High Performance Computing Queue Time Prediction using Clustering and Regression

Scott Hutchison[1][0000−0002−6698−3033], Daniel Andresen[1], Mitchell Neilsen[1], William Hsu[1], and Benjamin Parsons[2]

[1] Kansas State University, Manhattan KS 66506, USA
scotthutch@ksu.edu, dan@ksu.edu, neilsen@ksu.edu, bhsu@ksu.edu
[2] Engineering Research and Development Center, Vicksburg, MS 39180, USA
ben.s.parsons@erdc.dren.mil

Abstract. High Performance Computing (HPC) users are often provided little or no information at job submission time regarding how long their job will be queued until it begins execution. Foreknowledge of a long queue time can inform HPC user's decision to migrate their jobs to commercial cloud infrastructure to receive their results sooner. Various researchers have used different machine learning techniques to build queue time estimators. This research applies the proven technique of K-Means clustering followed by Gradient Boosted Tree regression on over 700,000 jobs actually submitted to an HPC system to predict a submitted job's queue time from HPC system characteristics and user provided job requirements. This method applied to HPC queue time prediction achieves better than 96% accuracy at classifying whether a job will start prior to an assigned deadline. Additionally, this research shows that historic HPC CPU allocation data can be used to predict future increases or decreases in job queue time with accuracy exceeding 96%.

1 Introduction

When a job is submitted to a High Performance Computing (HPC) cluster, a scheduling application, like SLURM, PBS, LoadLeveler, etc., handles the allocation of HPC resources in the future to the job's requirements as specified by the submitter. If adequate HPC resources are currently unavailable, the job enters a queue for execution in the future, and future resources are scheduled for that job's use. While a job is queued and awaiting execution, the job is making no forward progress toward its eventual completion. Worse still, it is often unclear to the user how long it will take until the job begins execution. The user knows the job is waiting to start, but there is often no way for the user to know if the job execution will begin in three hours, three days, or three weeks. Users with a time-critical application facing long queue delays may be willing to migrate jobs to commercial cloud infrastructure, like Amazon's AWS, Microsoft's Azure, Google's Cloud Computing, etc. Various techniques for predicting job queue times have been implemented in the past with different trade offs and accuracy. A machine learning pipeline which uses unsupervised K-Means clustering followed by Gradient Boosted Tree Regression has been used to lower

error rates when used in other applications. This research investigates if this machine learning technique can also be used to a predict queue times for HPC jobs.

The goal of this research is to answer the following questions: For an HPC system with a given current utilization, can we provide an accurate queue time prediction for a job which factors in the future state of the HPC cluster? Can we predict the execution of a job prior to an assigned deadline?

2 Background

Improving HPC utilization, decreasing job queue time, and decreasing job turnaround time are all active areas of research at Kansas State University (KSU). These areas would typically apply and are of interest to any HPC cluster manager. Developing an accurate queue time predictor can not only help inform job scheduling, but it can also provide additional information to users about their expected job start times, and perhaps more importantly, about the length of time they can expect to wait for their results.

Kumar and Vadhiyar [7] performed a similar job queue time prediction by using k-nearest neighbors followed by support vector machines to classify jobs into time bins of various sizes with their probabilities. Though this technique showed promise, using regression allows for a concrete prediction value for queue time, as opposed to the most likely time bin this job would fall into.

Jancauskas, et al. [6] conducted similar research on queue time prediction using Naive Bayes to return a list of probability estimates (t_i, p_i), where p_i was the probability that a job will start before t_i. They used similar features and achieved excellent accuracy, precision, and recall. An advantage of using clustering and regression over Naive Bayes is that a concrete start time prediction can be generated for the current load on the HPC system for an individual job with certain requirements. This research not only uses different machine learning techniques for prediction, but also factors in changes of the future state of the HPC system and the impact those changes have on job queue times, which Jancauskas, et al. considered outside the scope of their research.

Brown, et al. [2] used a very similar technique as this research, using k-nearest neighbors followed by gradient boosted tree regression, however they also did not factor in the future state of the HPC system.

Unsupervised K-Means [8] clustering followed by Gradient Boosted Tree Regression [3] (GBTR) has been used by various researchers to improve the accuracy of regression models. For instance, Zheng and Wu [4] used this technique to improve on short-term wind forecasting, and Liu et al. [9] used this technique to improve short-term power load forecasting. As this technique has shown promise in improving prediction accuracy in other areas of research, using clustering followed by regression could also improve the accuracy of predicting how long a job will be queued for execution on an HPC system.

3 Methodology

3.1 Data set

The HPC cluster at KSU is a Beowulf [1] HPC cluster called "Beocat". Beocat currently consists of 362 compute nodes with a total of 10980 compute cores and 5.57 Terabytes of memory, and it uses SLURM [16] as the job scheduler. SLURM logs data from all jobs submitted to Beocat and retains 105 different features about each job. These features include job submission time, start time, end time, the number of CPUs requested by the user, the amount of memory and time requested by the user, etc. The data set used for this research consisted of all jobs submitted in 2018, which totaled approximately 730,000 jobs. Figure 1 shows the CPU and memory allocation over time for Beocat for 2018. The calculated CPU utilization for 2018 was roughly 60%. Jobs can remain queued due to lack of available CPUs or lack of available memory. This data set was thought to be a good representative data set with enough data to produce meaningful results.

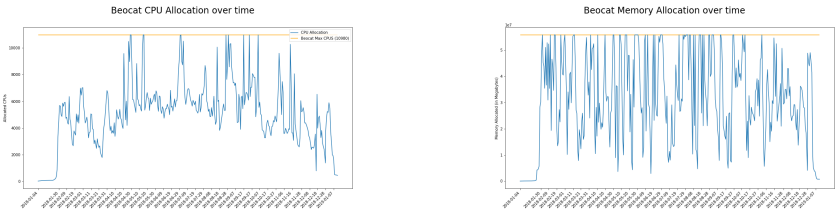

Fig. 1: CPU and memory allocation over time for KSU HPC system for 2018

3.2 Feature Selection and Calculation

The queue time of a job depends primarily on two factors: the amount of resources available in the HPC and the amount of resources a job is requesting. Table 1 summarizes and describes the features used for this research.

To calculate `BeocatCPUsInUse`, the 2018 jobs from the log data were sorted chronologically by their start and end times. Each time a new job began, the number of CPUs in use by the cluster was increased by the number of cores allocated to that job. Each time a job ended, the number of CPUs in use by the cluster was decreased by the number of cores allocated the that job. The same strategy was employed to calculate `BeocatMemoryInUse`.

To calculate `QueueDepth`, jobs were sorted by their submit times and their start times. Each time a job is submitted, the queue depth is increased by one. Each time a job starts, the queue depth is decreased by one.

The requested CPUs, requested memory, and requested time for each job were directly pulled from the log data to populate the `ReqCPUs`, `ReqMem`, and `ReqMinutes` features.

Table 1: Features used

Category	Feature	Description
HPC features	`BeocatCPUsInUse`	Current allocated CPUs
	`BeocatMemoryInUse`	Current allocated memory
	`QueueDepth`	The queue depth when job was submitted
Job features	`ReqCPUs`	Number of requested cores for a job
	`ReqMem`	Amount of memory requested for a job
	`ReqMinutes`	Amount of minutes requested for a job
	`OwnsResources`	True if user has priority access to compute nodes; False otherwise
Dependent variable	`QueueTimeInSec`	Number of seconds from submit until start

There are a number of compute nodes which are available for all Beocat users to use. Certain resources are owned by departments whose members have priority access and who can preempt running jobs. The `OwnsResources` feature was set to true if there were dedicated resources available which could run that job. If the job was submitted to only the queues common to all, the `OwnsResources` feature was set to false.

To calculate `QueueTimeInSec`, the submit time for each job was subtracted from its start time. This time delta object was converted into an integer that represented the number of seconds each job sat in the queue awaiting job execution.

3.3 Feature Normalization and Model Development

Min-Max scaling was used on Beocat CPU and memory allocation to return a value between 0 and 100 which represents the percentage of Beocat currently allocated.

A vector consisting of `ScaledBeocatCPUsInUse`, `ScaledBeocatMemoryInUse`, `QueueDepth`, `ReqCPUs`, `ReqMem`, and `OwnsResources` was constructed, and used to predict `QueueTimeInSec`. The log data containing roughly 730,000 jobs were randomly split into an 80% training batch and a 20% testing batch. The training batch contained roughly 583,000 jobs, and the test batch contained roughly 146,000 jobs.

A base GBTR model was trained using 5-fold cross validation on the training data, which was then evaluated using the test data. This base model was used later to predict clusters that contained fewer than 100 elements. This model will be referred to as the GBT_{base} in various figures throughout the remainder of this report.

Since the optimal number of clusters required to group the training data was initially unclear, iterative K-Means was used to cluster the data using an increasing number of clusters from 2 to 150. The training data was fed into K-Means and n clusters were returned, where $n = 2, 3, 4, \ldots, 150$. After clustering, a GBTR model was developed using 5-fold cross validation for each each cluster containing more than 100 elements. A small cluster containing fewer than 100

elements would use the GBT_{base} model to make queue time predictions. These models were developed using the training data, and then evaluated using the test data. An unseen-before test input would first be classified by the K-means model, and then the appropriate GBTR model was used to develop a prediction of the job's queue time. The machine learning pipeline's error rate overall on the test data was used to determine an ideal number of clusters that minimized the error. The number of clusters producing a local minimum error rate was identified, and then that pipeline was selected for further evaluation. This machine learning pipeline is outlined in Figure 2.

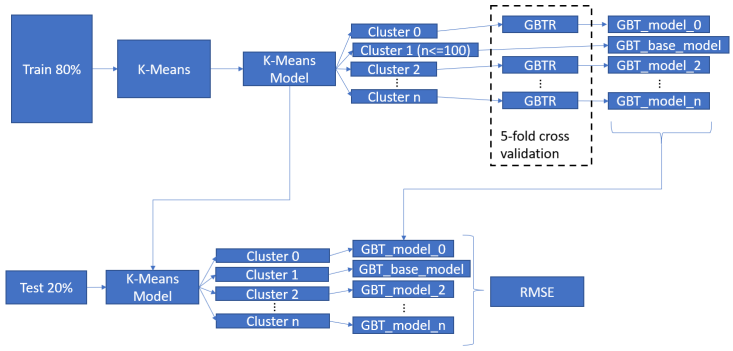

Fig. 2: The machine learning pipeline used for this research.

During actual use, users may have individual and specific deadlines. This information is not currently solicited or collected on Beocat, so it was unavailable in the log data. An arbitrary deadline for each job was set to be the average queue time for all jobs in 2018, or 13423 seconds (HH:MM:SS = 03:43:43). A queue time prediction was made for each job, and it was assessed whether this prediction was met or exceeded the assigned deadline. Since the actual queue time was known, a confusion matrix was generated to determine the overall accuracy, precision, recall, and F1 scores for the machine learning pipeline.

The above mentioned queue time prediction represents a snapshot in time given the overall HPC system resources allocated for a job with specific requirements. The future state of the HPC may also impact job queue time. For instance, if cluster allocation increases following a job submission when a large number of higher priority jobs are started, this queue time estimate may underestimate when a job would actually begin. Alternatively, an HPC allocation decrease following a job submission due to jobs finishing earlier than expected may cause a job to begin sooner.

Since the average queue time for Beocat in 2018 was roughly 3 hours and 45 minutes, a sliding time window of 4 hours was used to assess what impact a change in HPC CPU allocation would have on the change in queue times for HPC jobs. Figure 3 depicts how average Δ_{CPUs} was calculated. Average Δ_{queue_time}

was calculated in the same manner. In 2018, there were 1,520 four-hour time windows containing submitted jobs. The time windows were randomly split into an 80% training batch and 20% testing batch. A linear regression model was trained using the average Δ_{CPUs} from the training data and used to predict the average Δ_{queue_time} of the test data. Again, the actual change in queue time was known from the log data, which enabled the calculation of RMSE, accuracy, precision, recall and the F1 Score for this linear regression model.

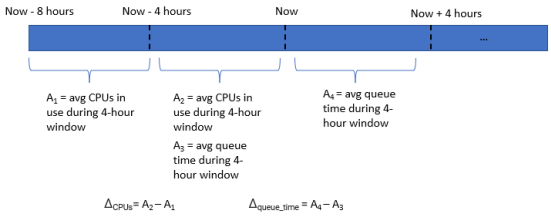

Fig. 3: Depiction of Δ_{CPUs} calculation

3.4 Evaluation

Feature correlation was measured using the Pearson Correlation Coefficient [12]. This statistical measure produces a value between -1 and 1, where correlation coefficient values closer -1 or 1 indicate a stronger correlation between two features and a value closer to 0 indicates no or very little correlation.

Each regression model contained some N elements. The machine learning pipeline and each regression model was evaluated using the Root Mean Squared Error (RMSE) metric, which is calculated according to the following equation:

$$RMSE = \sqrt{\frac{\Sigma_{i=0}^{N}(\text{actual queue time}_i - \text{predicted queue time}_i)^2}{N}}$$

Additionally, the machine learning pipeline was used to compare whether or not the predicted queue time for each job exceeded the assigned deadline. A confusion matrix, along with the metrics of accuracy, precision, recall, and the F1 Score were utilized. The metrics and their descriptions are laid out in Table 2:

The metrics used to assess the change in queue time given the change in CPU allocation are laid out below in Table 3:

These metrics were calculated in the following way:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{Precision} = \frac{TP}{TP+FP}$$
$$\text{Recall} = \frac{TN}{TP+FN} \quad \text{F1 Score} = \frac{2*TP}{2*TP+FP+FN}$$

Table 2: Metrics for Deadline Classification

Metric	**Description**
True Positive (TP)	Model predicts job will start before deadline, and it does
True Negative (TN)	Model predicts job will start after deadline and it does
False Positive (FP)	Model predicts job will start before deadline, but job does not
False Negative (FN)	Model predicts job will start after deadline, but job does not

Table 3: Metrics for Future Queue Time Classification

Metric	**Description**
True Positive (TP)	Model predicts average Δ_{queue_time} will decrease 4 hours from now, and it does
True Negative (TN)	Model predicts average Δ_{queue_time} will increase 4 hours from now, and it does
False Positive (FP)	Model predicts average Δ_{queue_time} will decrease 4 hours from now, and it does not
False Negative (FN)	Model predicts average Δ_{queue_time} will increase 4 hours from now, and it does not

4 Results

PySpark is an interface for Apache Spark [17] for the Python [15] programming language. PySpark was utilized for data wrangling and analysis. PySpark's machine learning library, MLlib [10], was utilized for statistical analysis, clustering, and regression tasks. Matplotlib [5] was used to generate plots and charts.

4.1 Correlation of Features

Table 4 lays out the Pearson Correlation Coefficients for the features used. "Slightly correlated" values in Table 4 are displayed using orange text and the stronger "somewhat correlated" features are displayed using red text. Perhaps unsurprisingly, there is a slight correlation between the HPC CPUs in use and the HPC memory in use at any given time, as well as a slight correlation between the amount of CPUs requested by a user and the amount of memory requested by a user. The queue depth and queue time are somewhat correlated, and the queue depth and the amount of memory allocated on the HPC are somewhat correlated. This makes sense given the relatively large amount of time Beocat spends with its allocated memory near or at its maximum (See Figure 1).

4.2 K-Means Clustering and GBT Regression

The GBT_{base} model had a RMSE of 23229.92. This was compared to two naive guessing strategies of guessing zero queue time for all jobs and guessing the average queue time from 2018 for all jobs. Naively guessing zero seconds produced a RMSE of 42818.2, and naively guessing the average queue time (13423.21

Table 4: Correlation of Features

Feature	BeocatCPUsInUse	BeocatMemoryInUse	QueueDepth	ReqCPUs	ReqMem	ReqMinutes	QueueTimeInSec
BeocatCPUsInUse	1	0.195	0.008	0.008	-0.010	-0.067	-0.009
BeocatMemoryInUse	0.195	1	0.392	-0.047	-0.020	-0.036	0.131
QueueDepth	0.008	0.392	1	-0.061	-0.028	-0.091	0.326
ReqCPUs	0.007	-0.047	-0.047	1	0.119	0.057	-0.002
ReqMem	-0.010	-0.020	-0.027	0.119	1	0.036	0.003
ReqMinutes	-0.067	-0.036	-0.091	0.057	0.036	1	0.074
QueueTimeInSec	-0.009	0.131	0.326	-0.002	0.003	0.074	1

seconds) produced a RMSE of 40659.8. It is clear that the base model has a lower RMSE than these two naive guessing strategies.

It was identified by iterating through the number of generated k-means clusters that 57 clusters produced a local minimum RMSE of 18119.23. As the number of clusters increased, there was not a significant improvement in accuracy, and it is thought that as the number of clusters continues to increase, the GBT_{base} model will be used for more and more clusters as the number of data points in each cluster decreases. Locating this "elbow" in the data [14] attempts to prevent overfitting and clustering beyond the point of diminishing returns. The RMSE of the machine learning pipeline as the number of clusters was varied is depicted in Figure 4.

Using 57 clusters produces 42 GBTR models for clusters containing more than 100 elements, and the machine learning pipeline uses the base GBTR model for the remaining 15 clusters. Each test data point was clustered, and then the appropriate GBTR model was used to predict the queue time for a job. Each job's queue time prediction was compared to its actual queue time, and it was evaluated if the predicted and actual queue time exceeded the assigned deadline. The confusion matrix and evaluation metrics can be found in Table 5. Overall, the machine learning pipeline was excellent at predicting future queue times, and its accuracy, precision, recall, and F1 Score were all greater than 96%.

Table 5: Confusion Matrix for Machine Learning Pipeline with Metrics

Total Jobs 145,658		**Actual**	
		Job runs before Avg Queue Time	Job runs after Avg Queue Time
Predicted	Job runs before Avg Queue Time	TP = 107,208	FP = 1,490
	Job runs after Avg Queue Time	FN = 3,366	TN = 33,594

Metric	**Value**
Accuracy	96.66%
Precision	98.63%
Recall	96.95%
F1 Score	97.79%

4.3 Future HPC Queue Time Prediction

The $(\Delta_{CPUs}, \Delta_{queue\_time})$ points and the line-of-best-fit provided by the linear regression model are depicted in Figure 5. The model achieved a RMSE

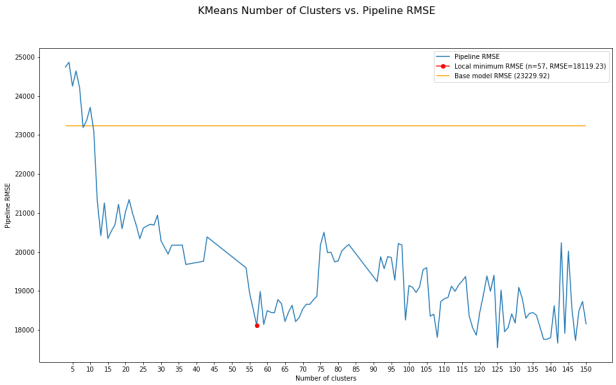

Fig. 4: RMSE of machine learning pipeline as the number of K-Means clusters was varied.

of 14691.17 seconds. The confusion matrix and evaluation metrics are found in Table 6. Overall, the linear regression model was excellent at predicting future queue times, and its accuracy, precision, recall, and F1 Score were all greater than 96%.

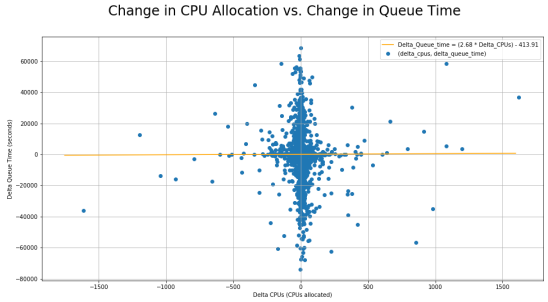

Fig. 5: Change in average CPU allocation vs. change in average queue time

5 Discussion

The correlation between queue depth and the HPC memory in use for Beocat is supported by the memory in use over time depicted in Figure 1. This confirms the observations made by Beocat's system administrators who have determined that more often than not, Beocat is constrained by its available memory rather than its available CPUs. This alone has informed the equipment requirements

Table 6: Confusion Matrix for Queue Time Increase with Metrics

Total Time Windows 1,520		**Actual**	
		Future average Δ_{queue_time} decreases	Future average Δ_{queue_time} increases
Predicted	Future average Δ_{queue_time} decreases	TP = 702	FP = 26
	Future average Δ_{queue_time} increases	FN = 24	TN = 768

Metric	**Value**
Accuracy	96.71%
Precision	96.43%
Recall	96.69%
F1 Score	96.56%

for purchases of new servers for the HPC system here at KSU. We now procure servers with larger memory to try to better accommodate our user's requirements. Doing a similar analysis might allow managers of other HPC systems to better identify hardware that can support the types of jobs their users often run.

As depicted in Table 5, the accuracy, precision, recall, and F1 Score were all greater than 96%. Although a somewhat arbitrary deadline was used for each user's deadline, this data could be provided by the users at submission time. This would give more meaningful information to the users of Beocat depending on how time sensitive their jobs are. Various other values for deadlines were used (1 hour, 8 hours, and 12 hours), all of which produced similar accuracy, precision, recall, and F1 scores exceeding at least 90%. It can only be concluded that the machine learning pipeline does a good job at predicting a reasonable start time for most jobs regardless of the pipeline RMSE.

Using clustering and regression as opposed to other techniques provides a concrete queue time estimate. The pipeline RMSE was roughly 5 hours of error, and the average queue time for jobs submitted to Beocat in 2018 was approximately 3 hours and 45 minutes. The HPC at KSU has comparatively low queue times for jobs, and other HPC clusters may have queue time measured in the range of days, or even weeks. An overall 5 hour error rate for the prediction for Beocat somewhat overshadows the average queue time in our case, but in other clusters, it might be more meaningful. In practice, queue times for Beocat are very left-skewed, and most of the jobs submitted to Beocat are executed after a very short period of time. Only very large jobs spend any significant amount of time in the queue waiting for resources.

It was shown that the average allocation of HPC CPUs over a 4 hour window was an effective predictor for an increase or decrease of future queue time for jobs. This information could further inform machine learning models attempting to predict queue time for jobs. For instance, the linear regression model could be run before the queue time deadline assessment to determine if this contributes to an increase in the accuracy of the queue time prediction from the machine learning pipeline.

Finally, this queue time estimation tool could inform a decision to migrate a job to cloud resources instead of facing a long queue delay. Okanlawon, et al. [11] conducted research to better inform a user's decision to either resubmit a job with different resources or migrate that job to commercial cloud infrastructure. An accurate queue time estimation tool could offer another data point informing a user's decision.

6 Conclusion and Future Work

This research demonstrated that clustering and regression can also be applied to the task of queue time estimation for HPC systems. The machine learning pipeline described in this paper was more than 96% accurate at classifying whether a job would start before an assigned deadline. A simple linear regression model also achieved greater than 96% when attempting to predict if future queue times will increase or decrease. These pieces of information could prove vital to a researcher with a time critical application. It is also a meaningful metric for all HPC users, so they will be better informed about the start times of their jobs.

Additional analysis is needed to determine why certain jobs were grouped together into the clusters provided by K-Means. This research fed the cluster and job feature vector into the K-Means algorithm in search of the number of cluster producing a local minimum error rate. It is thought that additional analysis of clusters might shed light onto what is causing certain kinds of jobs to queue for longer times. Are there certain characteristics of jobs that cause them to sit in the queue longer? Are there certain characteristics or limitations of the HPC cluster itself which is contributing to longer queue times? Could additional HPC user education or better documentation mitigate queue time in some way? These remain open questions.

Our experience has been that users tend to drastically overestimate their job requirements at submission times. There is very little downside for a user who overestimates their resources at submission time. However, there is a very large downside if a job is killed before completion due to a user requesting insufficient resources at submit time. In the aggregate, however, mass overestimation of required resources leads to longer queue times for all users, which can negatively impact user experience overall. Tanash, et al. [13] have looked to machine learning to determine how actual allocated resources compared to what users have requested at submit time. Since this queue time predictor relied upon user submitted requirements for each job, adding a more accurate estimate of actual resources used would presumably improve the accuracy of a model predicting queue time.

Finally, informing the machine learning pipeline with the future queue time prediction may further improve the accuracy of the prediction made by the machine learning pipeline. It remains to be seen if first applying the future state of the HPC queue time prediction has measurable impacts on the accuracy of the clustering and regression pipeline.

References

1. Becker, D.J., Sterling, T., Savarese, D., Dorband, J.E., Ranawak, U.A., Packer, C.V.: Beowulf: A parallel workstation for scientific computation. In: Proceedings, international conference on parallel processing. vol. 95, pp. 11–14 (1995)
2. Brown, N., Gibb, G., Belikov, E., Nash, R.: Predicting batch queue job wait times for informed scheduling of urgent hpc workloads. arXiv preprint arXiv:2204.13543 (2022)
3. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
4. Henriques, J., Caldeira, F., Cruz, T., Simões, P.: Combining k-means and xgboost models for anomaly detection using log datasets. Electronics **9**(7), 1164 (2020)
5. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
6. Jancauskas, V., Piontek, T., Kopta, P., Bosak, B.: Predicting queue wait time probabilities for multi-scale computing. Philosophical Transactions of the Royal Society A **377**(2142), 20180151 (2019)
7. Kumar, R., Vadhiyar, S.: Prediction of queue waiting times for metascheduling on parallel batch systems. In: Workshop on Job Scheduling Strategies for Parallel Processing. pp. 108–128. Springer (2014)
8. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern recognition **36**(2), 451–461 (2003)
9. Liu, Y., Luo, H., Zhao, B., Zhao, X., Han, Z.: Short-term power load forecasting based on clustering and xgboost method. In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). pp. 536–539. IEEE (2018)
10. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: Machine learning in apache spark. The Journal of Machine Learning Research **17**(1), 1235–1241 (2016)
11. Okanlawon, A., Yang, H., Bose, A., Hsu, W., Andresen, D., Tanash, M.: Feature selection for learning to predict outcomes of compute cluster jobs with application to decision support. In: 2020 International Conference on Computational Science and Computational Intelligence (CSCI). pp. 1231–1236. IEEE (2020)
12. Pearson, K.: Vii. note on regression and inheritance in the case of two parents. proceedings of the royal society of London **58**(347-352), 240–242 (1895)
13. Tanash, M., Dunn, B., Andresen, D., Hsu, W., Yang, H., Okanlawon, A.: Improving hpc system performance by predicting job resources via supervised machine learning. In: Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), pp. 1–8 (2019)
14. Thorndike, R.L.: Who belongs in the family. Psychometrika pp. 267–276 (1953)
15. Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)
16. Yoo, A.B., Jette, M.A., Grondona, M.: Slurm: Simple linux utility for resource management. In: Workshop on job scheduling strategies for parallel processing. pp. 44–60. Springer (2003)
17. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., et al.: Apache spark: a unified engine for big data processing. Communications of the ACM **59**(11), 56–65 (2016)