Benchmarking LHCb Applications
Preliminary Results

Hubert Degaudenzi[1]

[1]LHCb Collaboration

HEPiX Spring 2008
CPU technology, 08.05.2008

Outline
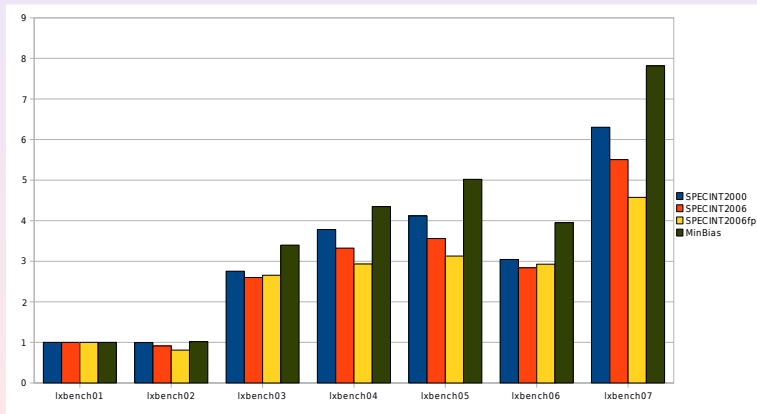
Outline

Outline

Outline

Outline

The Method

- One instance of the executable is started for each CPU core
- The event rate (mark) is computed internally in the executable itself. It allows to prevent the inclusion of the startup and shutdown times of the application. The final mark is the sum of the event rates for all instances.
- An external event rate is also computed for cross-checking purpose.
- If the executable needs an input file, it is copied beforehand on the local disk.
- The number of event processed should cover a reasonable amount of the parameters phase space without taking too much time. The number of 500 events has been chosen as a trade-off.
- The error estimator on the mark is deduced from the spread of the internal time of the instances ($\sim 2\%$).

Outline

1 The Method

2 The Results
- Gauss – Generation and Simulation
- Brunel – Reconstruction

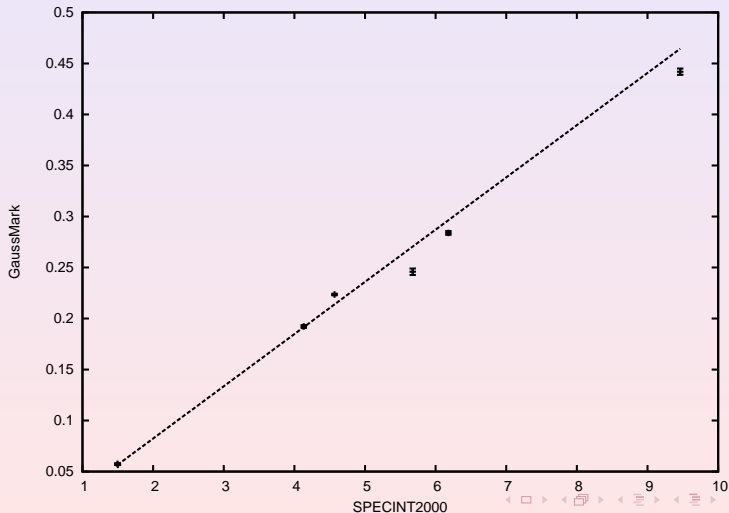## 3 Summary

4 Further Improvements

Gauss – Generation and Simulation

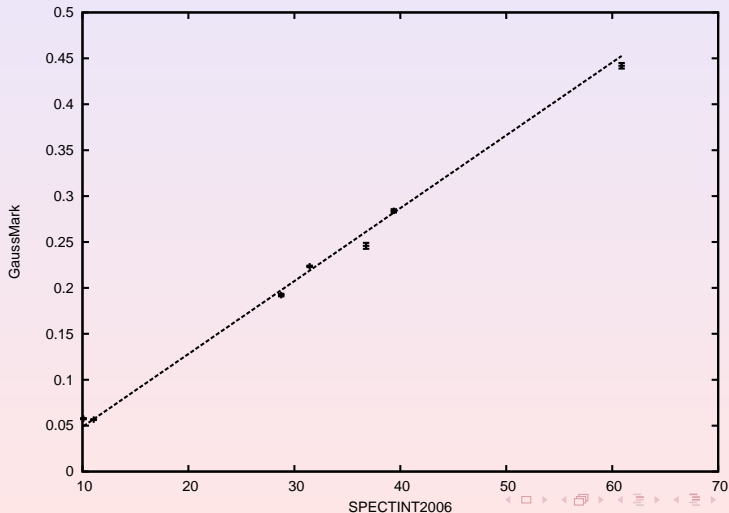- About 4 hours for each run.
- Minimum bias p-p events produced
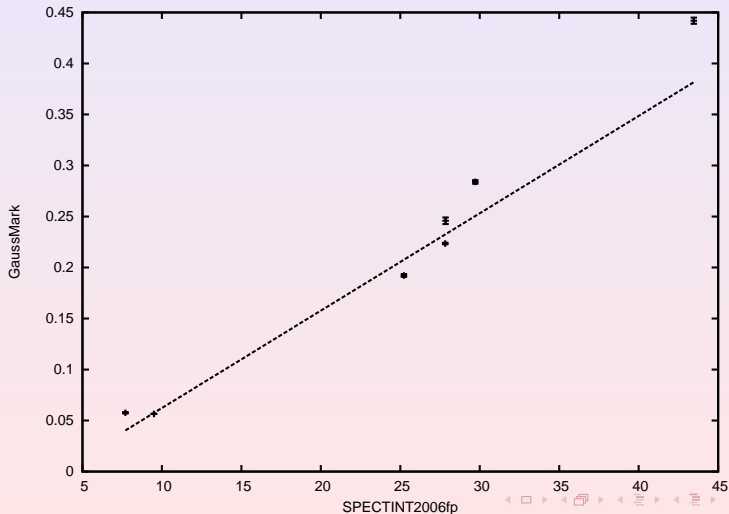
Gauss – SPECINT2000

$\chi^2/\text{ndf} = 62.52$

Gauss – SPECINT2006
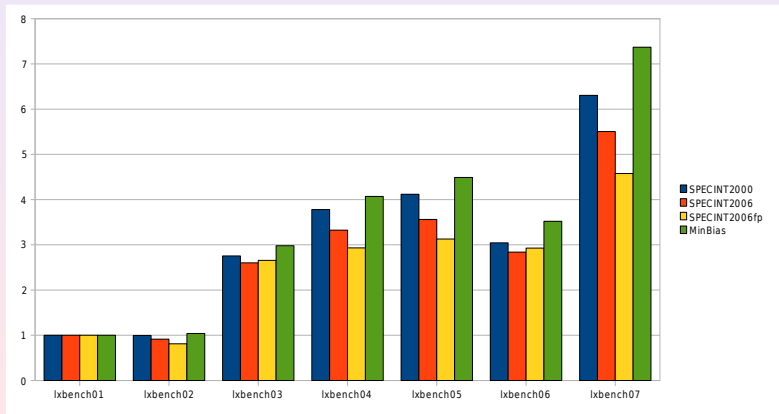
$\chi^2/\text{ndf} = 47.96$

Gauss – SPECINT2006fp

$\chi^2/\text{ndf} = 334.16$

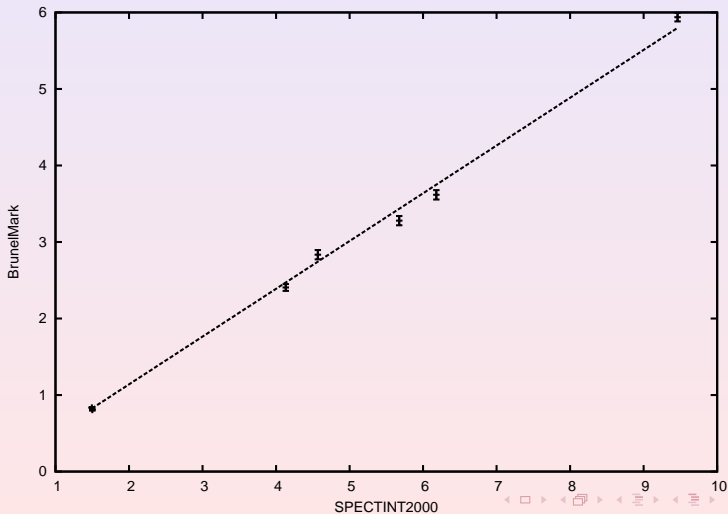# Brunel – Reconstruction

- about 20 minutes for each run.
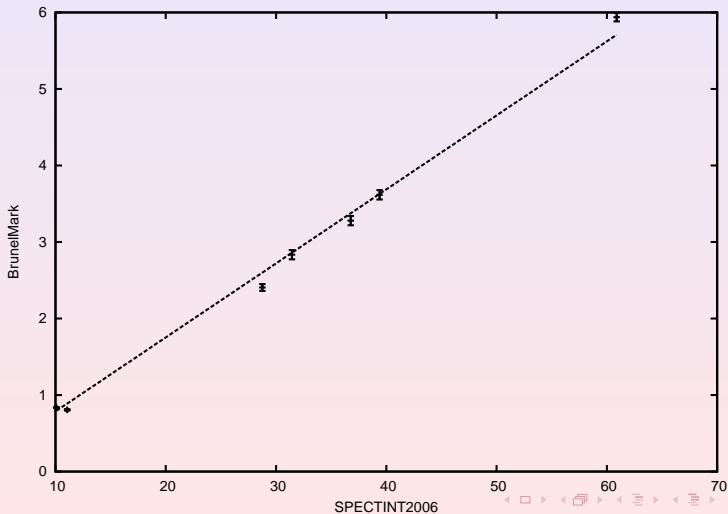- minimum bias digitized events as input.
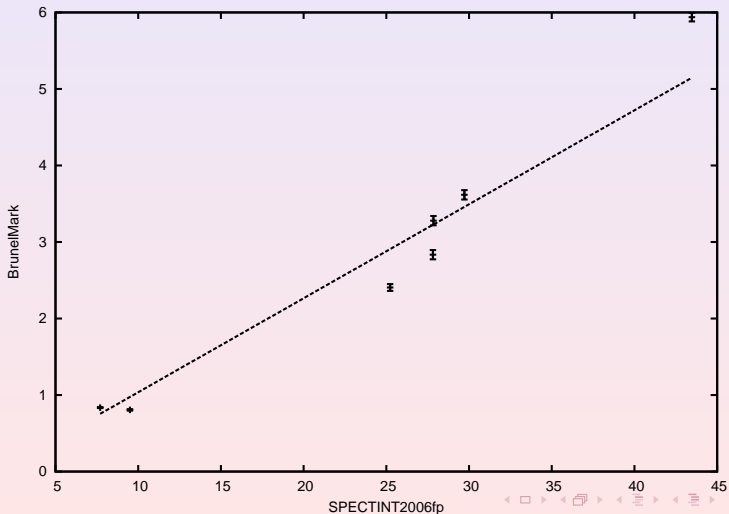
Brunel – SPECINT2000

$\chi^2/\text{ndf} = 6.20$

Brunel – SPECINT2006

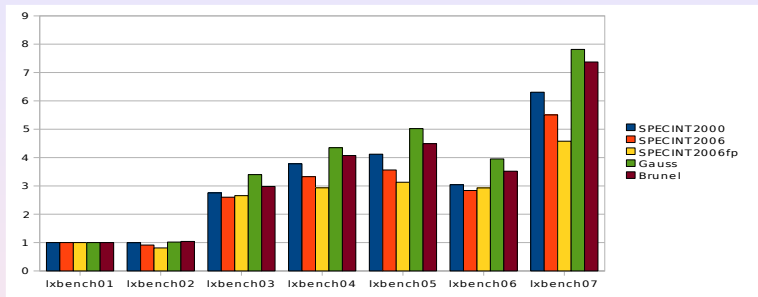$\chi^2/\mathsf{ndf} = 32.45$

Brunel – SPECINT2006fp

$\chi^2/\text{ndf} = 176.26$

Outline

1. The Method

2. The Results
   - Gauss – Generation and Simulation
   - Brunel – Reconstruction

3. Summary

4. Further Improvements

Summary

Proc.	Type	χ^2/ndf
Gauss	SPECINT2000	62.52
Gauss	SPECINT2006	47.96
Gauss	SPECINT2006fp	334.16
Brunel	SPECINT2000	6.20
Brunel	SPECINT2006	32.45
Brunel	SPECINT2006fp	176.26

Outline

Further Improvements

- Run the benchmark on the other applications:
    - Boole (digitalization)
    - DaVinci (stripping)
- Run the benchmark on a different data set.
- Find a better estimation for the error on the Mark.
- Comparison between i686 and x86_64 software builds.