Evaluating moldability of LHCb jobs for multicore job submission

Nathalie Rauschmayr

On behalf of LHCb collaboration

September 26, 2013

Outline

Outline

Introduction

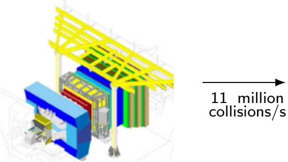
LHCb experiment:

- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics

Introduction

LHCb experiment:

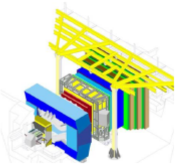
- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics



Introduction

LHCb experiment:

- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics

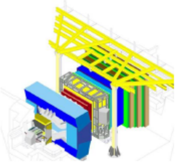


$\xrightarrow[\text{11 million collisions/s}]{}$ HLT $\xrightarrow[\text{500 collisions/s}]{}$

Introduction

LHCb experiment:

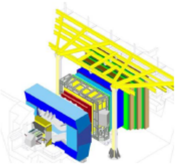
- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics



$\xrightarrow[\text{11 million collisions/s}]{}$ HLT $\xrightarrow[\text{500 collisions/s}]{}$ Reconstruction
$\sim$ 10 billion events per year

LHCb experiment:

- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics



Simulation

$\downarrow$

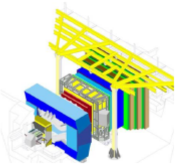
$\xrightarrow[\text{collisions/s}]{\text{11 million}}$ HLT $\xrightarrow{\text{500 collisions/s}}$ Reconstruction

$\sim$ 10 billion events per year

Introduction

LHCb experiment:

- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics



$\xrightarrow[\text{collisions/s}]{\text{11 million}}$ HLT $\xrightarrow{\text{500 collisions/s}}$ Reconstruction ($\sim$ 10 billion events per year)

Simulation $\xrightarrow{\sim \text{ 1 billion events per year}}$ Reconstruction

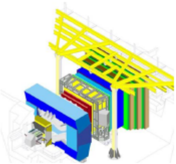
Introduction

LHCb experiment:

- One of the large LHC experiments
- Difference in Matter and Antimatter (CP Violation)
- B Physics

Simulation

$\downarrow$ ~ 1 billion events per year

 $\xrightarrow[\text{11 million collisions/s}]{}$ HLT $\xrightarrow{\text{500 collisions/s}}$ Reconstruction $\rightarrow$ Stripping

~ 10 billion events per year

Introduction

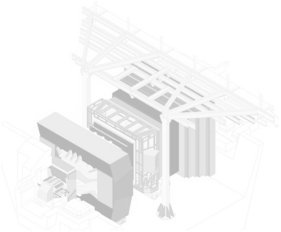
Main Problem: **Memory Footprint**

Two trends:

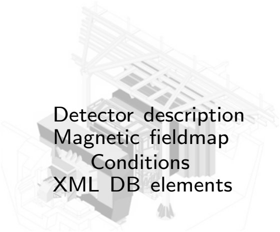
- Memory per core
    - Currently: 0.5 bytes/flop
    - Foreseen: < 0.1 bytes/flop
- LHC parameters
    - Larger events (30 kB in 2009 up to 60 kB in 2012)
    - Complexity of reconstruction

Introduction

Sharing of datasets:



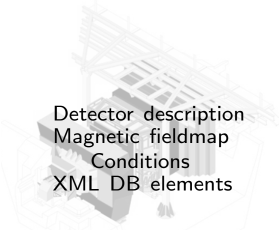
Sharing of datasets:



The more processes the larger the overall memory reduction:
$(n-1) \cdot SharedMemory$

Introduction

Sharing of datasets:



The more processes the larger the overall memory reduction:
$(n-1) \cdot \mathit{SharedMemory}$

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 8 cores available for 1 day
- What to do?
  - Run each job with all cores available $\rightarrow$ reasonable if good scaling

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 8 cores available for 1 day
- What to do?
  - Run each job with all cores available → reasonable if good scaling

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 8 cores available for 1 day
- What to do?
  - Run each job with all cores available → reasonable if good scaling

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 8 cores available for 1 day
- What to do?
  - Run each job with all cores available → reasonable if good scaling

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 8 cores available for 1 day
- What to do?
  - Run each job with all cores available $\rightarrow$ reasonable if good scaling

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 50 cores available for 1 day
- What to do?
  - Run each job with all cores available → reasonable if good scaling
  - Mix jobs in an appropriate way

Scheduling and multicore jobs

- Scheduling becomes more complex
- One more dimension: Number of processes
- Example: Worker node with 50 cores available for 1 day
- What to do?
  - Run each job with all cores available → reasonable if good scaling
  - Mix jobs in an appropriate way

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

→ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

$\rightarrow$ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

$\rightarrow$ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

→ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

$\rightarrow$ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

$\rightarrow$ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Scheduling and multicore jobs

How to use multicore CPUs more efficiently:

- Reduce gaps in schedule
- Limit loss due to non linear speedup

$\rightarrow$ Use moldability of jobs to optimize objective function

Main Problem: Run time prediction

- What does run time rely on?
- Can it be predicted within a given range?
- How large is the introduced error?

Example

Input: Certain number of jobs and cores
Test: All possible combinations

Example

Input: Certain number of jobs and cores
Test: All possible combinations

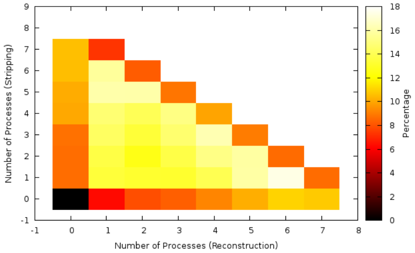


Figure : Used total CPU-time with different mixtures of parallel jobs

Example

Input: Certain number of jobs and cores
Test: All possible combinations

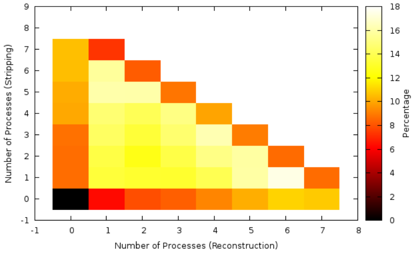


Figure : Used total CPU-time with different mixtures of parallel jobs

Can the optimum be predicted?
How is the result influenced by errors in run time prediction?

Example - How to find an optimum?

- NP hard problem
- Iterative approaches:
  - Assign an additional core to the most worthy job
  - Job which looses less CPU time due to non linear speedup
- Required input:
  - Speedup
  - Run time prediction

Speedup prediction

Downey Speedup Model:

$$S(n) = \begin{cases} \frac{An}{A+\sigma(n-1)/2} & 1 \leq n \leq A \\ \frac{An}{\sigma(A-1/2)+n(1-\sigma/2)} & A \leq n \leq 2A-1 \\ A & n \geq 2A-1 \end{cases}$$

Speedup prediction

Downey Speedup Model:

$$S(n) = \begin{cases} \frac{An}{A+\sigma(n-1)/2} & 1 \leq n \leq A \\ \frac{An}{\sigma(A-1/2)+n(1-\sigma/2)} & A \leq n \leq 2A-1 \\ A & n \geq 2A-1 \end{cases}$$

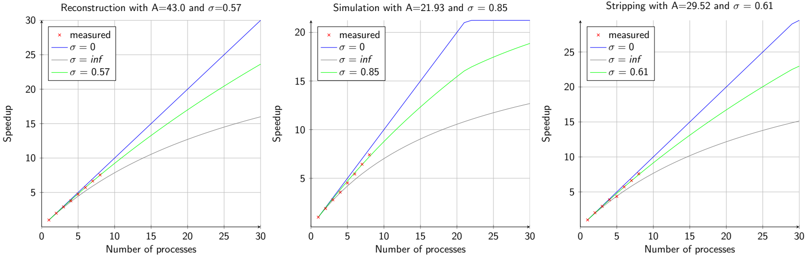


Figure : A = Average parallelism and σ = variance in parallelism

Run time prediction

- Using historical information
- Clustering
  - Grid Site
  - CPU-type
  - Eventtype
  - Production
  - Workernode
- Distribution of datasets: CPU-work per event (CPU-time · HEPSPEC-value)

$$RunTime = nEvt \cdot MaxLikelihood / PowerOfMachine$$

Run time prediction - Stripping

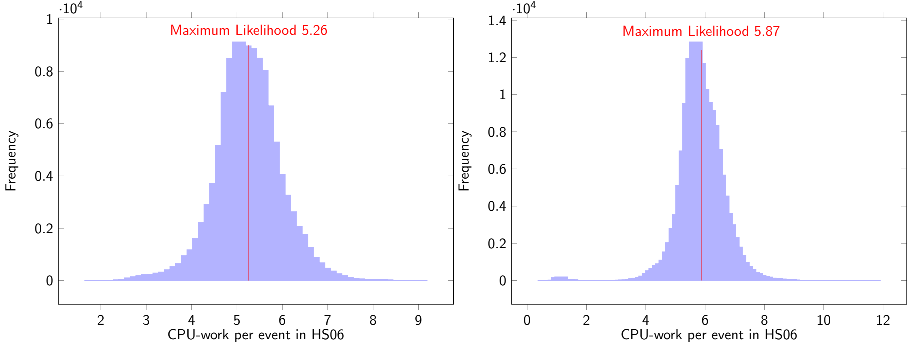
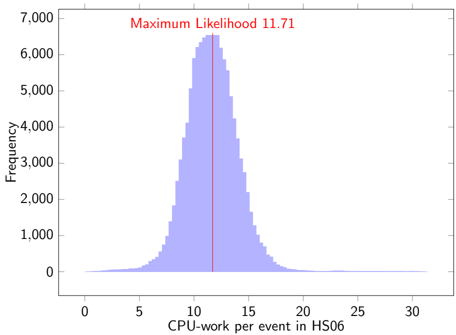


Figure : Stripping jobs of reprocessing 2011 versus 2012

Run time prediction - Reconstruction



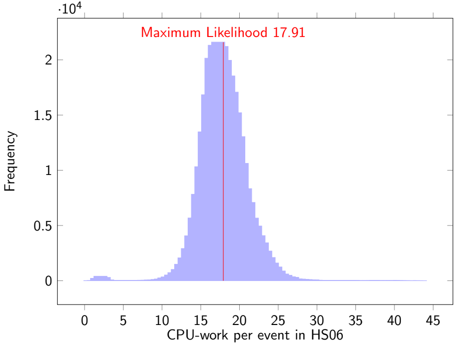


Figure : Reconstruction jobs of reprocessing 2011 versus 2012

Run time prediction - Simulation step

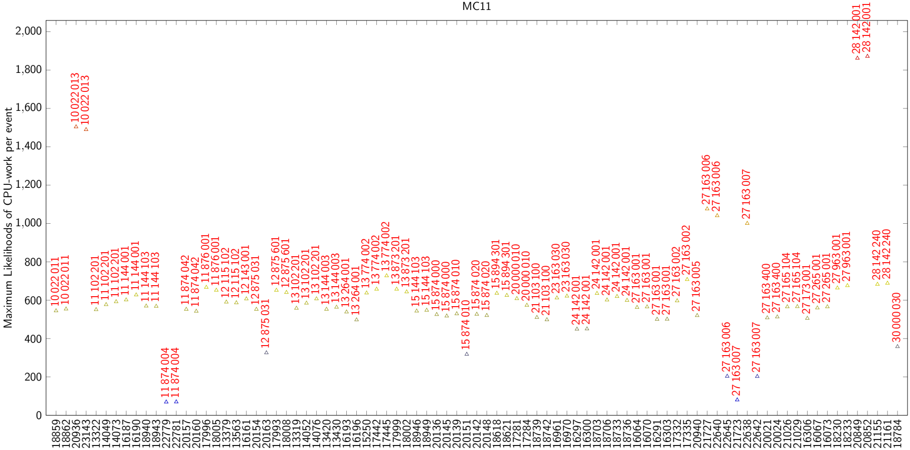


Figure : Maximum likelihoods of MC11 jobs with different event types and from different productions

Back to the example

Run time predicted as:

$$Workload_{min} = MaxLikelihood - x \cdot \sigma$$

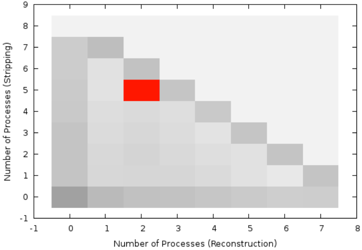
$$Workload_{max} = MaxLikelihood + x \cdot \sigma$$

Back to the example

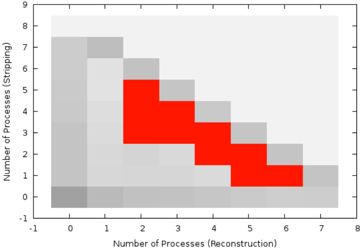
Run time predicted as:

$$Workload_{min} = MaxLikelihood - x \cdot \sigma$$

$$Workload_{max} = MaxLikelihood + x \cdot \sigma$$



(a) Confidence interval of 0σ



(b) Confidence interval of 2σ

Figure : Decision found by an iterative approach

Outcome

- Difficult to predict the optimum
- Theoretical optimum must not be the real one:
  - Jobs can influence each other (concurrent accesses)
  - Uncertainties in the prediction (LHC configuration)
  - Iterative approaches tend get stuck in local optima
- Approximation of global optimum already sufficient

Outcome

- Difficult to predict the optimum
- Theoretical optimum must not be the real one:
  - Jobs can influence each other (concurrent accesses)
  - Uncertainties in the prediction (LHC configuration)
  - Iterative approaches tend get stuck in local optima
- Approximation of global optimum already sufficient

Outcome

- Difficult to predict the optimum
- Theoretical optimum must not be the real one:
  - Jobs can influence each other (concurrent accesses)
  - Uncertainties in the prediction (LHC configuration)
  - Iterative approaches tend get stuck in local optima
- Approximation of global optimum already sufficient

Outcome

- Difficult to predict the optimum
- Theoretical optimum must not be the real one:
  - Jobs can influence each other (concurrent accesses)
  - Uncertainties in the prediction (LHC configuration)
  - Iterative approaches tend get stuck in local optima
- Approximation of global optimum already sufficient

Outcome

- Difficult to predict the optimum
- Theoretical optimum must not be the real one:
  - Jobs can influence each other (concurrent accesses)
  - Uncertainties in the prediction (LHC configuration)
  - Iterative approaches tend get stuck in local optima
- Approximation of global optimum already sufficient

Outcome

- Difficult to predict the optimum
- Theoretical optimum must not be the real one:
  - Jobs can influence each other (concurrent accesses)
  - Uncertainties in the prediction (LHC configuration)
  - Iterative approaches tend get stuck in local optima
- Approximation of global optimum already sufficient