3 New Performance Metrics Sets in LHCbPR2

Christoph Hasse, Stefan Roiser, Maciej Szymanski

9th Computing Workshop

18 May ‘17
Introduction

• For algorithms executed in the future Gaudi framework:

• Generate performance metrics from a nightly build and push them into LHCbPR2
• Done for 3 different sets of metrics
  • TimeLineSvc – execution time per event per algorithm
  • Callgrind– “CPU cost metrics”
  • PrChecker – “physics performance”
• E.g. via the LHCbPR2 trend service evaluate the performance evolution of a certain metric over time
 Metric Set 1: TimeLineSvc

- Developed by Christoph Hasse
- Extracting metrics on timing per algorithm, per event and total execution time (last algo on last event – first algo on first event)

- Options: Rec/Brunel/options/MiniBrunel.py
  - i.e. all algorithms converted so far, e.g. including RICH
  - Run on 100 events, 2 threads, 2 events in flight

- NB: Currently running on VMs, in the future on bare metal machines without interference from other sources
Metrics

• Metrics per <algorithm>:
  • min_<algorithm>
  • max_<algorithm>
  • mean_<algorithm>
  • sigma_<algorithm>

• Metrics per event
  • min_evt_time
  • max_evt_time
  • mean_evt_time
  • sigma_evt_time

• Total execution
  • total_execution_time

• <algorithm> = all algorithms converted to Gaudi functional
Metric Set 2: Callgrind

• Extract standard “cost” metrics from callgrind + some more additional calculated ones
  • E.g. branch miss predictions, simd instructions, l1 cache misses, ...

• Options: Rec/Brunel/options/MiniBrunelHLT1.py
  • i.e. only the “upgrade HLT1 sequence”
Metrics

- `<metric_name>_<algorithm>`

- E.g. “ifp32x4_PatPV3D” – SIMD 32x4 fp instructions of PatPV3D”
Standard CacheGrind Metrics:

- **ir** – instruction fetch
- **dr** – data read access
- **dw** – data write access
- **l1mr** – L1 instruction fetch miss
- **d1mr** – L1 data read miss
- **d1mw** – L1 data write miss
- **ilmr** – LL instruction fetch miss
- **dlmr** – LL data read miss
- **dlmw** – LL data write miss
- **bc** – conditional branch
- **bcm** – mispredicted conditional branch
- **bi** – indirect branch
- **bim** – mispredicted indirect branch
- **ifp32x1** – scalar f32 instruction
- **ifp64x1** – scalar f64 instruction
- **ifp32x2** – simdx2 f32 instruction
- **ifp64x2** – simdx2 f64 instruction
- **ifp32x4** – simdx4 f32 instruction
- **ifp64x4** – simd64 f64 instruction
- **ifp64x8** – simdx6 f32 instruction

**l1m** – L1 miss sum (ilmr + d1mr + dlmw), **llm** – last level miss sum (ilmr + dlmr + dlmw), **bm** – mispredicted branches (bim + bcm), **cest** – cycle estimation (ir + 10*bm + 10*l1m + 100*llm), **fp32** – f32 ops (ifp32x1 + 2*ifp32x2 + 4*ifp32x4 + 8*ifp32x8), **fp64** – f64 ops (ifp64x1 + 2*ifp64x2 + 4*ifp64x4), **vfp128** – simd 128 fp ops (4*ifp32x4 + 2*ifp64x4), **vfp256** – simd 256 fp ops (8*ifp32x8 + 4*ifp64x4), **vfp** – simd fp ops (2*ifp32x2 + vfp128 + vfp256), **sfp** – scalar fp ops (ifp32x1 + ifp64x1), **flop** – fp ops (fp32 + fp64), **btot** – total branches (bc + bi), **itot** – total instructions (ir + dr + dw)

Additional Metrics:

- **rsimd** – ratio simd instructions (vfp / flop)
- **rbm** – ratio branch misspredictions (bm / btot)
- **rcm** – ratio cache misses ((l1m + llm) / itot)
<algorithm>

- PrPixelTracking
- PrForwardTracking
- TrackEventFitter
- RawBankToSTClusterAlg
- PatPV3D
- RawBankToSTLiteClusterAlg
- PrStoreUTHit
- PRStoreFTHit
- FTRawBankDecoder
- PrVeloUT
Metric Set 3: MiniBrunel_PrChecker

- Check Physics Performance
- Report on Tracking Performance from Brunel log
- Metric names see e.g.

```plaintext
PrChecker2Fast... INFO **** Forward  15342 tracks including 907 ghosts [ 5.91 %], Event average 3.63 % ****
PrChecker2Fast... INFO **** for P>3GeV,Pt>0.5GeV  13953 tracks including 727 ghosts [ 5.21 %] ****
PrChecker2Fast... INFO  01_long 13669 from 33297 [ 41.05 %] 260 clones [ 1.87 %], purity: 99.42 %, hitEff: 95.30 %
PrChecker2Fast... INFO  02_long>5GeV 11801 from 21216 [ 55.62 %] 180 clones [ 1.56 %], purity: 99.48 %, hitEff: 95.63 %
PrChecker2Fast... INFO  03_long_strange>5GeV 468 from 1749 [ 26.76 %] 6 clones [ 0.27 %], purity: 99.13 %, hitEff: 95.45 %
PrChecker2Fast... INFO  04_long_strange>5GeV 370 from 830 [ 44.58 %] 1 clones [ 0.27 %], purity: 99.89 %, hitEff: 94.90 %
PrChecker2Fast... INFO  05_long_fromB>5GeV 42 from 80 [ 52.50 %] 1 clones [ 2.33 %], purity: 99.51 %, hitEff: 95.86 %
PrChecker2Fast... INFO  06_long_fromB>5GeV 36 from 53 [ 67.92 %] 1 clones [ 2.50 %], purity: 99.51 %, hitEff: 95.86 %
PrChecker2Fast... INFO  07_long_fromB>P>3GeV_Pt>0.5GeV 39 from 45 [ 86.67 %] 1 clones [ 2.50 %], purity: 99.51 %, hitEff: 95.86 %
```

```plaintext
PrChecker2Fast... INFO **** TT Efficiency for Rec/Track/ForwardFast **** 959 ghost, 3.94 TT per track
PrChecker2Fast... INFO **** for P>3GeV,Pt>0.5GeV 14065 tracks including 738 ghosts [ 5.2 %] ****
PrChecker2Fast... INFO  01_long 13929 tr 3.99 from 4.06 mcTT [ 98.3 %] 0.04 ghost hits on real tracks [ 1.0 %]
PrChecker2Fast... INFO  02_long>3TT 13914 tr 4.00 from 4.06 mcTT [ 98.3 %] 0.04 ghost hits on real tracks [ 1.0 %]
PrChecker2Fast... INFO  02_long>5GeV>3TT 11981 tr 4.00 from 4.06 mcTT [ 98.5 %] 0.04 ghost hits on real tracks [ 1.1 %]
PrChecker2Fast... INFO  03_long>3TT_P>3GeV_Pt>0.5GeV>3TT 11967 tr 4.00 from 4.06 mcTT [ 98.5 %] 0.04 ghost hits on real tracks [ 1.0 %]
PrChecker2Fast... INFO  03_long>3TT_P>3GeV_Pt>0.5GeV>3TT 0 tr 4.08 from 4.08 mcTT [100.0 %] 0.00 ghost hits on real tracks [ 0.0 %]
PrChecker2Fast... INFO  04_TT>3TT_P>3GeV_Pt>0.5GeV>3TT 40 tr 4.08 from 4.08 mcTT [100.0 %] 0.00 ghost hits on real tracks [ 0.0 %]
PrChecker2Fast... INFO  04_TT>3TT_P>3GeV_Pt>0.5GeV>3TT 40 tr 4.08 from 4.08 mcTT [100.0 %] 0.00 ghost hits on real tracks [ 0.0 %]
```

```plaintext
PrChecker2.Velo INFO **** Velo 111597 tracks including 2181 ghosts [ 1.88 %], Event average 1.39 % ****
PrChecker2.Velo INFO 01_velo 55127 from 55991 [ 98.62 %] 2086 clones [ 3.65 %], purity: 99.80 %, hitEff: 93.86 %
PrChecker2.Velo INFO 02_velo 33152 from 33297 [ 99.56 %] 756 clones [ 2.23 %], purity: 99.84 %, hitEff: 96.34 %
PrChecker2.Velo INFO 03_velo>5GeV 21146 from 21216 [ 99.67 %] 488 clones [ 1.89 %], purity: 99.84 %, hitEff: 96.92 %
PrChecker2.Velo INFO 04_velo>5GeV 1716 from 1749 [ 98.11 %] 24 clones [ 1.38 %], purity: 99.29 %, hitEff: 97.48 %
PrChecker2.Velo INFO 05_velo>5GeV 810 from 830 [ 97.59 %] 9 clones [ 1.16 %], purity: 99.89 %, hitEff: 97.89 %
PrChecker2.Velo INFO 06_velo>5GeV 80 from 80 [100.00 %] 3 clones [ 3.61 %], purity: 99.56 %, hitEff: 94.77 %
PrChecker2.Velo INFO 07_velo>5GeV 53 from 53 [100.00 %] 1 clones [ 1.85 %], purity: 99.79 %, hitEff: 96.18 %
```

```plaintext
PrChecker2.Forward INFO **** Forward 44121 tracks including 12389 ghosts [28.08 %], Event average 19.49 % ****
PrChecker2.Forward INFO 01_long 28786 from 33297 [ 86.45 %] 596 clones [ 2.03 %], purity: 99.23 %, hitEff: 95.11 %
PrChecker2.Forward INFO 02_long>5GeV 19586 from 21216 [ 92.32 %] 273 clones [ 1.37 %], purity: 99.48 %, hitEff: 95.36 %
PrChecker2.Forward INFO 02_long>5GeV 1450 from 1749 [ 82.90 %] 20 clones [ 1.36 %], purity: 98.88 %, hitEff: 95.09 %
PrChecker2.Forward INFO 04_long>5GeV>3TT 711 from 830 [ 85.69 %] 3 clones [ 3.89 %], purity: 99.15 %, hitEff: 96.07 %
PrChecker2.Forward INFO 05_long_fromB>5GeV 68 from 80 [ 85.00 %] 3 clones [ 4.23 %], purity: 98.69 %, hitEff: 95.11 %
```
How to show trends

https://lblhcbpr2.cern.ch/

1. Select “Trends”

2. Application: “Brunel”

3. Options: “MiniBrunel”, “MiniBrunel-Callgrind” or “MiniBrunel_PrChecker”

4. Executables: “lb-run-gaudirun” or “lb-run-gaudirun-callgrind”

5. Tick “Show Nightly versions”, put N into “Number of nightly versions to show” and select N versions under future”

6. Start typing your favorite metric string, eg. “mean...” → select metric
General considerations

• Important to run on “frozen” options
• Especially for TimeLineSvc metrics important to run in a “clean” environment without interferences
  • 2 bare metal machines for those tests are ordered and shall be delivered soon