A history-based estimation for LHCb job requirements

Nathalie Rauschmayr
on behalf of LHCb Computing

13th April 2015
A history-based estimation for LHCb job requirements

Introduction

How long will a job run and how much memory might it need?

- Underestimation: Job is killed. We lose the whole job!
- Overestimation: What to do with the remaining time
Introduction

- Well studied problem in High Performance Computing
- Some recent studies in HEP:
  - CMS, WLCG multicore task force

Multicore jobs: good runtime estimates allow better scheduling
Introduction

- Well studied problem in High Performance Computing
- Some recent studies in HEP:
  - CMS, WLCG multicore task force

Multicore jobs: good runtime estimates allow better scheduling
A lot of job meta data from past jobs: LHCb bookkeeping
A lot of job meta data from past jobs: LHCb bookkeeping

- Hardware specific:
  - CPU model
  - HS06
  - Cache size
  - RAM Size
A lot of job meta data from past jobs: LHCb bookkeeping

- Hardware specific:
  - CPU model
  - HS06
  - Cache size
  - RAM Size

- Input File
  - Size
  - Number of events
  - Run number (LHCb RunDB)
    - Trigger Configuration Key
    - Avg. Multiplicity
    - Avg. Luminosity ...
A lot of job meta data from past jobs: LHCb bookkeeping

- **Hardware specific:**
  - CPU model
  - HS06
  - Cache size
  - RAM Size

- **Input File**
  - Size
  - Number of events
  - Run number (**LHCb RunDB**)
    - Trigger Configuration Key
    - Avg. Multiplicity
    - Avg. Luminosity ...

- **Job**
  - Start/End time
  - Site
  - Worker Node
  - Memory Footprint
  - Runtime
A lot of job meta data from past jobs: LHCb bookkeeping

- **Hardware specific:**
  - CPU model
  - HS06
  - Cache size
  - RAM Size

- **Input File**
  - Size
  - Number of events
  - Run number (LHCb RunDB)
    - Trigger Configuration Key
    - Avg. Multiplicity
    - Avg. Luminosity ...

- **Job**
  - Start/End time
  - Site
  - Worker Node
  - Memory Footprint
  - Runtime
Our Proposal

Automate the prediction procedure based on prior jobs and given job meta data

Supervised Learning

Reduce false estimates
Simplify Production Manager's life
A history-based estimation for LHCb job requirements

Our Proposal

1. Find similar productions
2. RunDB Bookkeeping
3. Workload Analysis
4. Detect most important features
5. Prediction Formula
6. Submit a bunch of jobs
7. Obtain values from finished jobs
8. Update
Normalized CPU time per event:

\[
\frac{\text{CPUTime} \times \text{HEPSPECValue}}{\text{NumberOfEvents}}
\]

Approximate a Gaussian distribution:
Normalized CPU time per event:

\[
\frac{\text{CPU Time} \times \text{HEPSPEC Value}}{\text{Number of Events}}
\]

Approximate a Gaussian distribution:
A history-based estimation for LHCb job requirements

Normalized CPU time per event:
\[ \frac{\text{CPUTime} \times \text{HEPSPECValue}}{\text{NumberOfEvents}} \]

Approximate a Gaussian distribution:

Reprocessing 2011
Normalized CPU time per event:

\[
\frac{\text{CPU Time} \times \text{HEPSPEC Value}}{\text{Number Of Events}}
\]

Approximate a Gaussian distribution:

Reprocessing 2011

Maximum Likelihood 11.71 HS06.s
Comparison of 2011 and 2012 workloads:

Reprocessing 2011

Reprocessing 2012

Maximum Likelihood 11.71 HS06.s

Maximum Likelihood 17.91 HS06.s
Striping: Sort reconstructed events into different output streams

Reprocessing 2011

Reprocessing 2012

Maximum Likelihood 5.26 HS06.s

Maximum Likelihood 5.87 HS06.s
Detect Most Important Features

Certain well known correlations:

- Beam energy versus event size
Detect Most Important Features

Certain well known correlations:

- Beam energy versus event size
- Pileup versus complexity of reconstruction

Example: Reconstruction (2011 versus 2012)
Detect Most Important Features

Avoid overfitting...
Avoid overfitting...

Linear regression:

\[
\text{runtime per event} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \Theta_4 x_4 + \Theta_5 x_5 + \Theta_6 x_6
\]
Detect Most Important Features

Avoid overfitting...

Linear regression:

\[ \text{runtime per event} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \Theta_4 x_4 + \Theta_5 x_5 + \Theta_6 x_6 \]

Normalize

File Size
Avg. Event Size
HEPSPEC
Number Of Events
Avg. Luminosity
Avg. Multiplicity
Detect Most Important Features

Avoid overfitting...

Linear regression:

$$\text{runtime per event} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \Theta_4 x_4 + \Theta_5 x_5 + \Theta_6 x_6$$

Normalize File Size
Average Event Size
HEPSPEC
Number Of Events
Average Luminosity
Average Multiplicity

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Size</td>
<td>0.80</td>
</tr>
<tr>
<td>Avg. Event Size</td>
<td>0.19</td>
</tr>
<tr>
<td>HEPSPEC</td>
<td>-0.97</td>
</tr>
<tr>
<td>Number Of Events</td>
<td>-1.33</td>
</tr>
<tr>
<td>Avg. Luminosity</td>
<td>0.67</td>
</tr>
<tr>
<td>Avg. Multiplicity</td>
<td>-0.08</td>
</tr>
</tbody>
</table>

Evaluate RMSE
2.16 22% better than naive estimator like MLE
Detect Most Important Features

Avoid overfitting...

Linear regression:

\[ \text{runtime per event} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \Theta_4 x_4 + \Theta_5 x_5 + \Theta_6 x_6 \]

Normalize File Size

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Event Size</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>HEPSPEC</td>
<td>-0.97</td>
<td>-0.97</td>
</tr>
<tr>
<td>Number Of Events</td>
<td>-1.33</td>
<td>-1.55</td>
</tr>
<tr>
<td>Avg. Luminosity</td>
<td>0.67</td>
<td>0.59</td>
</tr>
<tr>
<td>Avg. Multiplicity</td>
<td>-0.08</td>
<td></td>
</tr>
</tbody>
</table>

Find best \( \Theta \) Remove small \( \Theta \)

Evaluate RMSE

- 2.16
- 22% better than naive estimator like MLE
**Detect Most Important Features**

Avoid overfitting...

**Linear regression:**

\[
\text{runtime per event} = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \Theta_4 x_4 + \Theta_5 x_5 + \Theta_6 x_6
\]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Best (\Theta)</th>
<th>Remove Small (\Theta)</th>
<th>RMSE</th>
<th>22% better than naive estimator like MLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Size</td>
<td>0.80</td>
<td>1.05</td>
<td>2.16</td>
<td></td>
</tr>
<tr>
<td>Avg. Event Size</td>
<td>0.19</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEPSPEC</td>
<td>-0.97</td>
<td>-0.97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number Of Events</td>
<td>-1.33</td>
<td>-1.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Luminosity</td>
<td>0.67</td>
<td>0.59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Multiplicity</td>
<td>-0.08</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detect Most Important Features

Stripping Jobs:

Normalize  Find best Θ Remove small Θ  Evaluate RMSE

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Size</td>
<td>-0.19</td>
<td>-0.18</td>
<td></td>
<td></td>
<td>0.54</td>
</tr>
<tr>
<td>Avg. Event Size</td>
<td>0.70</td>
<td>0.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEPSPEC</td>
<td>0.19</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number Of Events</td>
<td>0.23</td>
<td>0.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Luminosity</td>
<td>-0.08</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Multiplicity</td>
<td>0.003</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

25% better than
naive estimator
like MLE
Supervised learning: there exist some labelled training data

What if only little amount of training data available?
1. Find similar jobs which have already run

2. Predict requirements for the next k jobs using either (MLE/LR)

3. When k jobs have finished, update prediction formula with the new results obtained

4. Repeat step 2 and 3 until all jobs have finished
1. Find similar jobs which have already run

2. Predict requirements for the next $k$ jobs using either (MLE/LR)

3. When $k$ jobs have finished, update prediction formula with the new results obtained

4. Repeat step 2 and 3 until all jobs have finished
A history-based estimation for LHCb job requirements

1. Find similar jobs which have already run

2. Predict requirements for the next k jobs using either (MLE/LR)

3. When k jobs have finished, update prediction formula with the new results obtained

4. Repeat step 2 and 3 until all jobs have finished

### Error

- **Accumulated Error**

- **Training Data**

- **New Test Data**

| k Jobs + k Jobs + k Jobs | k Jobs |
1. Find similar jobs which have already run

2. Predict requirements for the next $k$ jobs using either (MLE/LR)

3. When $k$ jobs have finished, update prediction formula with the new results obtained

4. Repeat step 2 and 3 until all jobs have finished

$\text{k Jobs} + \text{k Jobs} + \text{k Jobs} + \text{k Jobs}$
Root Mean Squared Error:

$$\sqrt{\frac{\sum_{i=0}^{n}(\text{difference}_i)^2}{n}}$$

where difference is predicted minus real values.
Conclusion

- Historical data can help us to predict future jobs
- Certain meta data are strongly correlated with runtime
  - Improved prediction up to 25%
- Both models (naive estimator and linear regression) can be easily implemented in Production and support the Production Manager in his work
Questions
Trigger configuration changes over different runs
A history-based estimation for LHCb job requirements

Backup Slides

Stripping Jobs: Memory and Runtime per Event

Accumulated Error vs Job Number

Update after 100 jobs

MLE
LR

Accumulated Error

Job Number

0.2 0.4 0.6 0.8 1 1.2

\cdot 10^5

N. Rauschmayr
A history-based estimation for LHCb job requirements

Reconstruction: Memory Footprint

Reprocessing 2011

Reprocessing 2012

Maximum Likelihood 1.6 GB

Maximum Likelihood 1.76 GB

Memory in kB

Frequency

-σ

+σ

-σ + σ
Stripping: Memory Footprint

- Change from POOL to ROOT

Reprocessing 2011
- Maximum Likelihood 2.2 GB

Reprocessing 2012
- Maximum Likelihood 3.4 GB