Are We There Yet?

Predicting the Queue Wait times and Job Runtimes for HPC Jobs

Christin Whitton | Georgia Institute of Technology | HPC-DES
Mentors: Nathan DeBardeleben, Vanessa Job | HPC-DES

Motivation

TO OPTIMIZE FUNCTIONALITY IN HPC

This project used **historical SLURM data** from the Grizzly cluster to explore machine learning methods. The goal was to **predict** both the **job runtime** and the **queue wait time** (the time a job waited in a queue before running). We also used job scheduler simulation data to look at the queue wait times.

75% of users use only 15.9% of their requested wallclock limit

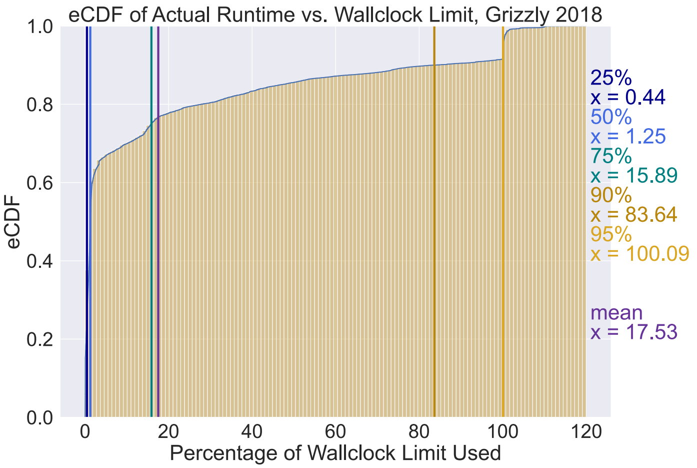
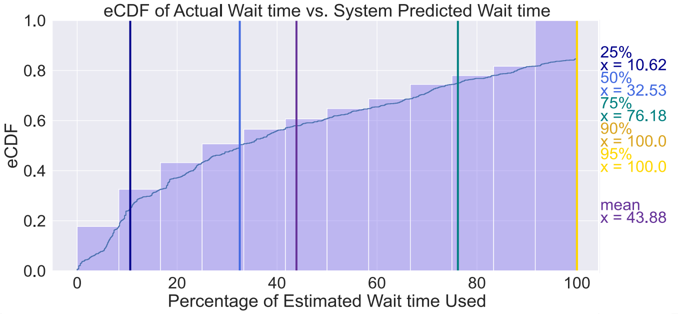


Chart 1. An empirical cumulative distribution showing the percentage of requested wallclock limit actually used for jobs.

Chart 2. An empirical cumulative distribution showing the percentage of the system-predicted queue times the job waited.



Methodology

THE DATA

- A colleague **derived variables** to **reflect the work in the queue** when a job is submitted, including % of queue utilization. We also extracted **temporal variables**.
- Since this **data is temporal**, we divided into training and testing data **chronologically**.

THE MODELS

- **Feature Selection** – used visual inspection, correlation matrices, and then found the "best" combination of variables by optimizing models with Optuna.
- **Model Selection** – used regression models, with Root Mean Squared Error (RMSE) as the performance indicator**.**
- **Model Tuning** – used Optuna to tune the hyperparameters for each method and model.

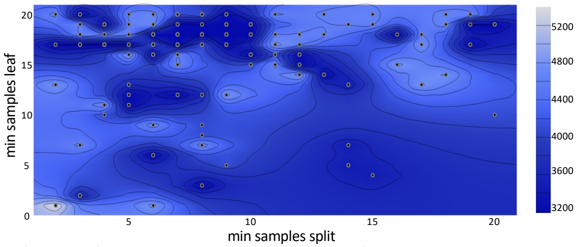
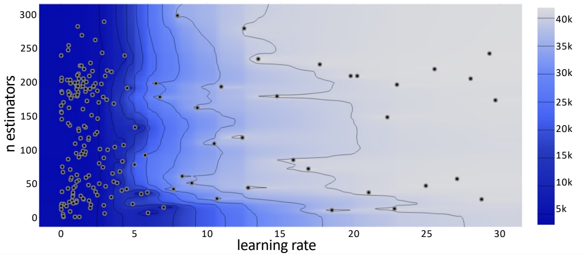


Chart 3. Optuna hyperparameter tuning for a decision tree model, involving 200 models run – this combination of hyperparameters has many options, which reflects the high variance of Decision Trees.

Chart 4. Optuna hyperparameter tuning for an Adaboost model – each point represents a model run. This combination of hyperparameters shows that the lower learning rate is important, which makes sense since too high of a contribution for weak learners could result in overfitting.



KEY TAKEAWAYS

- 🚐 **ML IS BETTER THAN USERS AT PREDICTING JOB RUNTIME**
- 🚐 **ML IMPROVES ON SYSTEM PREDICTED QUEUE TIMES**
- 🚐 **ADABOOST AND XGBOOST WERE CONSISTENTLY THE MOST EFFECTIVE ALGORITHMS**
- 🚐 **THE TEMPORAL AND QUEUE PRESSURE VARIABLES WERE HELPFUL**

Results

The models and their results are listed in the table below. With RMSE, **lower is better**. To evaluate results, we **compared** the **model RMSE**s with what the RMSE is with **the user predicted value**.

- Grizzly 2018 job runtime: 558.76
- Grizzly 2022 job runtime: 1419.34

RMSE by Data, Target Variable and Machine Learning Method		DATA/TARGET VARIABLE			
		Grizzly 2018 Minutes in Queue	Grizzly 2022 Minutes in Queue	Grizzly 2018 Runtime Minutes	Grizzly 2022 Runtime Minutes
Machine Learning Method	Decision Tree	3185.97	**1260.51**	284.54	249.19
	Random Forest	3207.85	1315.86	275.47	240.76
	SVR (Linear)	3605.69	1505.42	280.38	310.46
	SVR (RBF)	4417.22	1536.7	284.64	318.69
	Adaboost	**3104.81**	1262.46	277.30	**228.01**
	XGBoost	3132.14	1298.98	**272.43**	248.63

FUTURE WORK

- Explore more data!
- Use job simulation data to track queue wait times when different job events occur.
- Look further into SVR – why did it perform poorly here?
- Build a user interface with model results.

Los Alamos NATIONAL LABORATORY
NNSA National Nuclear Security Administration
Managed by Triad National Security, LLC, for the U.S. Department of Energy's NNSA.
LA-UR# 24-28283
Georgia Institute of Technology