Pythia8.244 code optimization

Dmitri Konstantinov, Grigory Latyshev
NRC «Kurchatov Institute» – IHEP, Protvino

02.03.2020

Introduction

- Profile typical LHC use-cases of Pythia8 with valgrind/gprof/prof tools and optimize if possible.
- Pythia8 244, LHAPDF-6.2.3
- GCC 9.2.0, -std=c++17 -fPIC -O2 -g
- configurations for benchmarking:
  - 13 TeV pp, hardQCD:all=on+LHAGrid1 [hardQCD+LHAGrid1]:
    - ★ LHAGrid1 is Pythia8 class to work with LHAPDF sets without LHAPDF6 - very general use case with some assumptions.
  - 13 TeV pp, hardQCD:all=on+LHAPDF6 [hardQCD+LHAPDF6]
  - 13 TeV pp, hardQCD:all=on+NNPDF class [hardQCD+NNPDF]
    - ★ NNPDF modified version of the code provided by NNPDF authors, used to read NNPDFs before LHAPDF arrived.

Systematic error of CPU time measurements is roughly 0.5-1%

Pythia8 code for benchmarking

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main(int argc, char* argv[]) {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  // Read in commands from external file.
  pythia.readFile(argv[1]);
  int nEvent = pythia.mode("Main:numberOfEvents");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < nEvent; ++iEvent) {
    if (!pythia.next()) continue;
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0;
    for (int i = 0; i < pythia.event.size(); ++i)
      if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
}
```

Three configurations for benchmarking

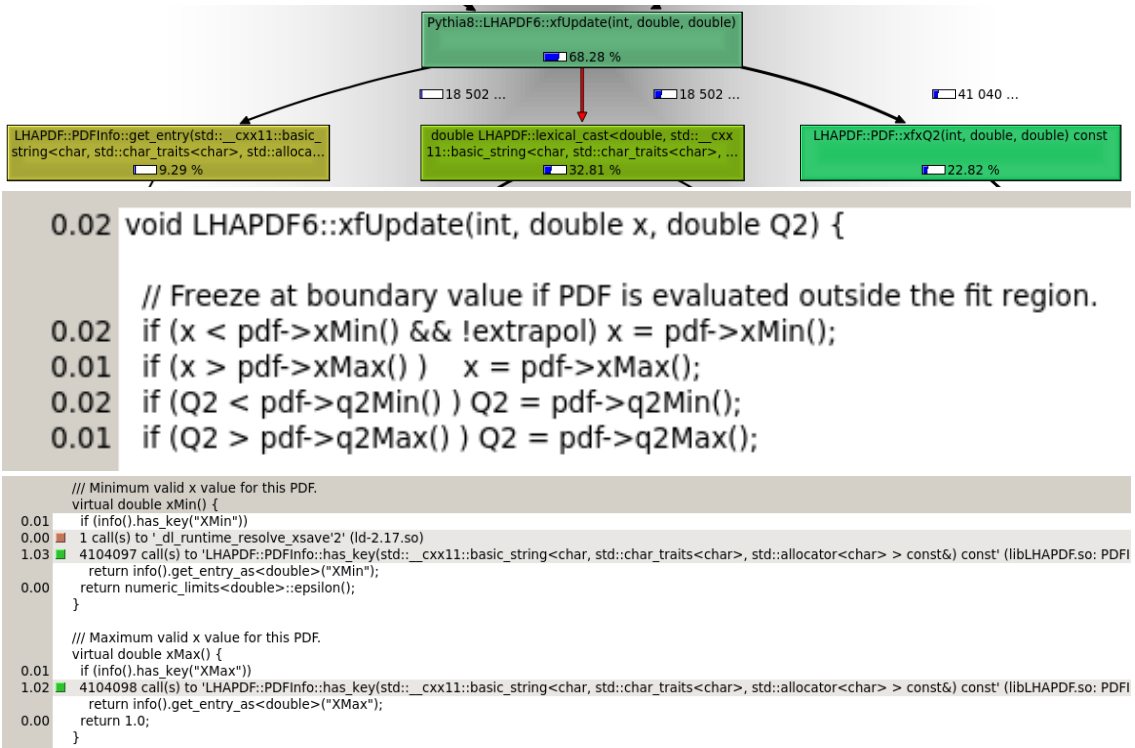
```
! 1) Settings used in the main program.
Main:numberOfEvents = 10000         ! number of events to generate

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 2212                    ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212                    ! second beam, p = 2212, pbar = -2212
Beams:eCM = 13000.                  ! CM energy of collision
PDF:pSet = 13                                          ! [hardQCD+NNPDF]
!PDF:pSet = LHAPDF6:NNPDF23_lo_as_0130_qed             ! [hardQCD+LHAGrid1]
!PDF:pSet = LHAGrid1:NNPDF23_lo_as_0130_qed_0000.dat   ! [hardQCD+LHAPDF]
! 4) Settings for the hard-process generation.

! Example 1: QCD + prompt photon production; must set pTmin.
HardQCD:all = on                    ! switch on all QCD jet + jet processes
PhaseSpace:pTHatMin = 20.    ! pTHatMin
```

[f_lhapdf] first glance at [HardQCD+LHAPDF6]

Incl.	Self	Called	Function
100.00	0.00	(0)	0x0000000000001120
100.00	0.00	5	_dl_runtime_resolve_xsave
100.00	0.00	1	0x000000000043778e
100.00	0.00	(0)	(below main)
100.00	0.01	1	main
95.71	0.00	1 000	Pythia8::Pythia::next()
93.15	0.01	1 000	Pythia8::PartonLevel::next(Pythia8::Event&, Pythia8::Event&)
80.24	8.33	26 393 504	Pythia8::BeamParticle::xfModified(int, int, double, double)
68.28	0.41	4 104 098	Pythia8::LHAPDF6::xfUpdate(int, double, double)
65.33	0.05	26 393 504	Pythia8::LHAPDF::xfSea(int, double, double)
65.28	0.58	26 393 504	Pythia8::PDF::xfSea(int, double, double)
62.55	0.13	117 559	Pythia8::SimpleSpaceShower::pTnext(Pythia8::Event&, double, double, int, bool)
62.19	0.47	2 226 292	Pythia8::SimpleSpaceShower::pT2nextQCD(double, double)
32.81	1.06	18 502 366	double LHAPDF::lexical_cast<double, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<c...
22.82	0.78	41 040 979	LHAPDF::PDF::xfxQ2(int, double, double) const
20.86	0.97	41 040 980	LHAPDF::GridPDF::_xfxQ2(int, double, double) const
20.01	0.52	55 708 952	std::basic_ios<char, std::char_traits<char> >::init(std::basic_streambuf<char, std::char_traits<char> >*)
18.78	1.67	41 040 979	LHAPDF::Interpolator::interpolateXQ2(int, double, double) const
18.68	1.00	117 563	Pythia8::SimpleTimeShower::pTnext(Pythia8::Event&, double, double, bool, bool)
17.90	0.77	55 708 962	std::basic_ios<char, std::char_traits<char> >::_M_cache_locale(std::locale const&)
17.07	1.45	15 000 346	Pythia8::SimpleTimeShower::pT2nextQCD(double, double, Pythia8::TimeDipoleEnd&, Pythia8::Event&)
13.73	0.09	356 546	Pythia8::MultipartonInteractions::sigmaPT2scatter(bool)
13.65	5.94	334 323 208	__dynamic_cast
12.29	3.56	41 040 980	LHAPDF::LogBicubicInterpolator::_interpolateXQ2(LHAPDF::KnotArray1F const&, double, unsigned long, double...
11.01	0.36	187 617 184	log
10.65	10.65	187 617 189	__ieee754_log_avx
10.27	5.30	98 633 755	std::_Rb_tree<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::pair<std::...
10.21	0.00	117 559	Pythia8::MultipartonInteractions::pTnext(double, double, Pythia8::Event&)
9.29	0.34	18 502 280	LHAPDF::PDFInfo::get_entry(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > c...
9.29	0.34	18 512 629	std::istream& std::istream::_M_extract<double>(double&)
8.57	0.68	18 512 631	std::num_get<char, std::istreambuf_iterator<char, std::char_traits<char> > >::do_get(std::istreambuf_iterato...
5.53	5.53	664 977 286	__memcmp_sse4_1
5.52	1.02	36 764 414	Pythia8::BeamParticle::xCompFrac(double)
4.89	0.35	43 003 994	pow
4.61	2.50	111 487 096	__cxxabiv1::__vmi_class_type_info::__do_dyncast(long, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::...
4.53	3.01	43 003 995	__ieee754_pow_sse2
4.11	4.11	280 061 140	LHAPDF::(anonymous namespace)::_dxf_dlogx(LHAPDF::KnotArray1F const&, unsigned long, unsigned long)
4.11	0.00	1	Pythia8::Pythia::init()
3.89	0.49	55 708 962	bool std::has_facet<std::ctype<char> >(std::locale const&)
3.86	0.45	55 778 132	std::ctype<char> const& std::use_facet<std::ctype<char> >(std::locale const&)
3.65	0.20	18 512 630	void std::__convert_to_v<double>(char const*, double&, std::_Ios_Iostate&, __locale_struct* const&)
3.63	0.00	2 292 302	Pythia8::LHAPDF::xf(int, double, double)
3.62	0.05	2 292 302	Pythia8::PDF::xf(int, double, double)
3.59	0.00	1	Pythia8::PartonLevel::init(Pythia8::Info*, Pythia8::Settings&, Pythia8::ParticleData*, Pythia8::Rndm*, Pythia8::B...
3.59	0.00	1	Pythia8::MultipartonInteractions::init(bool, int, Pythia8::Info*, Pythia8::Settings&, Pythia8::ParticleData*, Pythia...
3.57	0.00	1	Pythia8::MultipartonInteractions::jetCrossSection()
3.45	2.00	18 587 783	___strtod_l_internal
3.45	0.02	18 512 630	strtod_l

>30 % spent in `LHAPDF::lexical_cast`

[f_lhapdf] first glance at [HardQCD+LHAPDF6]



- many calls to `pdf->xMin()`, `pdf->xMax()`, `pdf->q2Min()`, `pdf->q2Min()` in `LHAPDF6::xfUpdate()`. These LHAPDF functions return boundary values(metadata) for PFG set converting strings to numbers.

Solution:

- Introduction of class members `_xMin`, `_xMax`, `_q2Min`, `_q2Min`
- `pdf->xMin()`, `pdf->xMax()`, `pdf->q2Min()`, `pdf->q2Min()` moved to initialization step.
- protection will be also implemented in LHAPDF6.

Save 40 % of CPU time
(when use LHAPDF6)

[f_xCompFrac*] BeamParticle::xCompFrac() refactor

f_xCompFrac

- BeamParticle::xValFrac() and BeamParticle::xCompFrac() inlined
- Number of calls log(xs) reduced to one.

```
553         case 2:
554            return xs * ( (1. - xs) * (19. + xs * (43. + 4. * xs))
555 -           + 6. * log(xs) * (1. + 6. * xs + 4.*xs*xs) ) /
556           ( 4. * ( (xs - 1.) * (1. + xs * (4. + xs) )
557 -         - 3. * xs * log(xs) * (1 + xs) ) );
```

f_xCompFrac2

Sequentially called xCompFrac() often uses the same input parameters;

- Add static allocated cache for previous computed result.

[f_refactor] [f_refactor2] `BeamParticle::xfModified()` refactor

`f_refactor`

- cached size of resolved vector used, as it is not changing in the function
- conditional expressions moved outside loops
- different loops with same iteration conditions joined

`f_refactor2`

- branching reduced in `xfModified()` by refactoring of large `else{...}` branch.

[f_refactor3] `BeamParticle::xfModified()` refactor

Overview:

- Often `xfModified()` is called inside `for` loops with constant `iSkip` and `Q2` parameters, causing re-calculations of identical lines;
- `xfModified()` is called in other classes via `BeamParticle::xfISR()` and `BeamParticle::xfMPI()`;

Optimizations

- Move almost all `iSkip` and `Q2` related code from `xfModified()` to new function `xfModifiedPrepare()` which returns struct with precomputed values.
- Other classes use precomputed results where possible and pass the struct to `xfModified()`;
- Several `for` loops are refactored;
- Original BeamParticle interface isn't changed but extended.

Save 10 % of CPU time

[f_polint] polynomial interpolation - `NNPDF::polint()`

- Pythia8: general (Neville's?) algorithm for general case (polynomials of degree n)
- Used with Pythia8 NNPDF class
- Pythia8 spent 23% of CPU cycles here

Modifications:

- pre-computed functions for 2 and 4 points (as only these cases are used internally in Pythia8)
- `NNPDF::polint()` is rewritten using Neville's algorithm (optimized in CPU instruction terms, used for n != 2, n != 4)
- inlining `Pythia8::polint*` functions (they are just expressions)
- using pure functions
- creation and filling of temporary array removed

Save 13 % when NNPDF class used

[f_lhagrid] refactoring of LHAGrid1 class

Rotate `pdfGrid[iid][x][q]` matrix to reduce read cache misses;
CPU reads data left-to-right sequentially. 'Jump' to next matrix row causes cache miss.

$$\begin{bmatrix} a_{0,0} & \dots & a_{0,n3q} \\ \vdots & \ddots & \vdots \\ a_{3,0} & \dots & a_{3,n3q} \end{bmatrix} \Rightarrow \begin{bmatrix} a_{0,0} & \dots & a_{0,4} \\ \vdots & \ddots & \vdots \\ a_{n3q,0} & \dots & a_{n3q,4} \end{bmatrix}$$

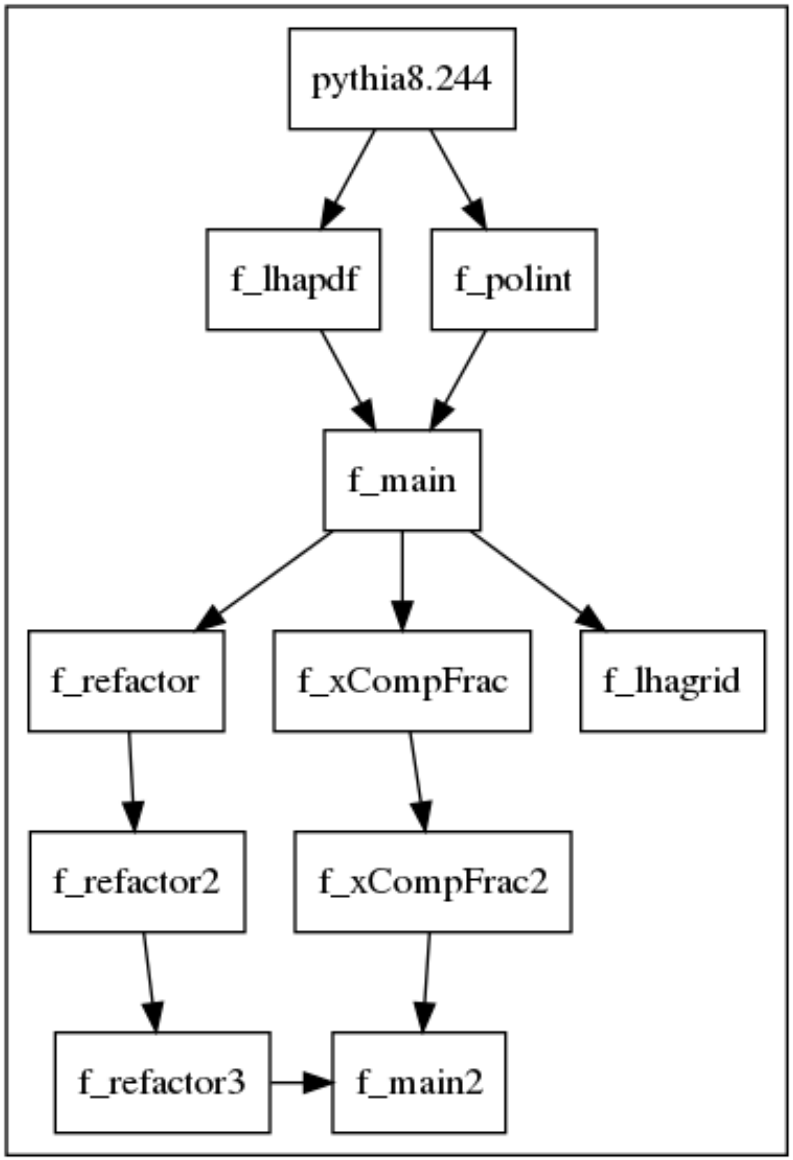
Before — Always 4 reading cache miss

After — 1/2/4 reading cache miss

Manually unrolled `for` loop can be optimized by compiler.
Numerical result is changed (probably) due to differences in `double` arithmetic. Physics validation needed

Save 4.5 % when Pythia8 interpolation used

Patches structure



Results: LHAPDF6

Branch	User time, sec	$\frac{\Delta t}{t_{Pythia8}}$, %
Pythia8.244	303.8	0.0
f_lhapdf	181.2	40.3
f_polint	304.9	−0.4
f_main	181.7	40.2
f_refactor	176.2	42.0
f_refactor2	176.6	41.9
f_refactor3	153.6	49.4
f_xCompFrac	173.0	43.0
f_xCompFrac2	171.0	43.7
f_lhagrid	180.7	40.5
f_main2	153.3	49.5

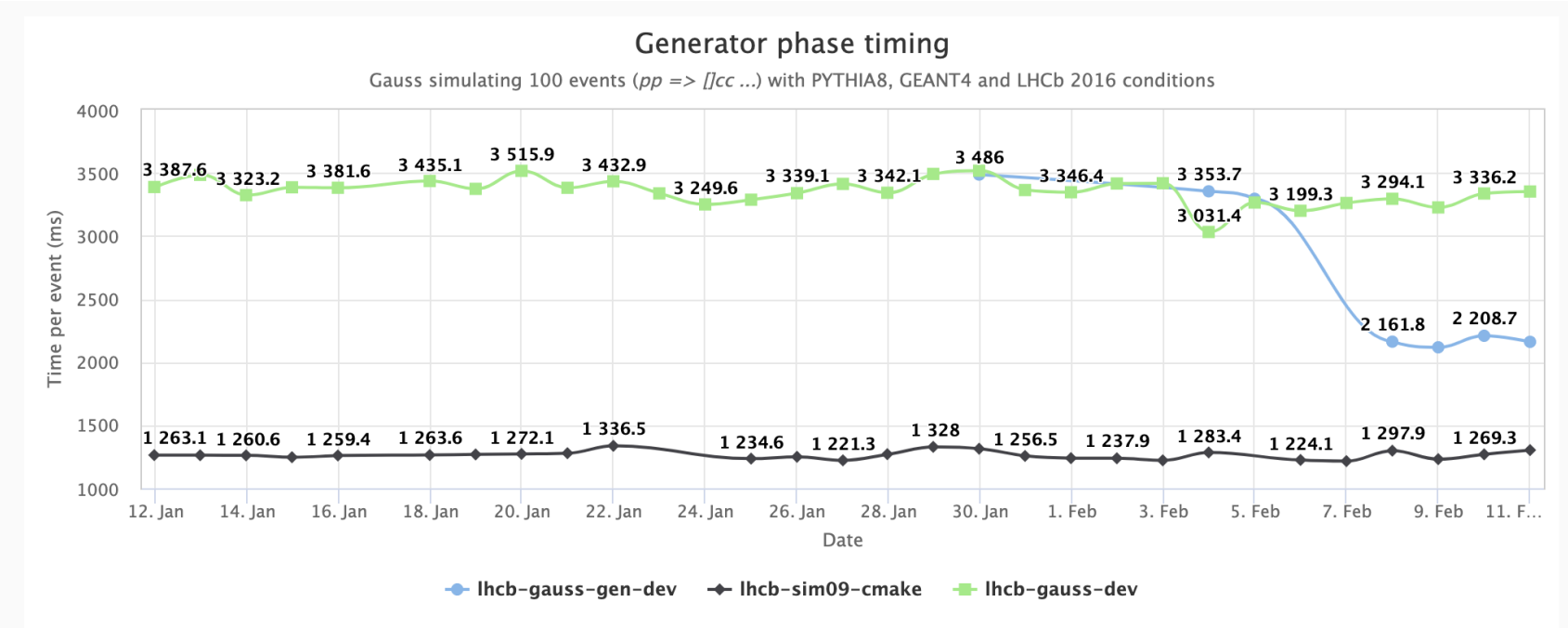
Results: NNPDF

Branch	User time, sec	$\frac{\Delta t}{t_{Pythia8}}$, %
Pythia8.244	157.5	0.0
f_lhapdf	149.5	0.0
f_polint	137.2	12.9
f_main	136.9	13.1
f_refactor	130.1	17.2
f_refactor2	130.2	17.3
f_refactor3	106.4	32.4
f_xCompFrac	127.2	19.2
f_xCompFrac2	125.5	20.3
f_lhagrid	135.2	14.1
f_main2	106.2	32.6

Results: LHAGrid1

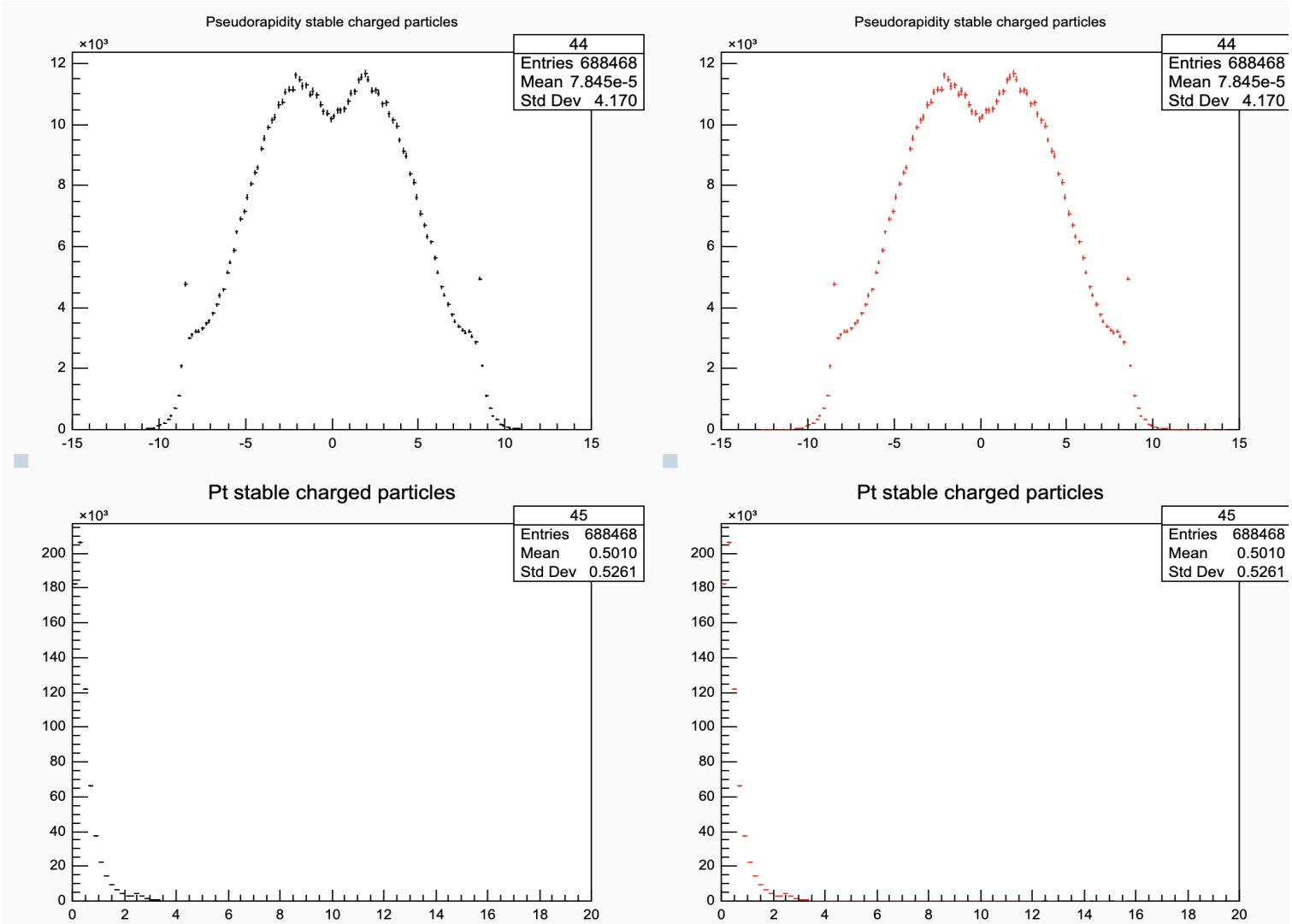
Branch	User time, sec	$\frac{\Delta t}{t_{Pythia8}}$, %
Pythia8.244	126.8	0.0
f_lhapdf	126.9	−0.1
f_polint	128.2	−1.1
f_main	127.8	−0.8
f_refactor	121.8	3.9
f_refactor2	122.0	3.8
f_refactor3	98.3	22.4
f_xCompFrac	118.1	6.8
f_xCompFrac2	116.7	7.9
f_lhagrid	121.0	4.5
f_main2	98.3	22.5

Validation in LHCb nightly tests with Pythia8240 (Gloria Corti)



32 % reduction seen by LHCb in different process.

Validation in LHCb nightly tests (Gloria Corti)



No numerical changes in physical results.

LHAPDF-6.2.3 - LogBicubicInterpolator

- 21 % spent in LHAPDF::LogBicubicInterpolator::_interpolateXQ2
- LHAPDF::LogBicubicInterpolator::_interpolateXQ2 is called 10 times (from LHAPDF6 indirectly) with same x and Q2 but with different PID.

```
// Give the parton distribution function set from LHAPDF6.

void LHAPDF6::xfUpdate(int, double x, double Q2) {

  // Freeze at boundary value if PDF is evaluated outside the fit region.
  if (x <_xMin && !extrapol) x =_xMin;
  if (x >_xMax)    x =_xMax;
  if (Q2 <_q2Min) Q2 =_q2Min;
  if (Q2 >_q2Max) Q2 =_q2Max;

  // Update values.
  xg     = pdf->xfxQ2(21, x, Q2);
  xu     = pdf->xfxQ2(2,  x, Q2);
  xd     = pdf->xfxQ2(1,  x, Q2);
  xs     = pdf->xfxQ2(3,  x, Q2);
  xubar  = pdf->xfxQ2(-2, x, Q2);
  xdbar  = pdf->xfxQ2(-1, x, Q2);
  xsbar  = pdf->xfxQ2(-3, x, Q2);
  xc     = pdf->xfxQ2(4,  x, Q2);
  xb     = pdf->xfxQ2(5,  x, Q2);
  xgamma = pdf->xfxQ2(22, x, Q2);

  // Subdivision of valence and sea.
  xuVal  = xu - xubar;
  xuSea  = xubar;
  xdVal  = xd - xdbar;
  xdSea  = xdbar;

  // idSav = 9 to indicate that all flavours reset.
  idSav = 9;
}
```

LogBicubicInterpolator::_interpolateXQ2

```
const double logx = log(x);
const double logq2 = log(q2);

// Fall back to LogBilinearInterpolator if either 2 or 3 Q-knots
if (nq2knots < 4) {
  // First interpolate in x
  const double logx0 = subgrid.logxs()[ix];
  const double logx1 = subgrid.logxs()[ix+1];
  const double f_ql = _interpolateLinear(logx, logx0, logx1, subgrid.xf(ix, iq2), subgrid.xf(ix+1, iq2));
  const double f_qh = _interpolateLinear(logx, logx0, logx1, subgrid.xf(ix, iq2+1), subgrid.xf(ix+1, iq2+1));
  // Then interpolate in Q2, using the x-ipol results as anchor points
  return _interpolateLinear(logq2, subgrid.logq2s()[iq2], subgrid.logq2s()[iq2+1], f_ql, f_qh);
}
// else proceed with cubic interpolation:

// Pre-calculate parameters
/// @todo Cache these between calls, re-using if x == x_prev and Q2 == Q2_prev
const double dlogx_1 = subgrid.logxs()[ix+1] - subgrid.logxs()[ix];
const double tlogx = (logx - subgrid.logxs()[ix]) / dlogx_1;
const double dlogq_0 = (iq2 != 0) ? subgrid.logq2s()[iq2] - subgrid.logq2s()[iq2-1] : -1; //< Don't evaluate (or
const double dlogq_1 = subgrid.logq2s()[iq2+1] - subgrid.logq2s()[iq2];
const double dlogq_2 = (iq2+1 != iq2max) ? subgrid.logq2s()[iq2+2] - subgrid.logq2s()[iq2+1] : -1; //< Don't eva
const double tlogq = (logq2 - subgrid.logq2s()[iq2]) / dlogq_1;
```

Introducing “static” for logx, logq2 + for all variables mentioned by Andy in @todo and reusing them for consequent calls with same x and Q2.

Save 10 % on top of patched Pythia8 version

Pythia83XX: VINCIA and DIRE - part of Pythia8

- VINCIA (VIrtual Numerical Collider with Interleaved Antennae) is a versatile dipole-antenna shower program, by Peter Skands and collaborators.
- DIRE (DIpole REsummation) is another dipole shower program for initial and final state radiation, by Stefan Prestel.

From VINCIA callgrind - 30 % of CPU time spent in new/delete/malloc:

- many vectors are used without allocation capacity (reserve())
- usage of map<int, T> instead of simple array or vector.

Fixing it we got 20 % speedup for VINCIA.

Summary

- three configurations are bench-marked and optimized:

Configuration	Pythia8.244, sec	f_main2, sec	$\frac{\Delta t}{t_{Pythia8}}$, %
hardQCD+LHAPDF6	303.8	153.3	49.5
hardQCD+LHAGrid1	126.8	98.3	22.5
hardQCD+NNPDF	157.5	106.2	32.6

- All patches are applicable to previous Pythia8 versions as well as to Pythia83XX
- few patches are changing numerical results and more thorough validation is required.
- All patches are sent to Torbjörn Sjöstrand and will be included into next Pythia83X release.
- additional 10 % can be gained from modification of LHAPDF6 (hopefully will be included into next LHAPDF6 release).

Notes

- Likely more speed-up can be achieved by revision of Pythia8 logic but it is much more work.
- continuous profiling is needed during development stage (better to have it from very beginning)

- Many thanks to everyone who helped us: Gloria Corti and LHCb, Philip Ilten, Torbjörn Sjöstrand, Peter Skands, Efe Yazgan, Ivan Razumov, Mihaly Novak, Richard Bachman, Pere Mato, Gerri Ganis.

Backup - xfModified

xfModified and xfModifiedPrepareData

```
494 -   double xfModified( int iSkip, int idIn, double x, double Q2);
```

```
510 +   double xfModified( int iSkip, int idIn, double x, double Q2){
511 +     xfModifiedPrepareData pre = xfModifiedPrepare(iSkip, Q2);
512 +     return xfModified(iSkip, idIn, x, Q2, pre);
513 +   }
514 +   double xfModified( int iSkip, int idIn, double x, double Q2, x
    fModifiedPrepareData& data);
515 +   // in case of resolved.size() === 0
516 +   double xfModified0( int iSkip, int idIn, double x, double Q2);
```

Use-case:

```
826 -       xPDFdaughter = beam.xfISR(iSysNow, idDaughter, xDaughter,
    pdfScale2);
827         if (xPDFdaughter < TINYPDF) {
828           xPDFdaughter  = TINYPDF;
829           hasTinyPDFdau = true;
...   @@ -844,13 +845,11 @@ void SimpleSpaceShower::pT2nextQCD( double
    pT2begDip, double pT2endDip) {
844
845           // Parton density of potential quark mothers to a g.
846           xPDFmotherSum = 0.;
847           for (int i = -nQuarkIn; i <= nQuarkIn; ++i) {
848 -           if (i == 0) {
849 -             xPDFmother[10] = 0.;
850 -           } else {
851 -             xPDFmother[i+10] = beam.xfISR(iSysNow, i, xDaughter,
    pdfScale2);
852               xPDFmotherSum += xPDFmother[i+10];
853 -           }
854         }
```

```
826 +       xfModifiedPrepareData xfprepared = beam.xfModifiedPrepare
    (iSysNow, pdfScale2);
827 +       xPDFdaughter = beam.xfISR(iSysNow, idDaughter, xDaughter,
    pdfScale2, xfprepared);
828         if (xPDFdaughter < TINYPDF) {
829           xPDFdaughter  = TINYPDF;
830           hasTinyPDFdau = true;
...   @@ -844,13 +845,11 @@ void SimpleSpaceShower::pT2nextQCD( double
    pT2begDip, double pT2endDip) {
845
846           // Parton density of potential quark mothers to a g.
847           xPDFmotherSum = 0.;
848 +         xPDFmother[10] = 0.;
849           for (int i = -nQuarkIn; i <= nQuarkIn; ++i) {
850 +           if (i == 0) continue;
851 +             xPDFmother[i+10] = beam.xfISR(iSysNow, i, xDaughter,
    pdfScale2, xfprepared);
852               xPDFmotherSum += xPDFmother[i+10];
853         }
```