21st International Conference on Computing in High Energy and Nuclear Physics **CHEP2015** Okinawa Japan: April 13 - 17, 2015

LHCb DIRAC
LHCb GRID SOLUTION

A history-based estimation for LHCb job requirements

Nathalie Rauschmayr
on behalf of LHCb Computing

13th April 2015

LHCb

Introduction

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

Production Manager

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

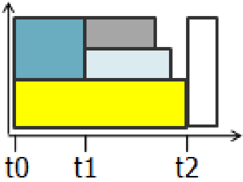
Introduction

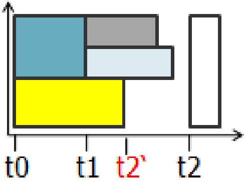
- Well studied problem in High Performance Computing
- Some recent studies in HEP:
  - CMS, WLCG multicore task force

Multicore jobs: good runtime estimates allow better scheduling

- 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

Introduction

A lot of job meta data from past jobs: LHCb bookkeeping

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

Introduction

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 ...

Introduction

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

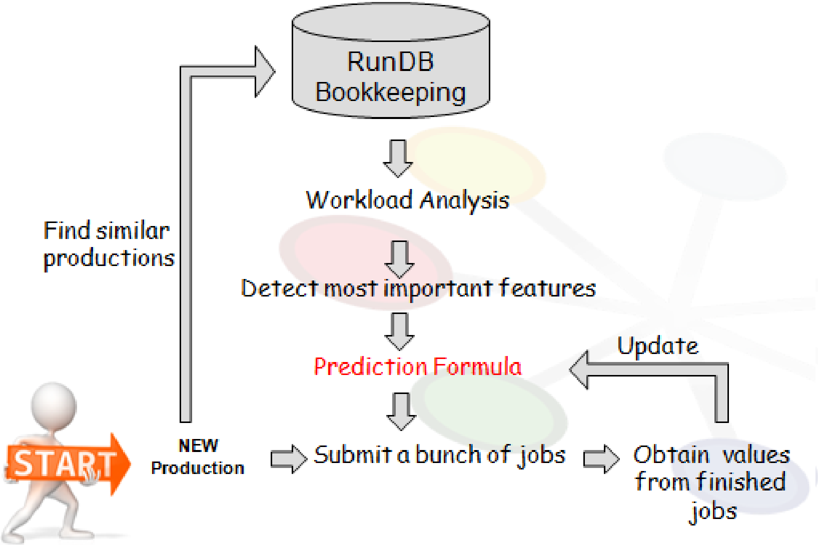
Automate the prediction procedure based on prior jobs and given job meta data

Supervised Learning

Reduce false estimates

Simplify Production Manager's life

Our Proposal



Workload Analysis - Reconstruction

Normalized CPU time per event:

$$\frac{CPUTime * HEPSPECValue}{NumberOfEvents}$$

Approximate a Gaussian distribution:

Workload Analysis - Reconstruction

Normalized CPU time per event:

$$\frac{CPUTime * HEPSPECValue}{NumberOfEvents}$$

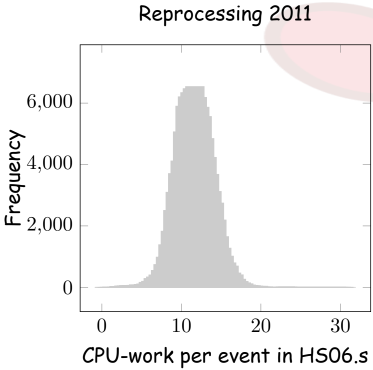
Approximate a Gaussian distribution:

Workload Analysis - Reconstruction

Normalized CPU time per event:

$$\frac{\text{CPUTime} * \text{HEPSPECValue}}{\text{NumberOfEvents}}$$

Approximate a Gaussian distribution:

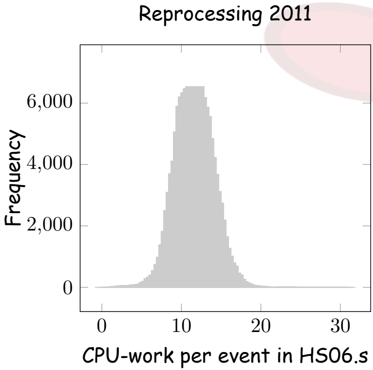


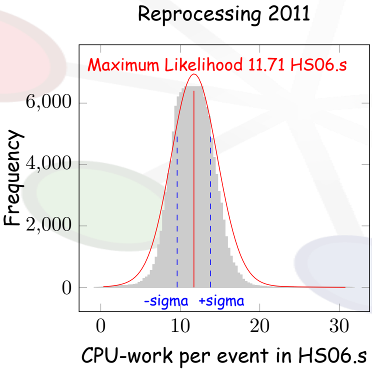
Workload Analysis - Reconstruction

Normalized CPU time per event:

$$\frac{CPUTime * HEPSPECValue}{NumberOfEvents}$$

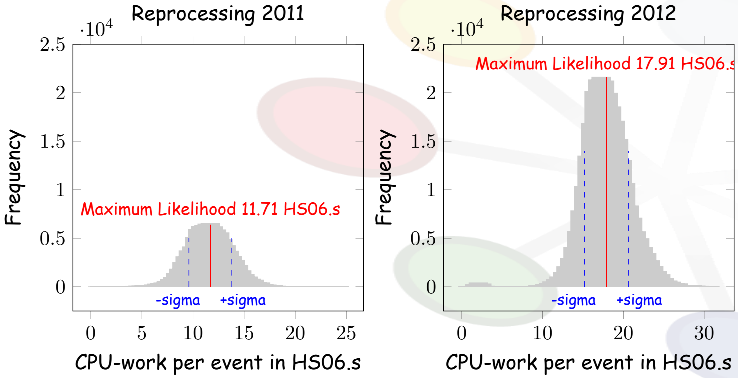
Approximate a Gaussian distribution:





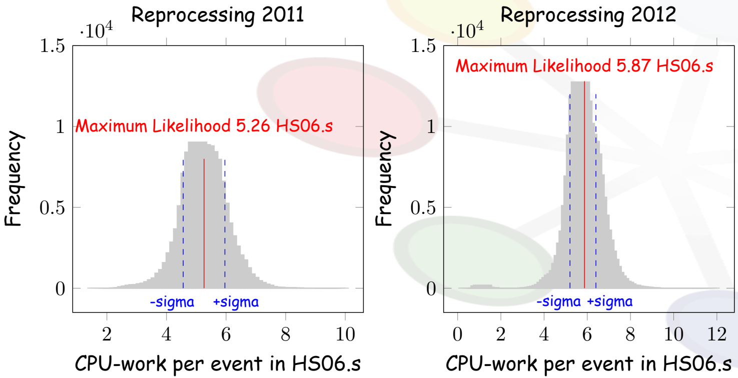
Workload Analysis - Reconstruction

Comparison of 2011 and 2012 workloads:



Workload Analysis - Stripping

Stripping: Sort reconstructed events into different output streams



Detect Most Important Features

Certain well known correlations:

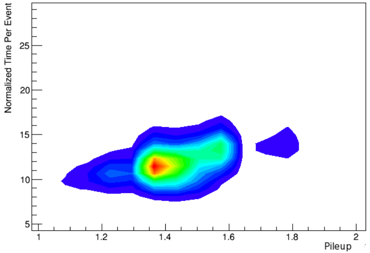
- Beam energy versus event size

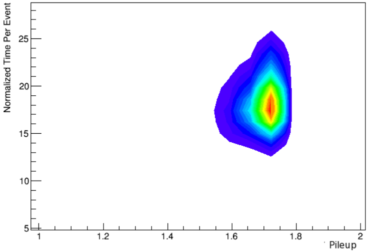
Detect Most Important Features

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...

Detect Most Important Features

Avoid overfitting...

Linear regression:

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:

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:

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	Find best Θ
File Size	0.80
Avg. Event Size	0.19
HEPSPEC	-0.97
Number Of Events	-1.33
Avg. Luminosity	0.67
Avg. Multiplicity	-0.08

Detect Most Important Features

Avoid overfitting...

Linear regression:

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	Find best Θ	Remove small Θ
File Size	0.80	1.05
Avg. Event Size	0.19	x
HEPSPEC	-0.97	-0.97
Number Of Events	-1.33	-1.55
Avg. Luminosity	0.67	0.59
Avg. Multiplicity	-0.08	x

Detect Most Important Features

Avoid overfitting...

Linear regression:

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	Find best Θ	Remove small Θ	Evaluate RMSE
File Size	0.80	1.05	2.16
Avg. Event Size	0.19	x	
HEPSPEC	-0.97	-0.97	
Number Of Events	-1.33	-1.55	
Avg. Luminosity	0.67	0.59	
Avg. Multiplicity	-0.08	x	

22% better than naive estimator like MLE

Detect Most Important Features

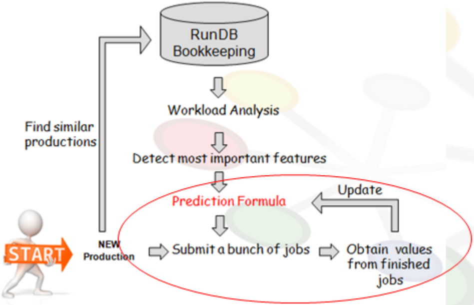
Stripping Jobs:

	Normalize	Find best Θ	Remove small Θ	Evaluate RMSE
File Size		-0.19	-0.18	0.54
Avg. Event Size		0.70	0.62	
HEPSPEC		0.19	0.19	
Number Of Events		0.23	0.27	
Avg. Luminosity		-0.08	x	
Avg. Multiplicity		0.003	x	

25% better than naive estimator like MLE

Supervised Learning

Supervised learning: there exist some labelled training data



What if only little amount of training data available?

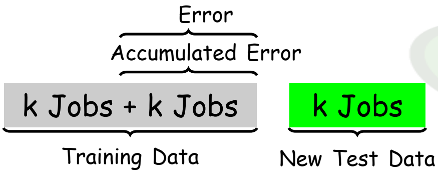
Supervised Learning

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

k Jobs	k Jobs
Training Data	New Test Data

Supervised Learning

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



Supervised Learning

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

k Jobs + k Jobs + k Jobs

Training Data

k Jobs

New Test Data

Supervised Learning

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

k Jobs + k Jobs + k Jobs + k Jobs

k Jobs

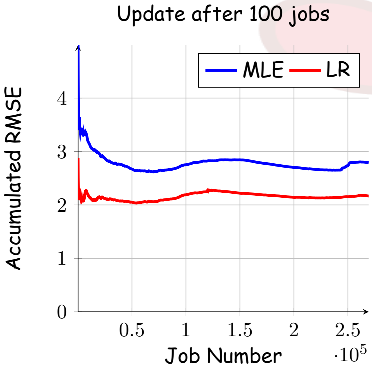
Training Data

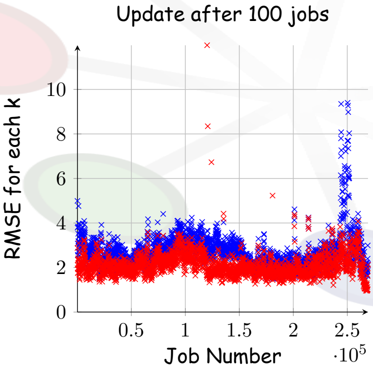
New Test Data

Root Mean Squared Error:

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

where difference is predicted minus real values



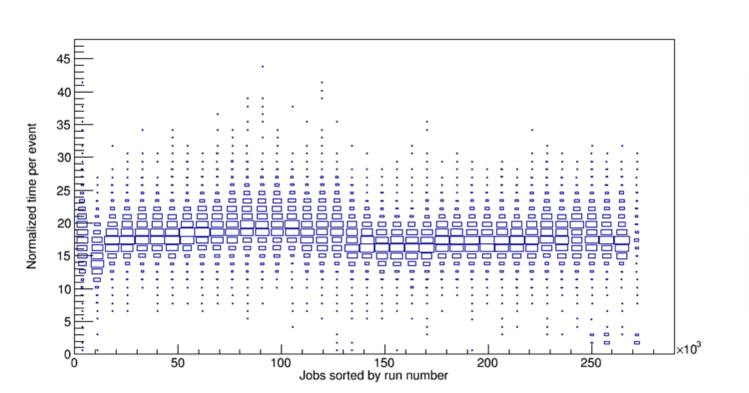


Conclusion

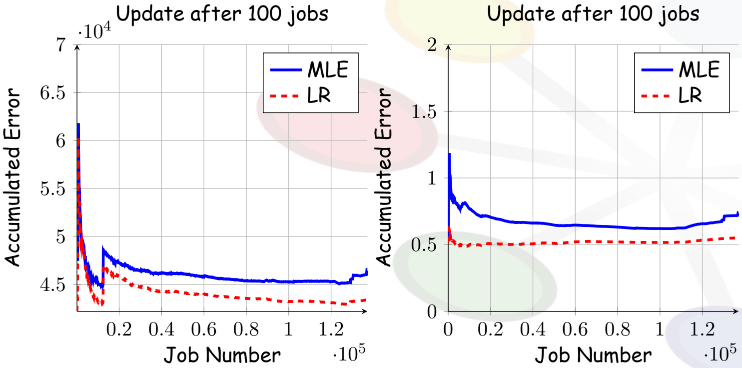
- Historical data can help us to predict future jobs
- Certain meta data are strongly correlated with runtime
  - Improved prediction up to 25%
- Both models (naive estimator and linear regression) can be easily implemented in Production and support the Production Manager in his work

Questions

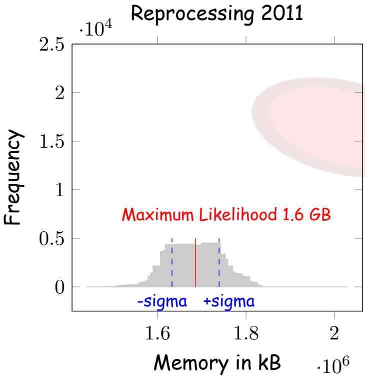
Trigger configuration changes over different runs

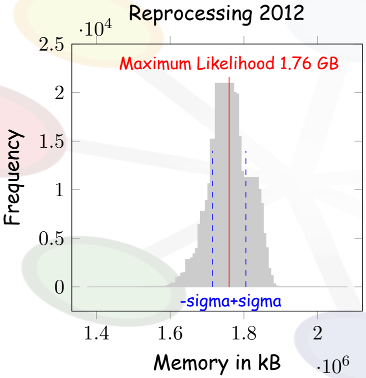


Stripping Jobs: Memory and Runtime per Event



Reconstruction: Memory Footprint





Stripping: Memory Footprint

- Change from POOL to ROOT