Helping HPC Users Specify Job Memory Requirements via Machine Learning

Eduardo Rodrigues

HUST’16 - SuperComputing’16 - Salt Lake City
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.
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)
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.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Category</td>
<td>User who submitted the job</td>
</tr>
<tr>
<td>Group ID</td>
<td>Category</td>
<td>User group that submitted the job</td>
</tr>
<tr>
<td>Queue ID</td>
<td>Category</td>
<td>Number of the queue the job has been submitted to</td>
</tr>
<tr>
<td>Working directory</td>
<td>Category</td>
<td>Directory where the job executes</td>
</tr>
<tr>
<td>ResReq</td>
<td>Category</td>
<td>Resources requested (e.g. architecture type, GPU)</td>
</tr>
<tr>
<td>Command</td>
<td>Category</td>
<td>Command executed</td>
</tr>
<tr>
<td>Priority</td>
<td>Number</td>
<td>User priority</td>
</tr>
<tr>
<td>Submission time</td>
<td>Number</td>
<td>Time at which the job was submitted</td>
</tr>
<tr>
<td>Requested time</td>
<td>Number</td>
<td>Amount of time requested to execute the job</td>
</tr>
<tr>
<td>Requested processors</td>
<td>Number</td>
<td>Number of processors requested at the submission time</td>
</tr>
<tr>
<td>Weekday</td>
<td>Number</td>
<td>Day of the week in which the job was submitted</td>
</tr>
<tr>
<td>Time since midnight</td>
<td>Number</td>
<td>Time of the day at which the job was submitted</td>
</tr>
</tbody>
</table>
### Accuracy of the methods

2,128-node x86 system

<table>
<thead>
<tr>
<th>segment</th>
<th>mode</th>
<th>svm-1</th>
<th>svm-2</th>
<th>rforest</th>
<th>mlp-1</th>
<th>mlp-2</th>
<th>knn-1</th>
<th>knn-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.7910</td>
<td><strong>0.8606</strong></td>
<td>0.3948</td>
<td>0.6546</td>
<td><strong>0.8610</strong></td>
<td><strong>0.7278</strong></td>
<td>0.6558</td>
<td><strong>0.6826</strong></td>
</tr>
<tr>
<td>1</td>
<td>0.6024</td>
<td><strong>0.7448</strong></td>
<td>0.1202</td>
<td><strong>0.7544</strong></td>
<td><strong>0.7448</strong></td>
<td>0.7180</td>
<td><strong>0.7540</strong></td>
<td>0.5492</td>
</tr>
<tr>
<td>2</td>
<td>0.6626</td>
<td><strong>0.8640</strong></td>
<td>0.0484</td>
<td><strong>0.8698</strong></td>
<td><strong>0.8630</strong></td>
<td><strong>0.8666</strong></td>
<td>0.6778</td>
<td>0.6770</td>
</tr>
<tr>
<td>3</td>
<td>0.8066</td>
<td><strong>0.8836</strong></td>
<td>0.2464</td>
<td><strong>0.8918</strong></td>
<td>0.7726</td>
<td><strong>0.8842</strong></td>
<td><strong>0.8792</strong></td>
<td>0.5166</td>
</tr>
<tr>
<td>4</td>
<td>0.7742</td>
<td><strong>0.8038</strong></td>
<td>0.7568</td>
<td>0.7812</td>
<td><strong>0.8014</strong></td>
<td><strong>0.8140</strong></td>
<td>0.7772</td>
<td><strong>0.8034</strong></td>
</tr>
</tbody>
</table>

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

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

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

x86 system

Prediction performance in the x86 system

Accuracy

Segment

Mode

Poll
## Accuracy of the methods

26-node Power8 system

<table>
<thead>
<tr>
<th>segment</th>
<th>mode</th>
<th>svm-1</th>
<th>svm-2</th>
<th>rforest</th>
<th>mlp-1</th>
<th>mlp-2</th>
<th>knn-1</th>
<th>knn-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.9856</td>
<td><strong>0.9856</strong></td>
<td>0.0024</td>
<td><strong>0.9890</strong></td>
<td>0.9848</td>
<td>0.0666</td>
<td><strong>0.9796</strong></td>
<td><strong>0.9796</strong></td>
</tr>
<tr>
<td>1</td>
<td>0.9628</td>
<td><strong>0.9610</strong></td>
<td>0.0014</td>
<td><strong>0.9656</strong></td>
<td>0.9610</td>
<td>0.7772</td>
<td><strong>0.9620</strong></td>
<td><strong>0.9610</strong></td>
</tr>
<tr>
<td>2</td>
<td>0.9398</td>
<td><strong>0.9436</strong></td>
<td>0.0042</td>
<td><strong>0.9428</strong></td>
<td>0.9398</td>
<td>0.1322</td>
<td><strong>0.2912</strong></td>
<td>0.2826</td>
</tr>
<tr>
<td>3</td>
<td>0.8276</td>
<td><strong>0.8264</strong></td>
<td>0.0092</td>
<td><strong>0.8360</strong></td>
<td>0.8162</td>
<td>0.7952</td>
<td><strong>0.8278</strong></td>
<td><strong>0.8168</strong></td>
</tr>
<tr>
<td>4</td>
<td>0.7386</td>
<td><strong>0.7412</strong></td>
<td>0.0232</td>
<td><strong>0.7474</strong></td>
<td><strong>0.7400</strong></td>
<td>0.5162</td>
<td><strong>0.7410</strong></td>
<td>0.7274</td>
</tr>
</tbody>
</table>
Comparison between mode and poll

Power8 system

Prediction performance in the POWER8 system

Accuracy
0.986 0.989
0.963 0.965
0.940 0.942
0.828 0.835
0.739 0.754

Segment
0 1 2 3 4

mode poll
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 ?