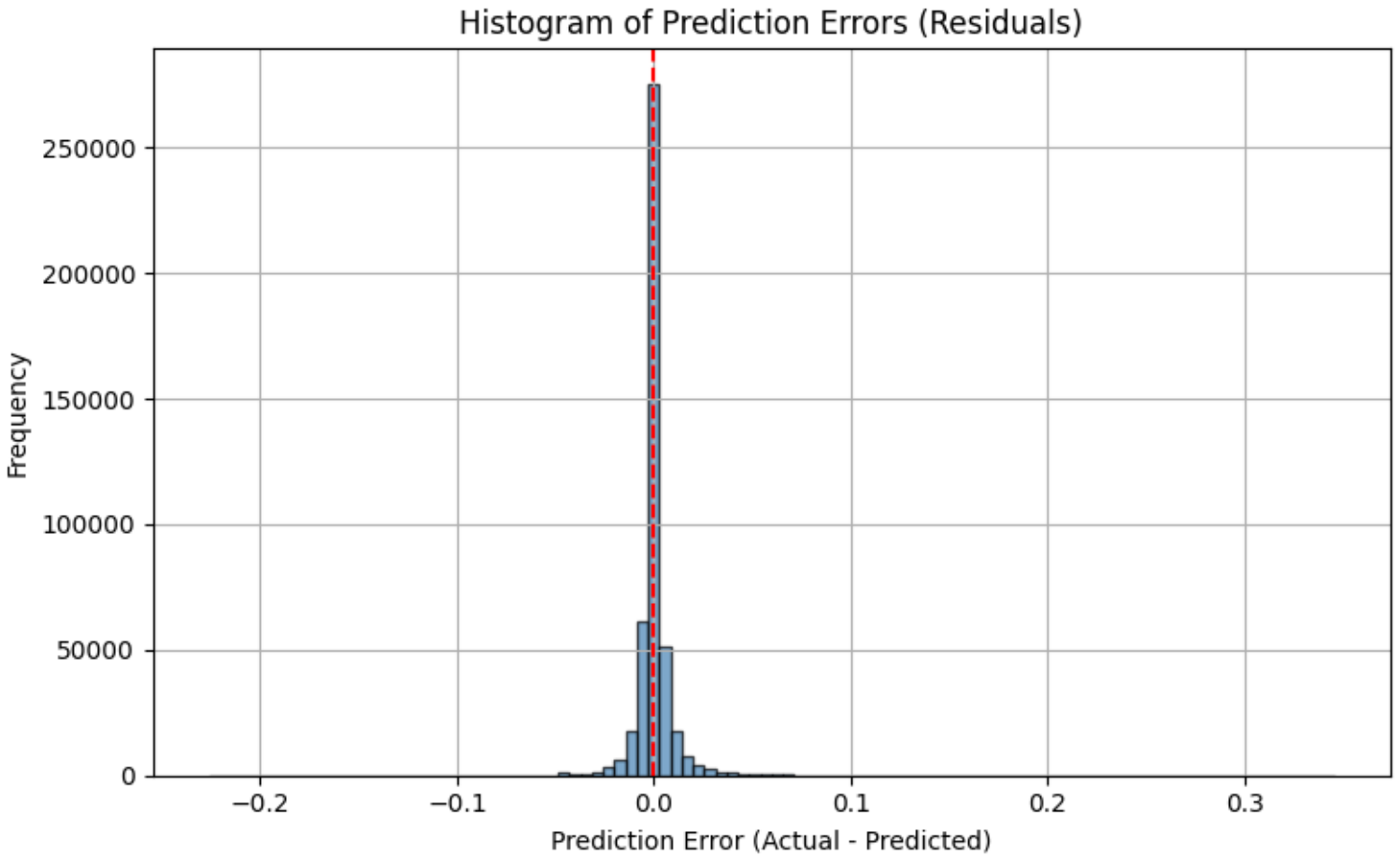
- **Model Accuracy**



Prediction vs Actual CPU Rate

Result: Data points align closely along the diagonal, confirming high correlation and minimal bias.

Evaluation (cont.)

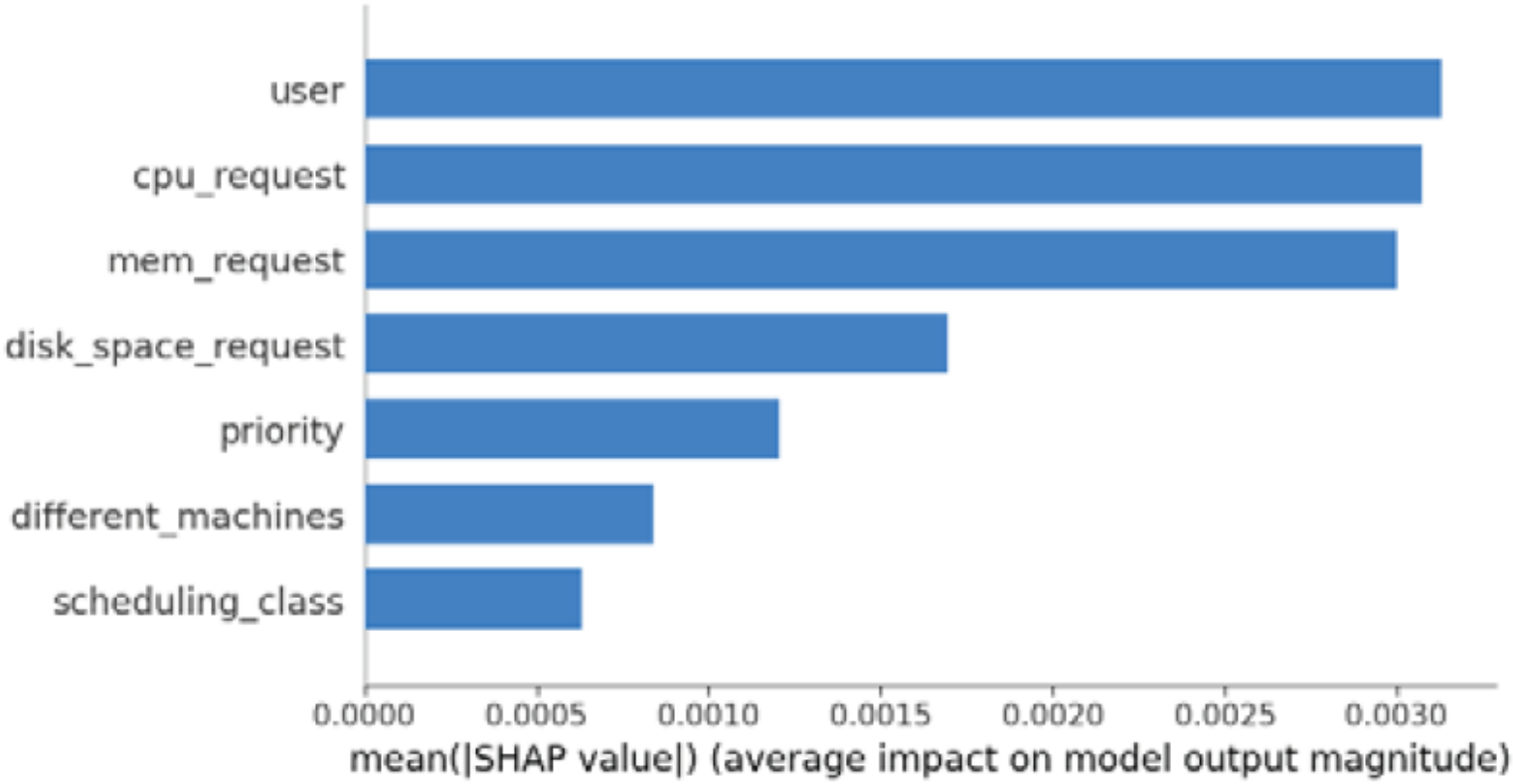
- 

Histogram of prediction errors (residuals) for the XGBoost model.

Result: Errors centered around zero, symmetric distribution → unbiased predictions.

Evaluation (cont.)

- **Interpretability Results (SHAP)**



Feature importance based on SHAP values

Top Features: user, cpu_request, mem_request, disk_space_request.
Insights: (1) *user* emerges as the dominant factor, representing application/workload groups rather than individuals; (2) This feature captures consistent behavioral patterns, making it a proxy for workload type; (3) Resource request variables (CPU, memory, disk) reinforce prediction accuracy by modeling real consumption patterns.

Evaluation (cont.)

- **Comparative Performance**
  - The XGBoost + SHAP model outperforms heuristic scheduling algorithms (e.g., FCFS, Round Robin).
  - **Efficiency Gain:** ~16.22% improvement in CPU demand prediction accuracy.
  - Indicates more optimal VM placement and reduced over-provisioning in practical scenarios.

$$\text{Gain } (\%) = 100 \times \frac{\mathbb{E}[|y - \hat{y}|]_{\text{baseline}} - \mathbb{E}[|y - \hat{y}|]_{\text{proposed}}}{\mathbb{E}[|y - \hat{y}|]_{\text{baseline}}}.$$

$$\hat{y}_{\text{mean}} = 0.026498884, \qquad y_{\text{baseline, adj}} = 0.0228000028$$

$$\text{Gain } (\%) = 100 \times \frac{0.0228000028 - 0.026498884}{0.0228000028} = 16.22\%.$$

Implications

- **Implications:**
  - Improved CPU prediction reduces over-provisioning and underutilization, leading to higher resource efficiency and operational cost savings.
  - The study demonstrates the potential of explainable AI in advancing sustainable, transparent, and adaptive cloud scheduling.

Limitations

- Despite the promising results, several limitations were identified in this study:
  - **Single-Resource Focus:**

    The current model only predicts per task-CPU utilization. Other critical resources such as memory, disk I/O, and GPU were not incorporated, which limits full multi-resource optimization.

Talk Outline

- Introduction
- Problem Statement
- Proposed Approach: Sailfish
- Design of Sailfish
- Performance Evaluation
- **Conclusions and Future Work**

Conclusions and Future Work

- **Our contributions**
  - Propose an **explainable ML-based approach** to predicting VM resource demand for enhancing VM placement decisions and improving resource utilization in clouds
  - Use XGBoost model to predict CPU demand and employ SHAP to assess the contribution of input features to the model's predictions and enhance interpretability so that our approach can balance interpretability, scalability, and performance
  - Identify a list of predictive features that can help improve the accuracy of CPU usage prediction, which can help improve resource utilization in resource scheduling
- **Future work**
  - Multi-Resource Prediction Framework: extend the XGBoost + SHAP pipeline to jointly predict CPU, memory, disk I/O, and GPU demands to enable holistic cloud workload optimization
  - Use different cloud/cluster workloads (including machine learning workloads) to fully verify the performance of our method
  - Develop an explainable and ML-based VM scheduling system for high resource utilization in Clouds

Thank you!

Questions & Comments?

Jinwei Liu, Kalab M. Kiros

{jinwei.liu, kalab1.kiros}@famu.edu

Department of Computer and Information Sciences

Florida A&M University