Helping HPC Users Specify Job Memory Requirements via Machine Learning

Eduardo Rodrigues

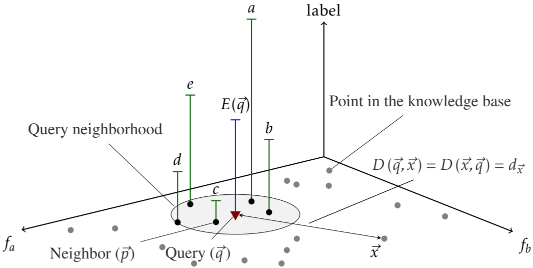
HUST’16 - SuperComputing’16 - Salt Lake City

Motivation

- End-users must specify several parameters in their job submissions to the queue system, e.g.:
  - Number of processors
  - Queue / Partition
  - Memory requirements
  - Other resource requirements
- Those parameters have direct impact in the job turnaround time and, more importantly, in the total system utilization
- Frequently, end-users are not aware of the implications of the parameters they use
- System log keeps valuable information that can be leveraged to improve parameter choice

Related work

- Karnak has been used in XSEDE to predict waiting time and runtime
- Useful for users to plan their experiments
- The method may not apply well for other job parameters, for example memory requirements



Memory requirements

- System owner wants to maximize utilization
- Users may not specify memory precisely
- Log data can provide training examples for a machine learning approach for predicting memory requirements

- This can be seen as a supervised learning task
- We have a set of features (e.g. user id, cwd, command parameters, submission time, etc)
- We want to predict memory requirements (label)

The Wisdom of Crowds

There are many learning algorithms available, e.g. Classification trees, Neural Networks, Instance-based learners, etc

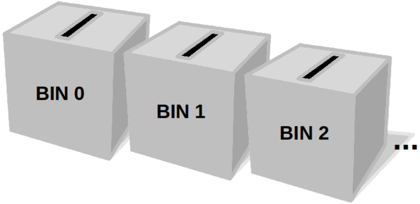
Instead of relying on a single algorithm, we aggregate the predictions of several methods

"Aggregating the judgment of many consistently beats the accuracy of the average member of the group"

The voting strategy

- We turn the task of memory prediction into a classification task
- The predictions fall into regular bins

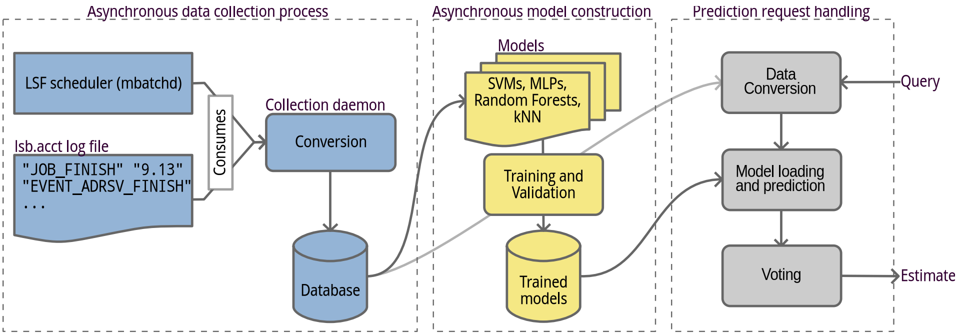
- A set of methods is used
- Each method is trained with log data
- The out-of-sample accuracy is estimated by validation
- During prediction, the accuracy weights the vote of each method



Three main components

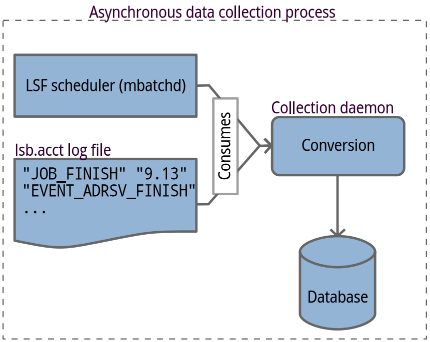
The system has three modules

- Asynchronous data collection process
- Asynchronous model construction
- Prediction request handling



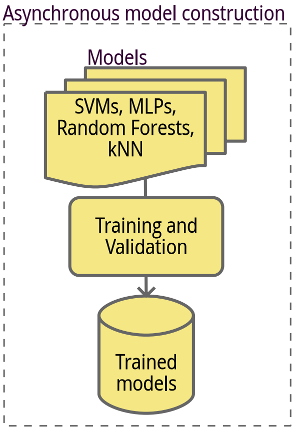
Asynchronous data collection process

- On-line and off-line mode
- Data curation
- Database independent



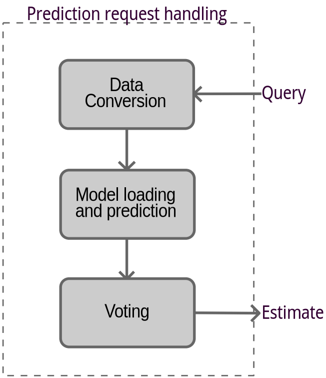
Asynchronous model construction

- Each model is trained and validated in parallel
- Models are stored in a database



Prediction request handling

- Receives as input a bsub command / script
- Returns the predicted memory requirement
- Optionally, submit job with the memory requirement set to the predicted value



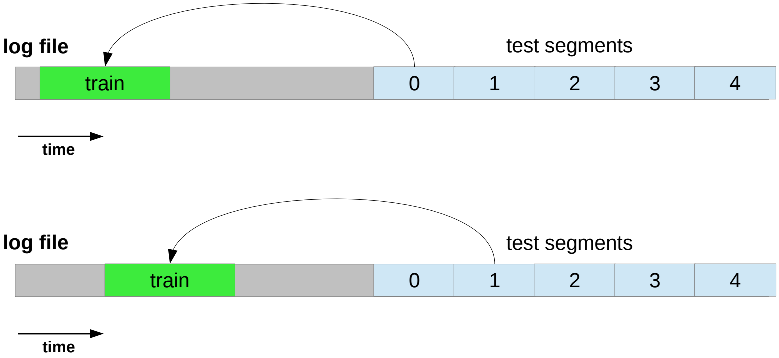

```
[user@masternode ~]$ lspredict -h
usage: lspredict [-h] [-s] bsub [bsub ...]

positional arguments:
  bsub          the bsub command

optional arguments:
  -h, --help    show this help message and exit
  -s, --submit  automatically submit the job with the predicted value
```

Learning methods

Methods used:

- Support Vector Machines (SVM)
  - multi label classification by one versus all approach
  - two models (svm-1, svm-2)
- Random Forests
- Neural Networks (mlp-1, mlp-2)
- K-Nearest Neighbors (KNN)
  - regular voting-based classification (knn-1)
  - same method used in XSEDE for queue time and runtime predictions (knn-2)

Performance evaluation



In production, the tool does not need to wait a specific number of jobs to be retrained

Features used

Feature	Type	Description
User ID	Category	User who submitted the job
Group ID	Category	User group that submitted the job
Queue ID	Category	Number of the queue the job has been submitted to
Working directory	Category	Directory where the job executes
ResReq	Category	Resources requested (e.g. architecture type, GPU)
Command	Category	Command executed
Priority	Number	User priority
Submission time	Number	Time at which the job was submitted
Requested time	Number	Amount of time requested to execute the job
Requested processors	Number	Number of processors requested at the submission time
Weekday	Number	Day of the week in which the job was submitted
Time since midnight	Number	Time of the day at which the job was

Accuracy of the methods

2,128-node x86 system

Test performance								
segment	mode	svm-1	svm-2	rforest	mlp-1	mlp-2	knn-1	knn-2
0	0.7910	**0.8606**	0.3948	0.6546	**0.8610**	**0.7278**	0.6558	**0.6826**
1	0.6024	**0.7448**	0.1202	**0.7544**	**0.7448**	0.7180	**0.7540**	0.5492
2	0.6626	**0.8640**	0.0484	**0.8698**	**0.8630**	**0.8666**	0.6778	0.6770
3	0.8066	**0.8836**	0.2464	**0.8918**	0.7726	**0.8842**	**0.8792**	0.5166
4	0.7742	**0.8038**	0.7568	0.7812	**0.8014**	**0.8140**	0.7772	**0.8034**

Accuracy of the methods

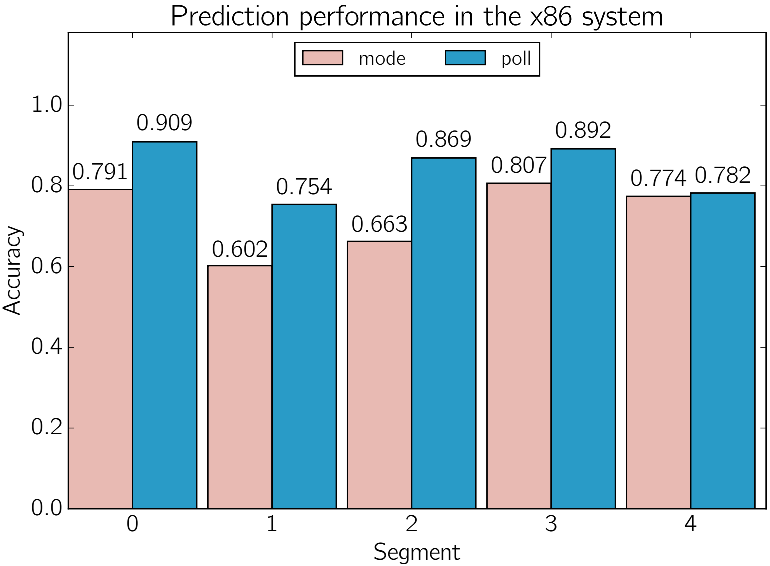
2,128-node x86 system

Test performance								
segment	mode	svm-1	svm-2	rforest	mlp-1	mlp-2	knn-1	knn-2
0	0.7910	**0.8606**	0.3948	0.6546	**0.8610**	**0.7278**	0.6558	**0.6826**
1	0.6024	**0.7448**	0.1202	**0.7544**	**0.7448**	0.7180	**0.7540**	0.5492
2	0.6626	**0.8640**	0.0484	**0.8698**	**0.8630**	**0.8666**	0.6778	0.6770
3	0.8066	**0.8836**	0.2464	**0.8918**	0.7726	**0.8842**	**0.8792**	0.5166
4	0.7742	**0.8038**	0.7568	0.7812	**0.8014**	**0.8140**	0.7772	**0.8034**

KNN-2, which is used to predict waiting time and running time, was not very consistent.

Comparison between mode and poll

x86 system



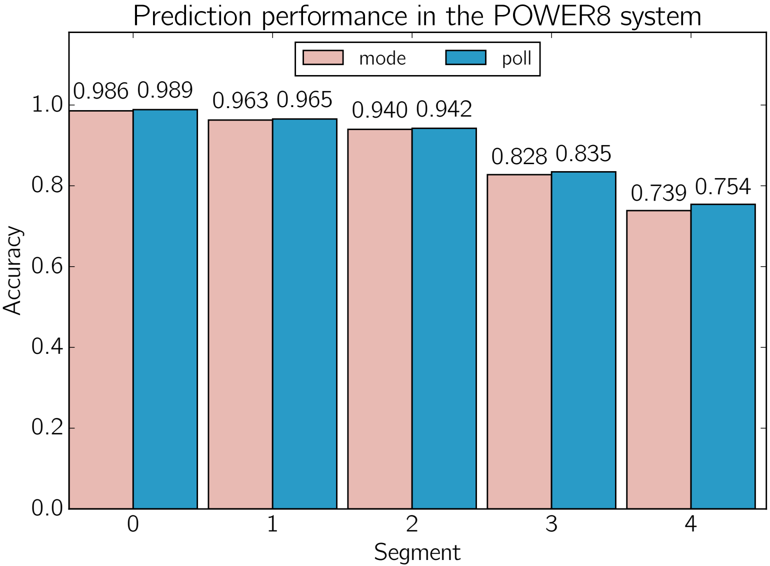
Accuracy of the methods

26-node Power8 system

segment	mode	svm-1	svm-2	rforest	mlp-1	mlp-2	knn-1	knn-2
0	0.9856	**0.9856**	0.0024	**0.9890**	**0.9848**	0.0666	**0.9796**	**0.9796**
1	0.9628	**0.9610**	0.0014	**0.9656**	**0.9610**	0.7772	**0.9620**	**0.9610**
2	0.9398	**0.9436**	0.0042	**0.9428**	**0.9398**	0.1322	**0.2912**	0.2826
3	0.8276	**0.8264**	0.0092	**0.8360**	0.8162	0.7952	**0.8278**	**0.8168**
4	0.7386	**0.7412**	0.0232	**0.7474**	**0.7400**	0.5162	**0.7410**	0.7274

Comparison between mode and poll

Power8 system



Final remarks

- System log data can be leveraged to improve resource requirement specifications
- One machine learning method may not fit all
- In this tool we explored memory prediction, but other resources can be predicted as well

QUESTIONS ?