AMPRO-HPCC: A Machine-Learning-Tool for Predicting Resources On Slurm HPC Clusters

Mohammed Tanash, Daniel Andresen, and William Hsu
Presenter name: Mohammed Tanash – Kansas State University
tanash@ksu.edu

IARIA

Kansas State University | Computer Science

Presenter's Short Resume

- ❑ Ph.D candidate in Computer Science – Kansas State University, USA, Expected graduation (Fall 2021).
- ❑ MSC in Computer Science – New Mexico State University, USA (2014).
- ❑ MSC in Information Technology – University Utara Malaysia, Malaysia (2008).
- ❑ BSC in Computer Science – Irbid National University, Jordan (2005).
- ❑ HPC Application Specialist – University of Nebraska-Lincoln, USA, (July 2020 – Current).
- ❑ Instructor of Computer Science (2008 – 2012).
- ❑ A Cyberinfrastructure Architect, New Mexico State University (Jan 2017 – Jan 2019).
- ❑ XSEDE Student Campus Champion (2017 – Current). XSEDE Fellow (2018 – 2019).
- ❑ **Awards:**
  - ❑ Phil Andrews Award – PEARC 21 https://pearc.acm.org/pearc21/program/awards/
  - ❑ Best Full Paper Awards - Systems and Systems Software track winner – PEARC 21 https://pearc.acm.org/pearc21/program/awards/
  - ❑ Department of Computer Science's Graduate Student-of-the-Month for September 2021.
  - ❑ Outstanding Technical Staff Award, Computer Science and Engineering Department, University of Nebraska-Lincoln, May 2021.
  - ❑ Best Student Poster Award, PEARC18 Conference. https://www.pearc18.pearc.org/awards
  - ❑ Best Poster Award, Rocky Mountain Advanced Computing Consortium- RMACC17 and RMACC19
  - ❑ Best Teaching Assistant Award, Computer science Department, New Mexico State University, NM, USA, Fall 2014.

Topics of Research Interest

- Mohammed Tanash is currently a Computer Science Ph.D. candidate at Kansas State University under the supervision of prof Daniel Andresen (https://www.cs.ksu.edu/about/people/faculty/andresen/).

- High Performance Computing, and Machine Learning.

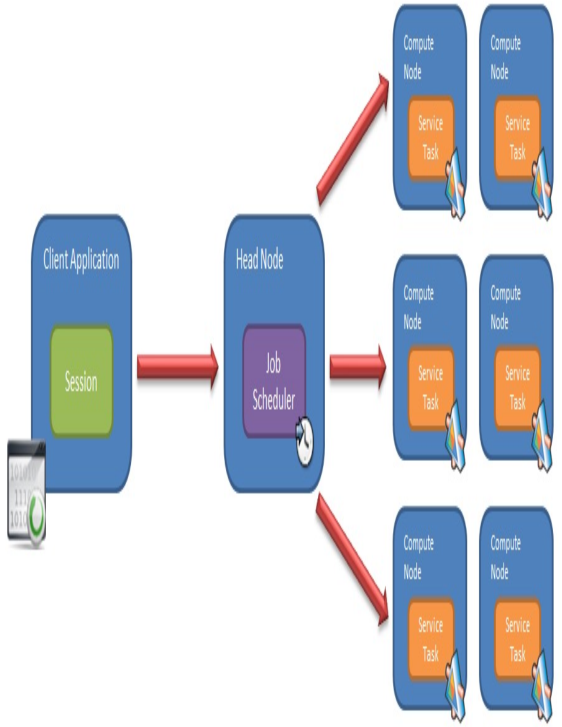
- At a high level, Mohammed's research is in the High Performance Computing (HPC) area. He's working specifically in improving the performance of the HPC systems using the techniques of Machine Learning. Currently, our group is working on improving the performance of Slurm workload manager by predicting the job resources needed for the submitted jobs on the cluster using Machine Learning techniques.

Introduction

- becomes more available and widespread

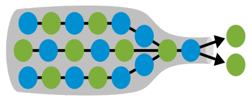
- The majority of our HPC come from other disciplines than Computer Science

- High Performance Computing (HPC) users rely on running their extensive computations on HPC clusters



Introduction

- **Scheduler** is the software system that controls what jobs runs where in the HPC systems

- HPC scheduling becomes a  of the HPC systems

- Improving HPC scheduling performance leads
  - Higher throughout
  - Shorter response time

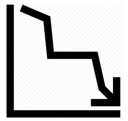
Introduction

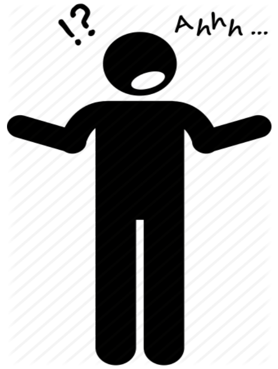
- In this paper we introduce the first ever open source, stand alone, highly accurate, fully-offline, and fully automated tool called (AMPRO-HPCC)

  - Help HPC users determine the allocation of HPC resources needs (memory and time) using supervised ML over Slurm historical data (sacct).
  - Our open-source tool can be found on GitHub using the following link (https://github.com/tanash1983/AMPRO-HPCC).

Problem Statement

- HPC users have difficulties to decide the required amount of resources for their submitted jobs on the cluster
- There is no software that can help to predict and determine the needed resources required for submitted jobs
- Users are encouraged to over-estimate resources for their submitted jobs
  - Waste and devours HPC resources
  - Inefficient cluster utilization





Research Goals

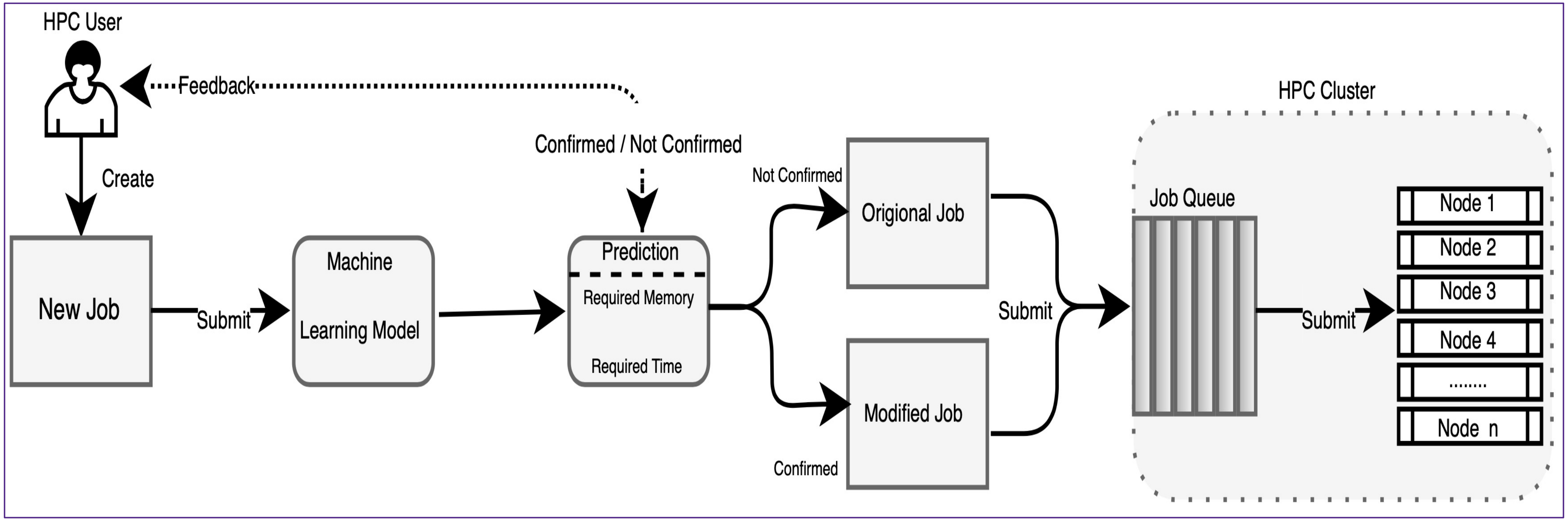
- ❑ Implement supervised machine learning tool for predicting resources needed (Memory, and Time) for submitted jobs

- ❑ Study the potential benefits of using supervised machine learning system for predictive analytics on the Slurm
  - ❑ Increase the efficiency of the scheduler and HPC systems
  - ❑ Decrease average job turnaround time
  - ❑ Decrease average job wait time
  - ❑ Increase system utilization

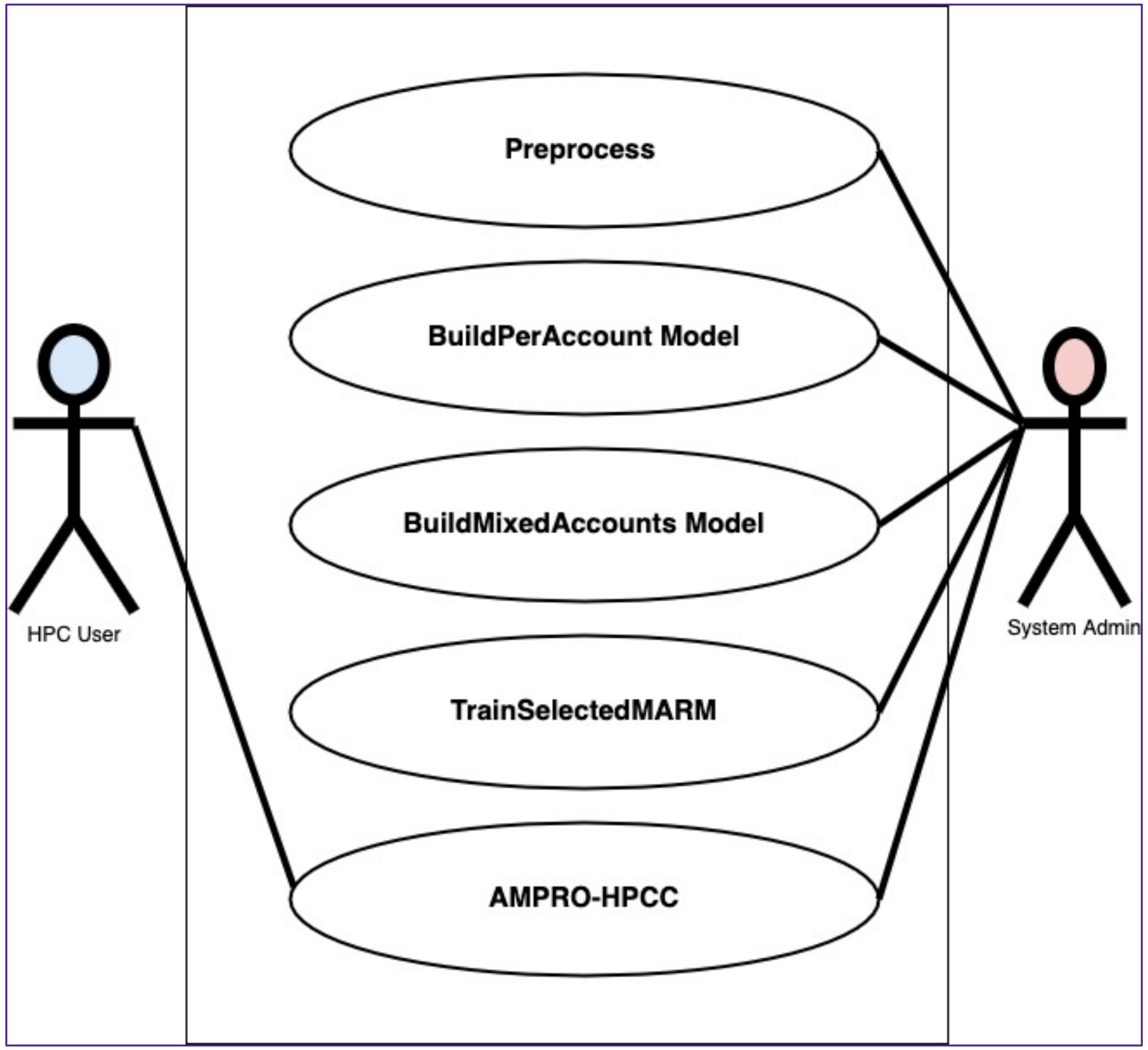
Slurm Simulator

- ❑ Modification to the actual Slurm code
- ❑ Supports most scheduling related options
- ❑ Inherits the actual Slurm behavior
- ❑ Slurm simulator was chosen because it is implemented from a modification of the actual Slurm code while disabling some unnecessary functionalities which do not affect the functionality of the real Slurm,.
- ❑ Perform a several months workload simulation in several real days for a large HPC system (17 days of work in an hour)

AMPRO-HPCC Workflow Model



AMPRO-HPCC Use-Case Diagram



Data Preparation (Selected Features)

Feature	Type	Description
Account	Text	Account the job ran under
ReqMem	Text	Minimum required memory for the job (in MB per CPU or MB per node).
Timelimit	Text	Timelimit set for the job in [DD-[HH:]]MM:SS format. (Predicted)
ReqNodes	Numeric	Requested number minimum Node count
ReqCPUS	Numeric	Number of requested CPUs
QOS	Text	Name of Quality of Service.
Partition	Text	The partition on which the job ran
MaxRSS	Numeric	Maximum resident set size of all tasks in job (in MB). (Predicted)
CPUTimeRaw	Numeric	Time used (Elapsed time * CPU count) by a job (in seconds)
State	Text	The job status

Data Treatment and Filtration

- ❑ Dealt with missing values (NaN) in the data and removed certain types of jobs.
- ❑ Jobs missing values for either MaxRSS or CPUTimeRAW were removed.
- ❑ Jobs belonging to priority access Partition / QOS were removed.
- ❑ Jobs with incomplete State (‘Cancelled’, ’Failed’, ’Deadline’) were removed.

Feature Standardization.

- Timelimit was parsed to numeric hours.
- MaxRSS was standardized to gigabytes (GB).
- Account, QOS, and Partitions were factorized to unique integer codes.
- ReqMem was converted from MB per CPU (suffix c) or MB per node (suffix n) to a numeric total MB, and was subsequently standardized to GB.

Data Normalization

- ❑ Only Account, ReqMem, ReqNodes, Timelimit, QOS, MaxRSS, and CPUTimeRAW were selected for further processing and analysis.

- ❑ All seven features except QOS and Account were normalized by shifting to their respective means and scaling by their respective standard deviation, using the Standard-ScalarTransform() in Scikit-learn Python package
- ❑ Data Sets:
  - ❑ Before data Preparation and Feature Analysis
    - ❑ **Beocat** has 17**.6** million instances and cover the years from **2018 – 2021**
  - ❑ After data Preparation and Feature Analysis
    - ❑ **Beocat** has **7.8** million instances and cover the years from **2018 – 2021**

Regression Models

- ❑ Our objective was to model time (CPUTimeRAW) and memory(MaxRSS) as a function of requested parameters Account, Time-limit, ReqNodes, ReqMem, ReqCPUS and QOS.
- ❑ Thus, we considered the following seven popular regression models for the task:

1. Lasso Least Angle Regression (LL)
2. Linear Regression (LR)
3. Ridge Regression (RG)
4. Elastic Net Regression (EN)
5. Classification and Regression Trees (DTR)
6. Random Forest Regression (RFR)
7. LightGBM (LGBM)

- ❑ We used scikit-learn's implementation for all models and performance metrics.

Evaluating Mixed Account Regression Models

$$MARM(N,M,X,Y) = \begin{cases} N' & N = 1 \\ MARM(N-1,M,X,Y) \cup N' & \text{otherwise} \end{cases}$$

- Where N'∈N is the Account that results in the best over all aggregate score in terms of R^2 on training (R^2_{tr}) and testing(R2te) datasets and number of jobs (SN′), given by

$$N' = \arg\max_{n \in N}(R2_{tr}(M, X_{A[n]}, Y_{A[n]}), R2_{te}(M, X_{A[n]}, Y_{A[n]}), S_{A[n]})$$

- Where $X_{A[n]}$ and $Y_{A[n]}$ correspond to independent and dependent variables, respectively for an unique AccountA[n].
- The MARM of N accounts depends upon the MARM of N−1 accounts appended with the best overall Account N′ that results in the best overall performance.

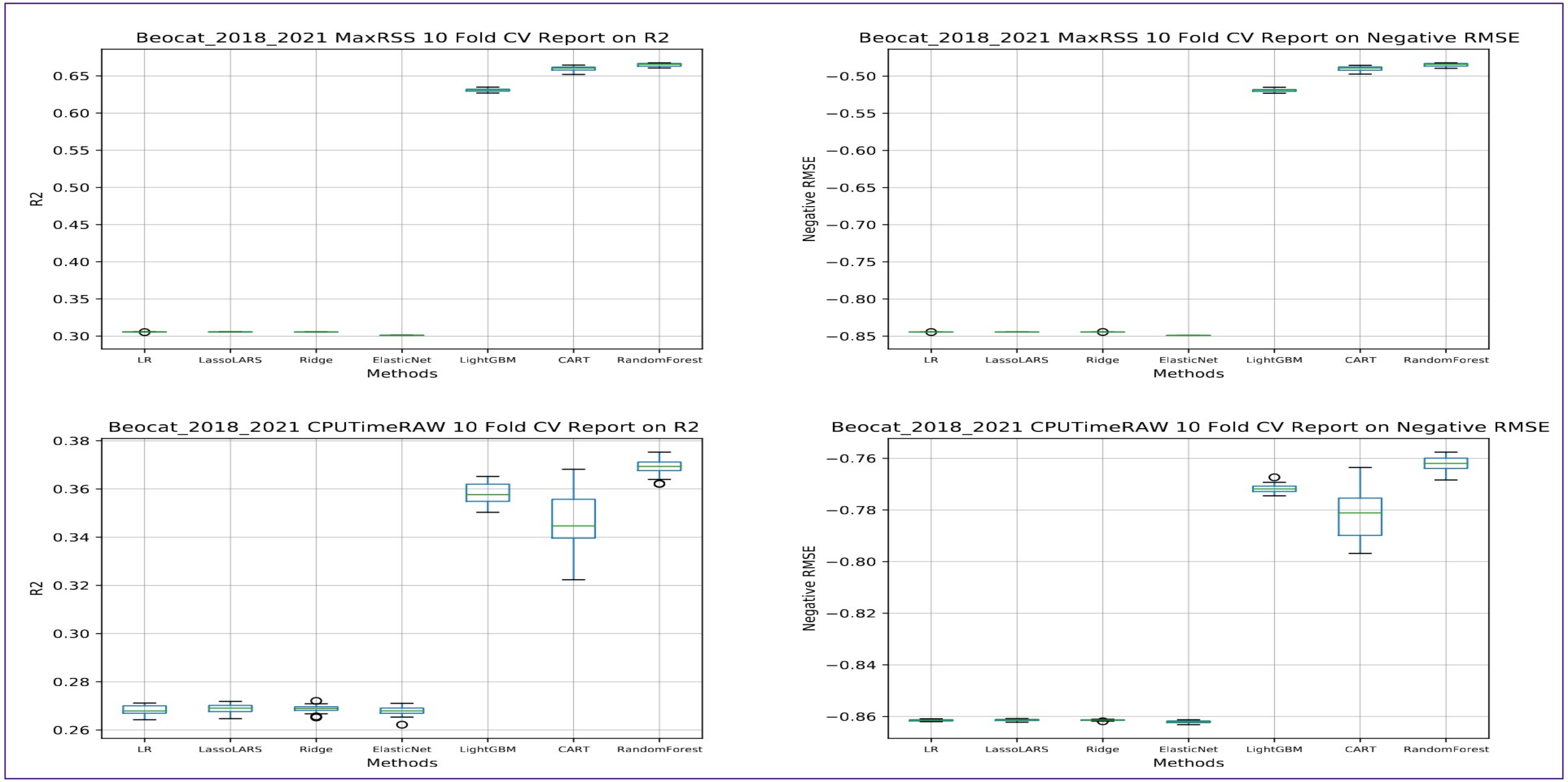
- A com- prehensive explanation of Mixed Account Regression Model (MARM) can be found in our publication "**M. Tanash, H. Yang, D. Andresen, and W. Hsu, "Ensemble prediction of job resources to improve system performance for slurm-based hpc systems," in Practice and Experience in Advanced Research Computing, 2021, pp. 1–8**".

Evaluating Our Model

- ❑ **What are we going to measure?**
  - ❑ Coefficient of determination (R^2)
  - ❑ Root mean squared error (RMSE) to evaluate the regression models.
  - ❑ Submission and Execution Time
  - ❑ System Utilization
  - ❑ Backfill-Sched Performance
  - ❑ Average Waiting and Turnaround Time

- ❑ **Compare the results of :**
  - ❑ Running each testbed using user **requested** memory and time
  - ❑ Running each testbed using the **actual** memory usage and duration
  - ❑ Running each testbed using **Predicted** memory and time

Results for R^2 and RMSE Across All Accounts



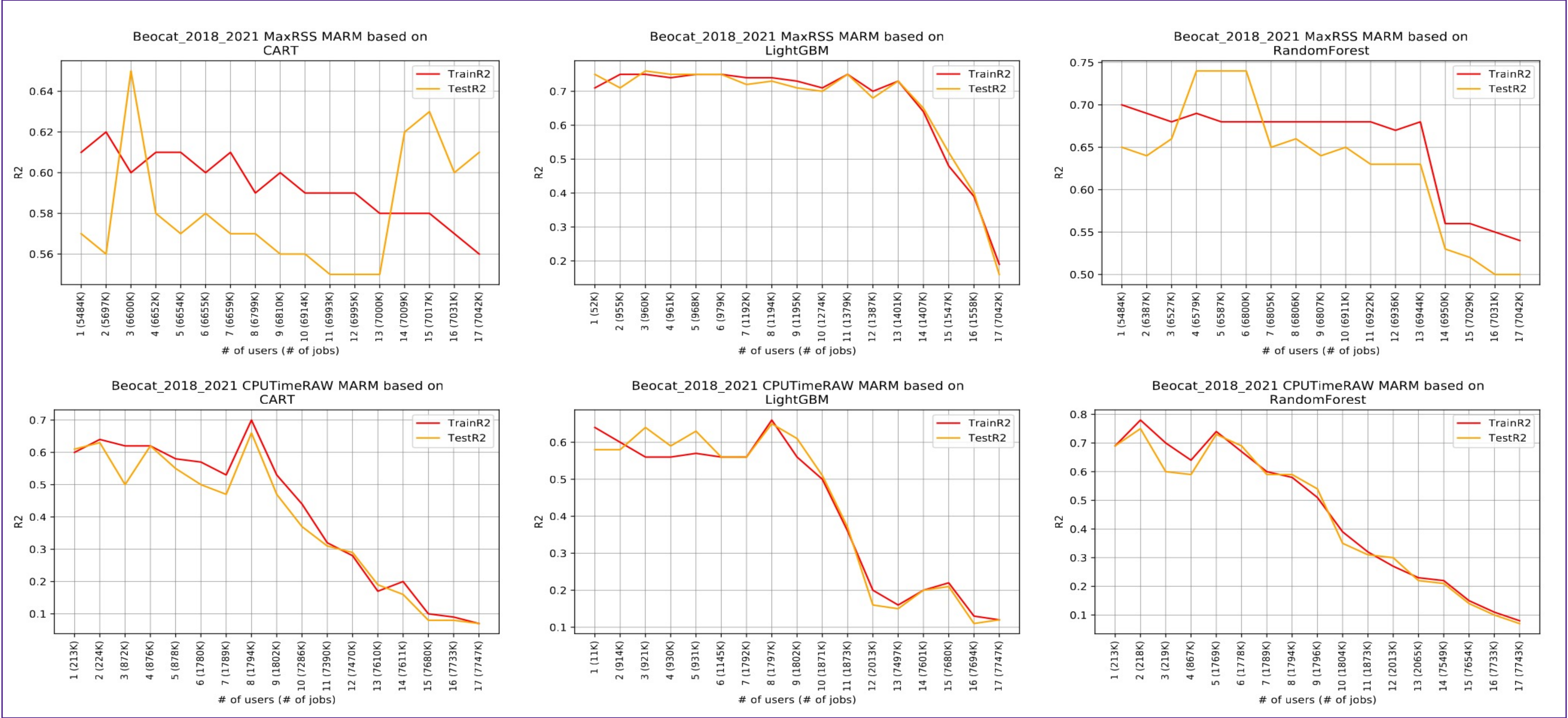
R^2 and Negative RMSE of Seven Methods Across 21 Accounts in BEOCAT

Kansas State University | Computer Science

IARIA

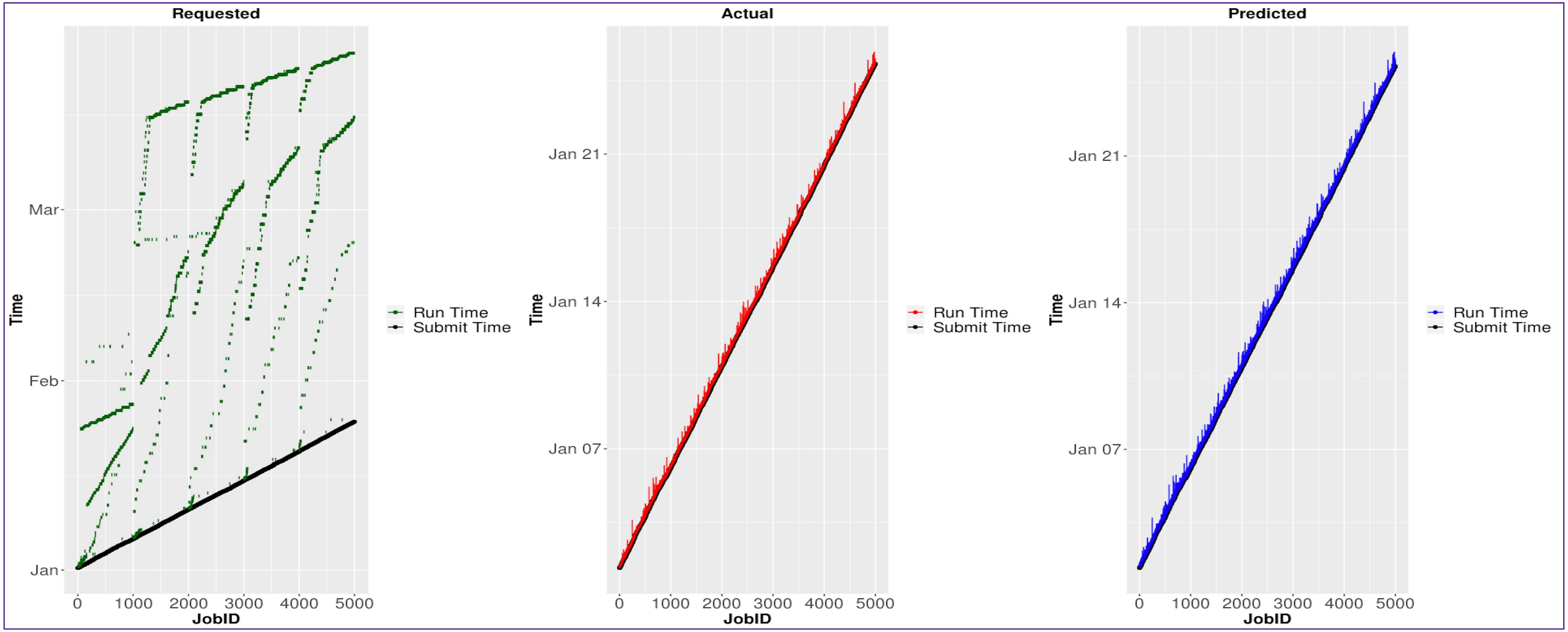
NIH NSF K-STATE

Results for R^2 and RMSE Using MARM



R^2 Versus Number of Accounts in Predicting Memory and Time Using MARM Across BEOCAT

Results for Beocat Resources

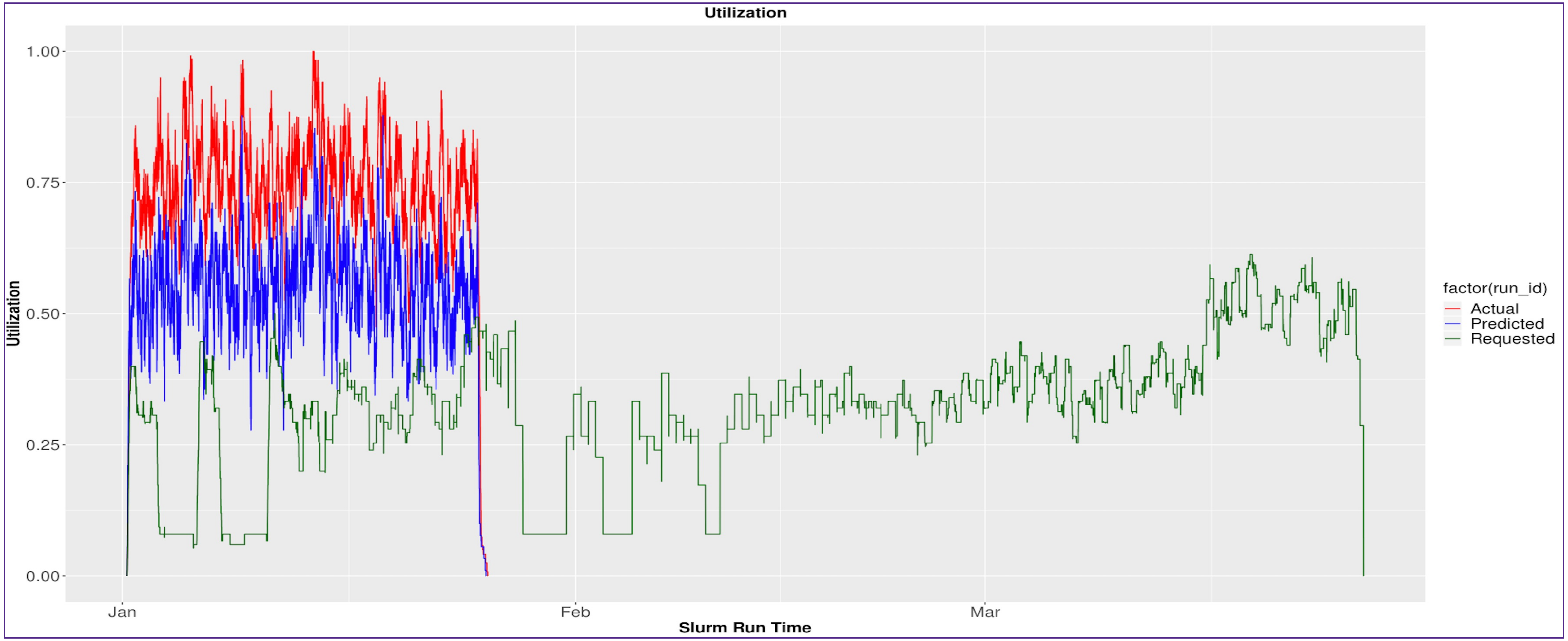


Jobs Submission and Running time (Requested vs Actual vs Predicted) for Jobs in Beocat

Note dramatic improvement of Y axis range between graphs

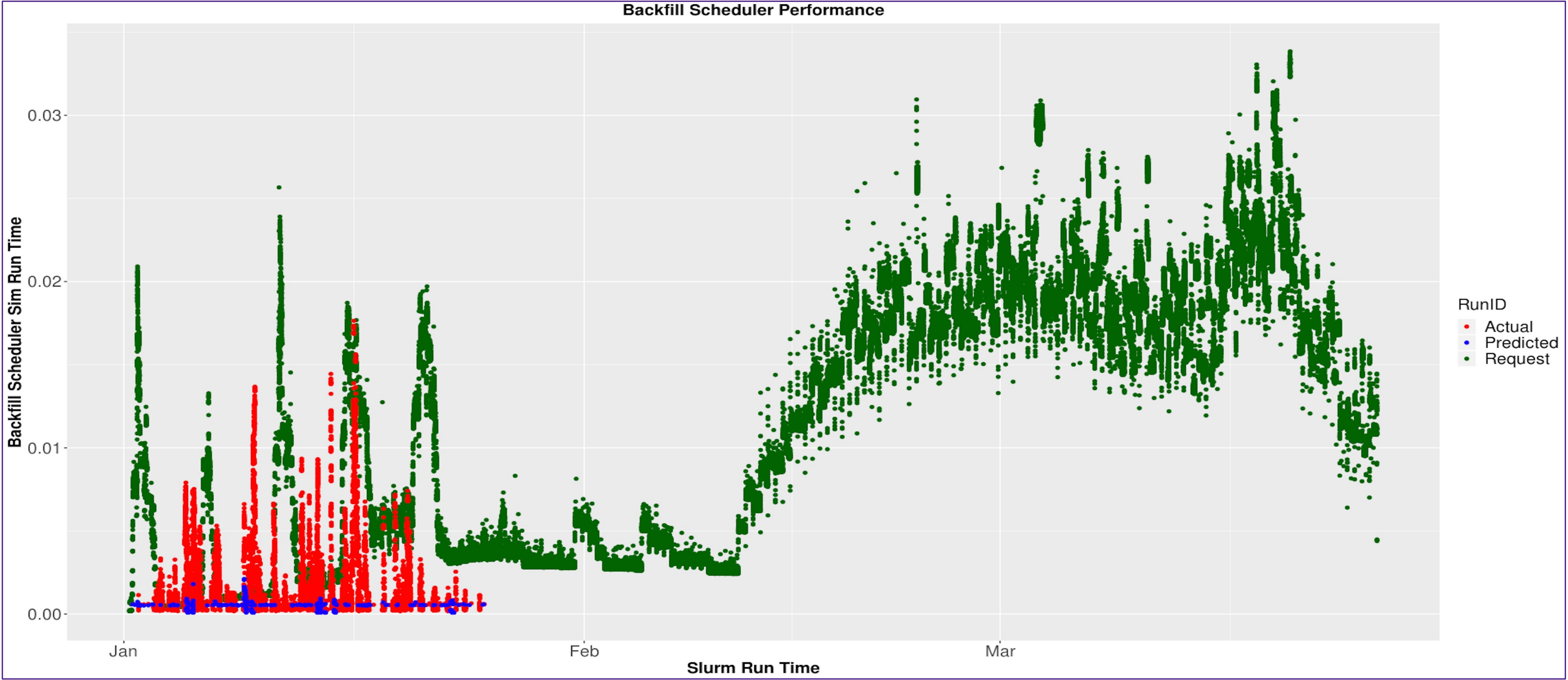
Kansas State University | Computer Science

Results for Beocat Resources



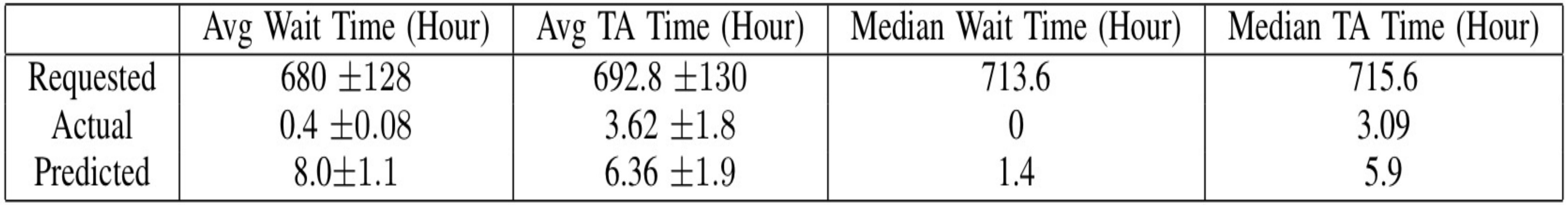
Utilization (Requested vs Actual vs Predicted) for for Jobs in Beocat

Results for Beocat Resources



Backfill-Sched Performance (Requested vs Actual vs Predicted) for for Jobs in Beocat

Results for Average Waiting and Turnaround Time



	Avg Wait Time (Hour)	Avg TA Time (Hour)	Median Wait Time (Hour)	Median TA Time (Hour)
Requested	680 ±128	692.8 ±130	713.6	715.6
Actual	0.4 ±0.08	3.62 ±1.8	0	3.09
Predicted	8.0±1.1	6.36 ±1.9	1.4	5.9

Average Waiting and Turnaround Time (Requested vs Actual vs Predicted) For Beocat

Conclusion

- ❑ Determining the allocation of HPC resources for submitted jobs is a difficult process for HPC users.

- ❑ It is still an open question on how much resources the user should specify (memory and time) for their submitted jobs on the cluster.

- ❑ We have developed a novel and the first ever open-source, stand-alone, fully automated, accurate, and fully-offline ML tool to help HPC users to determine the amount of required resources for their submitted jobs.

- ❑ Our tool is built based on implementation of different machine learning algorithms (Six discriminative models from the scikit-learn and Microsoft Light-GBM) applied on the historical data (sacct data) from Slurm.

- ❑ Our tools helps to reduce computational time, increase utilization of the HPC system, decrease average waiting time, and decrease the average turn-around time for the submitted jobs. Which leads maximize efficiency and decrease the power consumption of the cluster.

Thank You