Poster submitted to:
sc18.supercomputing.org/

Cloud Crystal - The HPC resource forecaster for Epic

James Sharpe[a], David Standingford[a], Mike Turner[a],
Marco De Angelis[b,c], Alejandro Diaz[b], Alfredo Garbuno[b], Peter Hristov[b], Bright Uchenna Oparaji[b], Edoardo Patelli[b], Jonathan Sadeghi[b]

[a]Zenotech ltd., *Bristol and Bath Science Park, Dirac Crescent, Emersons Green, Bristol BS16 7FR, UK*
[b]Institute for Risk and Uncertainty, University of Liverpool, *Chadwick Building, Peach Street, L69 7ZF, Liverpool, UK*
[c]marco.de-angelis@liverpool.ac.uk

ABSTRACT

The aim of this work is to forecast the queue time of jobs in the EPIC portal to supercomputing and cloud HPC. The implementation will provide the user with recommendations about what resource the job should be submitted to in order to minimize the queue time. As a result the user will be able to understand the cost implications prior to job submission. Ultimately, the uncertainty / certainty obtained with the prediction will help the users better plan their time and manage the available financial resources.

EPIC connects to a wide range of HPC resources. It can be used as a simple submission interface to specialist supercomputing resources or as a tool to quickly run your own HPC cluster. Access world class supercomputing and cloud HPC resources in one place.

https://epic.zenotech.com/

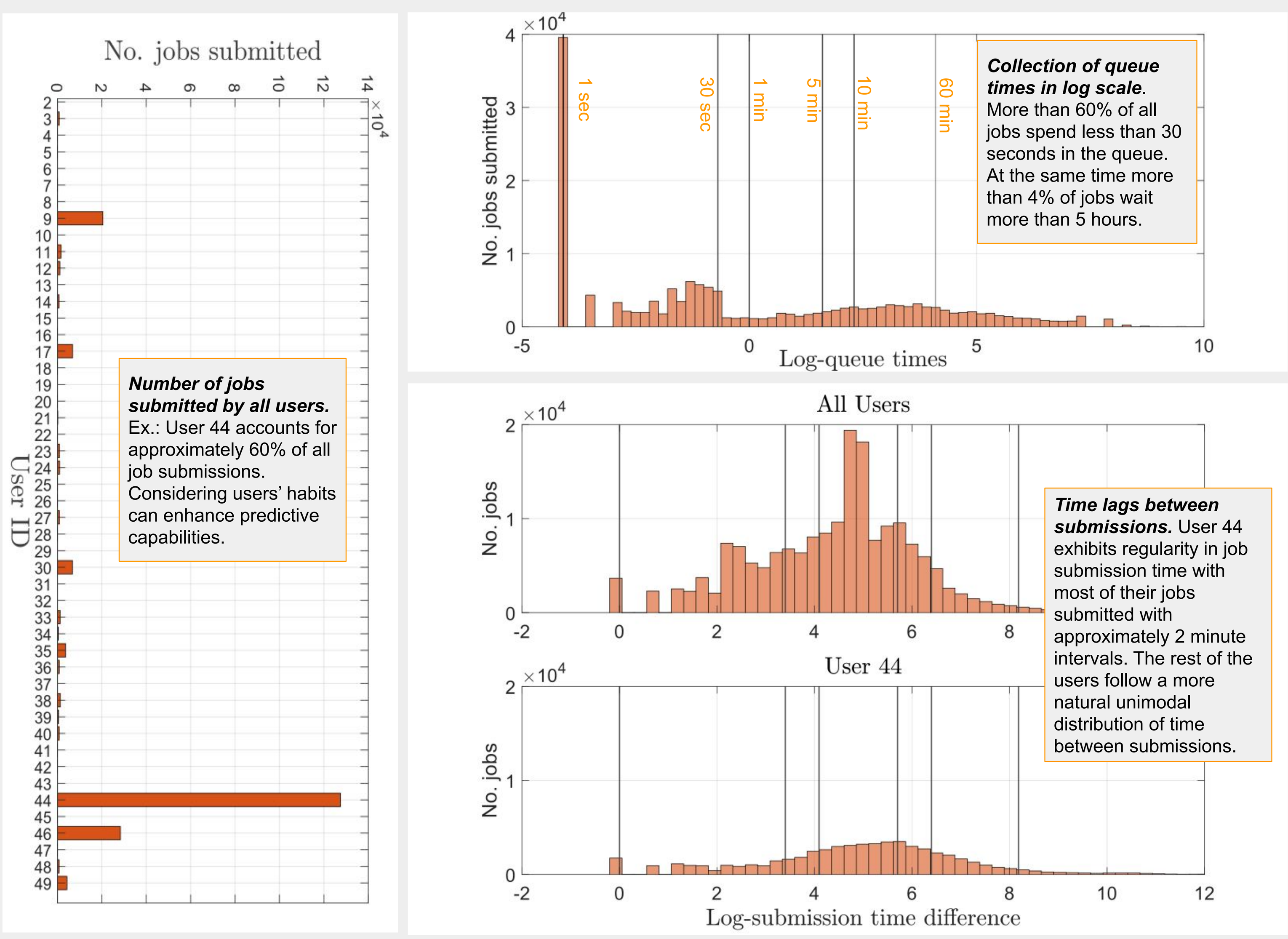
GOAL

We present the user with an estimation of queue time for each resource and we make it accessible before job submission. Here is a summary of our goals:

1. Enable users to get a forecast of job queuing time based on historical data;
2. Present the user with a range of estimated times for each possible cluster accessible via EPIC;
3. Train the model on the data from the EPIC clusters and update it daily;
4. Output the "certainty" associated with the prediction to comply with robustness.

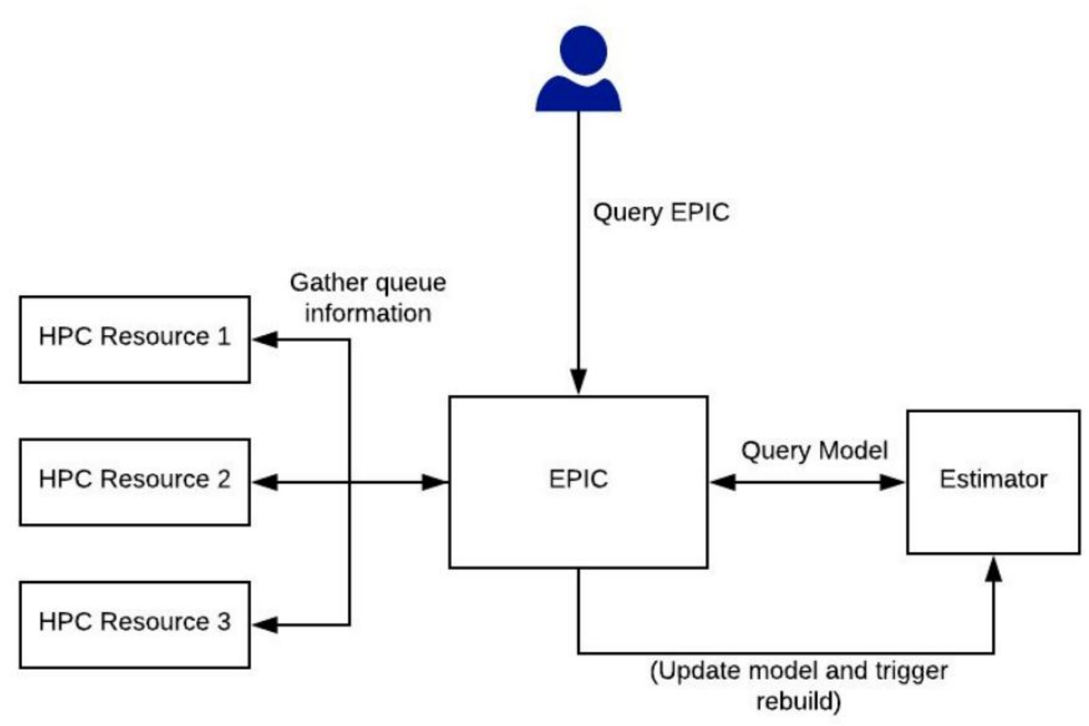
DATASET

In this section we summarise the exploratory analysis conducted on the given dataset.



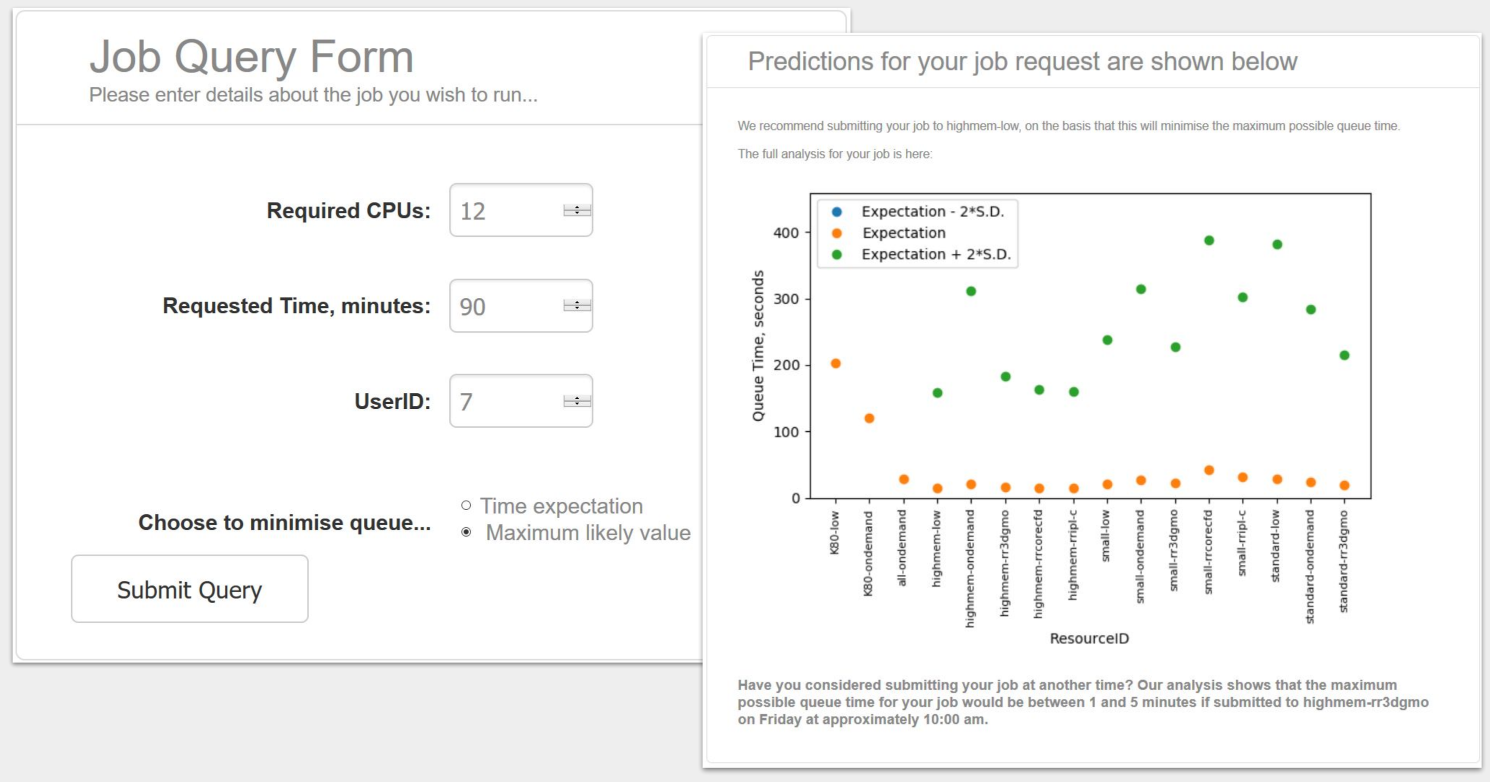
SYSTEM OVERVIEW

The system will give response information about the expected queue times for each cluster and a possible recommendation about the best place to submit the job. A further enhancement to this might be to recommend resources based on more detailed application / simulation knowledge. For example, the system would predict a resource requirement and queue time for each cluster type based on number of iterations and mesh size.



FORECASTER

The model can be queried by the user prior to job submission. The forecaster will give the user insight into the cluster activity and suggest a convenient time to perform the analysis. As shown in the screenshots below.

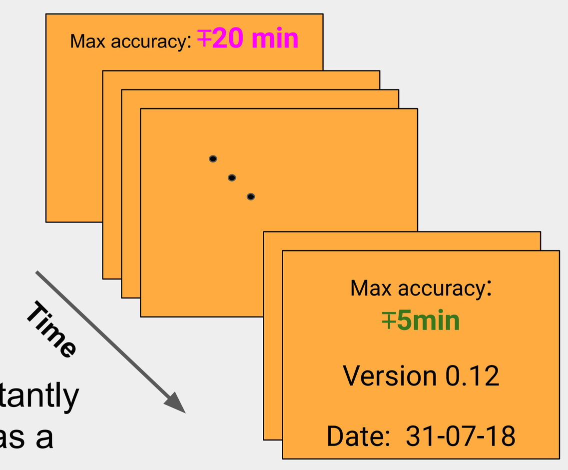


VERSION CONTROL

We are developing a protected online repository of trained models that can be uploaded on the EPIC system and queried in real time.

- Models are re-trained periodically;
- We save the models using *pickle*;
- Archive of trained models is indexed and retained;
- Predictive accuracy is monitored;
- Predictions are made dynamically.

The version control will give the developers the chance to constantly re-train and improve the predictive power of the model,as well as a back-up of the previously trained models.



METHODOLOGY

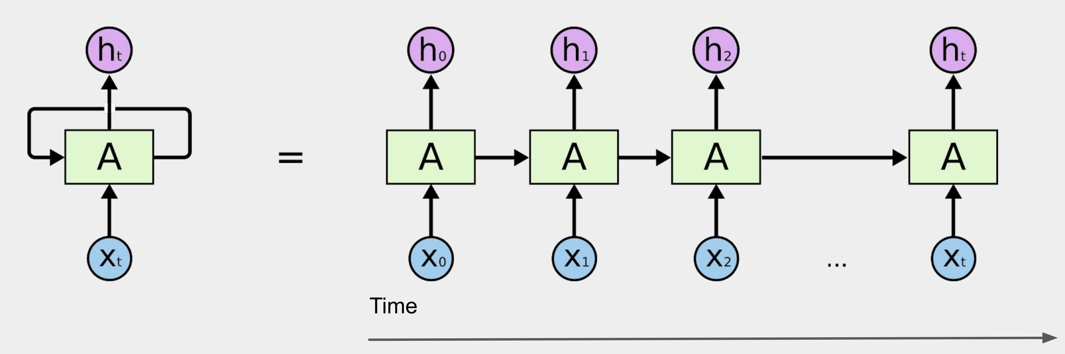
An anonymized database consisting of two months worth of queue data from the EPIC cluster was analyzed. To ease the integration of the learning algorithms we used only Open Source software libraries.

- Python TensorFlow [1] for the neural network;
- GPFlow [2] for the Gaussian process;
- TPOT [3] for the automated model selection;
- OpenCossan [4] for the interval predictors model.

The preliminary study involved the testing of the dataset on the Gaussian process and on our interval predictor, as well as the selection of the regressor using a genetic algorithm. The optimal regressor was obtained solving the Bayesian model selection problem with automatic optimization. This method was implemented using the TPOT Python library on Github.

In the final study the dataset was explored to better understand patterns and users' trends. This study led us to consider a more informed model for the prediction, this time using an artificial neural networks. More specifically, we now consider the time varying nature of the historical data set, and therefore train a recurrent neural network (RNN) to better account of time patterns in the data series [5]. A long short-term memory (LSTM) RNN was developed and trained to make the prediction.

LSTM-RNN architecture



SUMMARY and FUTURE WORK

To recap we:
- Analysed job submission records from HPC clusters;
- Trained recurrent neural network models on the data;
- Periodically retrain and store the model on safe repository;
- Provide users with queue time info' and recommendation for their job.

Future work
- Couple learning algorithm with convergence results from simulation; (i.e. CFD mesh size, number of iterations left, etc.)
- Use a classification approach on request times to speed up prediction;
- See prediction "certainty" as a cost variable, and better plan use of resources;
- Extend the algorithm to reinforcement learning for automatic updating.

Acknowledgements

This work is being grant funded by the UK research and innovation non-departmental public body **Innovate-UK**, which is greatly acknowledged for their support.

REFERENCES

[1] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. 2016.
[2] Matthews, Alexander G. de G., et al. "GPflow: A Gaussian process library using TensorFlow.", *arXiv preprint arXiv:1610.08733* (2016).
[3] Olson, Randal S., and Jason H. Moore. "TPOT: A tree-based pipeline optimization tool for automating machine learning." *Workshop on Automatic Machine Learning*. 2016.
[4] Patelli, Edoardo, et al. "OPENCOSSAN 2.0: an efficient computational toolbox for risk, reliability and resilience analysis." *Proceedings of the joint ICVRAM ISUMA UNCERTAINTIES conference*. 2018.
[5] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).