Herwig Parallel

Radek Podskubka

ITP, KIT

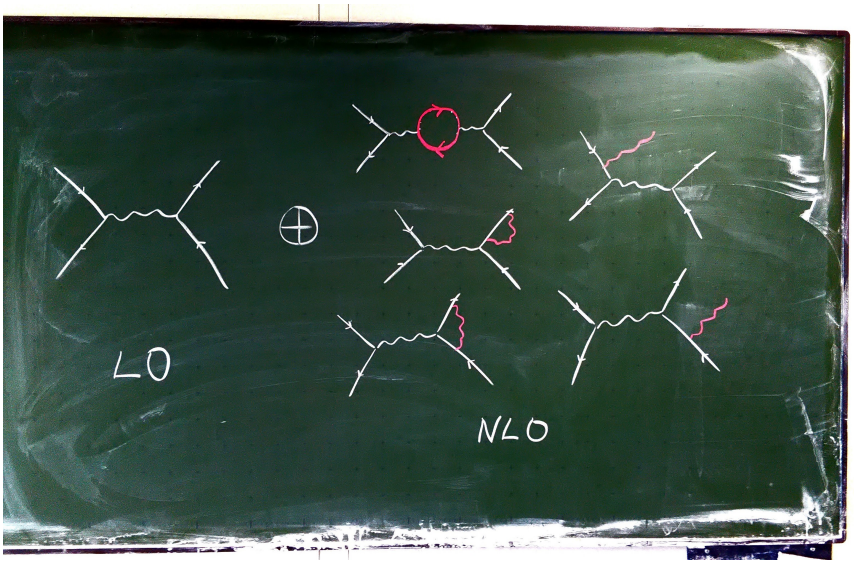
April 6, 2016

Outline

1. Motivation
   - Why do we need parallelization?
   - What actually is Herwig-Parallel?
2. Monte Carlo method and statistics
   - Error calculation derivation
   - Error calculation in Rivet
   - Patching Rivet
   - YODA output combination
3. Example
4. Conclusions

Why do we need parallelization?

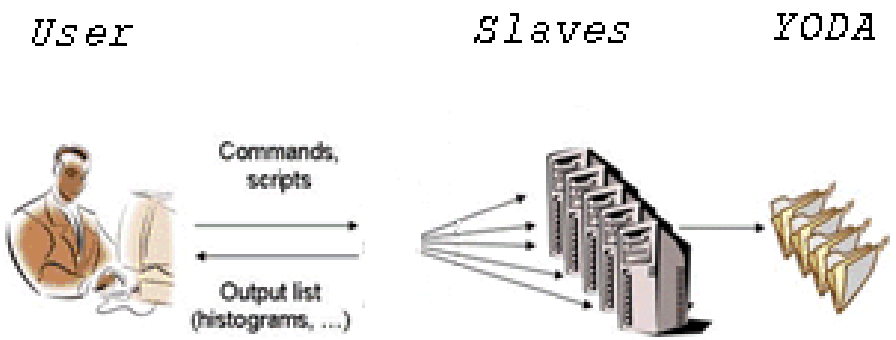
- With collider experiments running on higher energies we need a higher precision calculations to describe data
- In Herwig 7 we automatize calculation of matrix element on NLO level and successfully combine NLO matrix element with parton shower to achieve higher accuracy



Why do we need parallelization?

- The other requirement for high quality results is lowering statistical uncertainties to desired level which implies need of rich event samples

- For higher order calculations and production of big event samples we have to pay a price in high computation time.



What actually is Herwig-Parallel?

- Bunch of Python scripts that ensure sending jobs to the cluster, monitoring the progress and correct combining the final results
- Read Daniels thesis
  https://www.itp.kit.edu/en/publications/diploma
- Patch for Rivet 2.4.0 and YODA 1.5.5 which book-keeps some additional values important for correct combining
- To understand why do we need to patch Rivet lets recall some basics of Monte Carlo method and statistics

Error calculation

- Definition of generalized Riemann integral in d dimensions

$$\Sigma = \int_{\Omega} g\left(\vec{x}\right) \mathrm{d}^{d}\vec{x} = \lim_{N\to\infty} \frac{V}{N} \sum_{i=1}^{N} g\left(\vec{x_i}\right) \tag{1}$$

- If we replace grid of cells $\vec{x_i}$ by uniformly distributed sequence of random numbers $\vec{r_i}$ we get a good approximation of the integral Σ

$$\sigma = \frac{V}{N} \sum_{i=1}^{N} g\left(\vec{r_i}\right) \tag{2}$$

- It is possible to show

$$\mathrm{Var}\left[\sigma\right] = \frac{V^2}{N} \mathrm{Var}\left[g\right] \tag{3}$$

Error calculation derivation

- With the fact that the standard deviation of a sample of N points is given by

$$\frac{1}{\sqrt{N-1}}\sqrt{\sum_{i=1}^{N}\left(g_i-\bar{g}\right)^2} \tag{4}$$

- By combining (3) and (4) we get the equation for calculation of cross section uncertainty

$$\delta_\sigma = \frac{V}{\sqrt{N\left(N-1\right)}}\sqrt{\sum_{i=1}^{N} g_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} g_i\right)^2} \tag{5}$$

Error calculation in Rivet

- In terms of quantities available in YODA output files we can write

$$\delta_\sigma = \sqrt{\frac{N}{N-1}} \sqrt{S_{w^2} - \frac{1}{N} S_w^2} \tag{6}$$

- Lets have a look how errors are calculated in Rivet

```
/// Error computed using binomial statistics
/// on the sum of bin weights,
/// i.e. err_area = sqrt{sum{weights}}
double areaErr() const {
        return sqrt(sumW2());
}
```

- Which written as equation is

$$\delta_\sigma = \sqrt{S_{w^2}} \tag{7}$$

Patching Rivet

- In terms of quantities available in YODA output files we can write

$$\delta_\sigma = \sqrt{\frac{N}{N-1}}\sqrt{S_{w^2} - \frac{1}{N}S_w^2} \tag{8}$$

- What is the quantity N? It is not a number of events

Patching Rivet

```
BEGIN YODA_HISTO1D /ATLAS_2012_I1082936/d01-x01-y01
Path=/ATLAS_2012_I1082936/d01-x01-y01
ScaledBy=7.2491398895521045e-07
Title=
Type=Histo1D
XLabel=
YLabel=
# Mean: 3.290286e+02
# Area: 1.473990e+02
# ID      ID      sumw     sumw2    sumwx    sumwx2   numEntries     numAttempts
Total           Total           1.473990e+02    4.664740e+01    4.849849e+04    2.413886e+07    325833  1379474
Underflow       Underflow       0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00    0       1379474
Overflow        Overflow        0.000000e+00    0.000000e+00    -3.582429e-01   -5.380857e+02   46      1379474
# xlow    xhigh    sumw     sumw2    sumwx    sumwx2   numEntries     numAttempts
2.000000e+01    3.000000e+01    1.361645e+01    1.902168e+00    3.304397e+02    8.116876e+03    34686   1379474
3.000000e+01    4.500000e+01    1.174140e+01    2.775334e+00    4.283559e+02    1.584152e+04    28926   1379474
4.500000e+01    6.000000e+01    7.060193e+00    1.148506e+00    3.677388e+02    1.928084e+04    17559   1379474
6.000000e+01    8.000000e+01    5.835322e+00    2.193023e+00    4.236510e+02    3.089121e+04    15794   1379474
8.000000e+01    1.100000e+02    6.704028e+00    5.688859e-01    6.279001e+02    5.935778e+04    15802   1379474
1.100000e+02    1.600000e+02    7.103806e+00    8.528563e-01    9.617835e+02    1.317594e+05    16364   1379474
1.600000e+02    2.100000e+02    6.573426e+00    1.672009e+00    1.243521e+03    2.362267e+05    11008   1379474
2.100000e+02    2.600000e+02    5.557652e+00    4.813307e-01    1.306648e+03    3.082494e+05    8498    1379474
2.600000e+02    3.100000e+02    5.266879e+00    9.779449e-01    1.503049e+03    4.300405e+05    7048    1379474
3.100000e+02    4.000000e+02    8.927710e+00    2.040810e+00    3.178028e+03    1.138938e+06    12505   1379474
4.000000e+02    5.000000e+02    2.192993e+01    4.817699e+00    1.009915e+04    4.667834e+06    23346   1379474
5.000000e+02    6.000000e+02    2.994596e+01    1.071454e+01    1.616343e+04    8.748074e+06    44054   1379474
6.000000e+02    8.000000e+02    1.498700e+01    2.926938e+00    9.989734e+03    6.698905e+06    57640   1379474
8.000000e+02    1.000000e+03    2.057810e+00    1.678012e-01    1.776758e+03    1.539417e+06    29664   1379474
1.000000e+03    1.200000e+03    9.348342e-02    5.213386e-04    1.009053e+02    1.092316e+05    2445    1379474
1.200000e+03    1.500000e+03    -1.827045e-03   5.297820e-05    -2.245141e+00   -2.765102e+03   448     1379474
END YODA_HISTO1D
```

Patching Rivet

- Rivet

  Analysis.h
  AnalysisHandler.h

  Analysis.cc
  AnalysisHandler.cc

- ThePEG

  NLORivetAnalysis.cc
  RivetAnalysis.cc

- YODA

  AnalysisObject.h
  Axis1D.h
  Bin1D.h
  Counter.h
  Dbn0D.h
  Dbn1D.h
  Histo1D.h
  HistoBin1D.h

  Dbn0D.cc
  Dbn1D.cc
  Histo1D.cc
  ReaderYODA.cc
  WriterYODA.cc

  declarations.pxd
  Counter.pyx
  Histo1D.pyx

YODA output combination

- How to combine multiple yodafiles which are output of parallel runs?
- With Herwig-Parallel we can make use of number of attempts value
- It is possible to show that for M fully independent runs

$$S_w = \frac{1}{N}\sum_{j=1}^{M} N_{(j)} S_w^{(j)} \qquad S_{w^2} = \frac{1}{N^2}\sum_{j=1}^{M} N_{(j)}^2 S_{w^2}^{(j)} \tag{9}$$

Example

- Running parallel jobs after successful installation is very straightforward
- First step - build and integration

```
herwig-parallel-read -j 10 -Q local -q ITP-short -n Example LHC-Example.in
```

```
-j Number of integration jobs
-Q Build queue
-q Integration queue
-n Run name
```

- We immediately see the progress of integration

Example

- Second step - event production

```
herwig-parallel-run -e 100000 -j 5 -q ITP-albatros Example/
-e Number of events per job
-j Number of event production jobs
-q Event production queue
```

- To check the progress just write

```
herwig-parallel-monitor Example/
```

Example

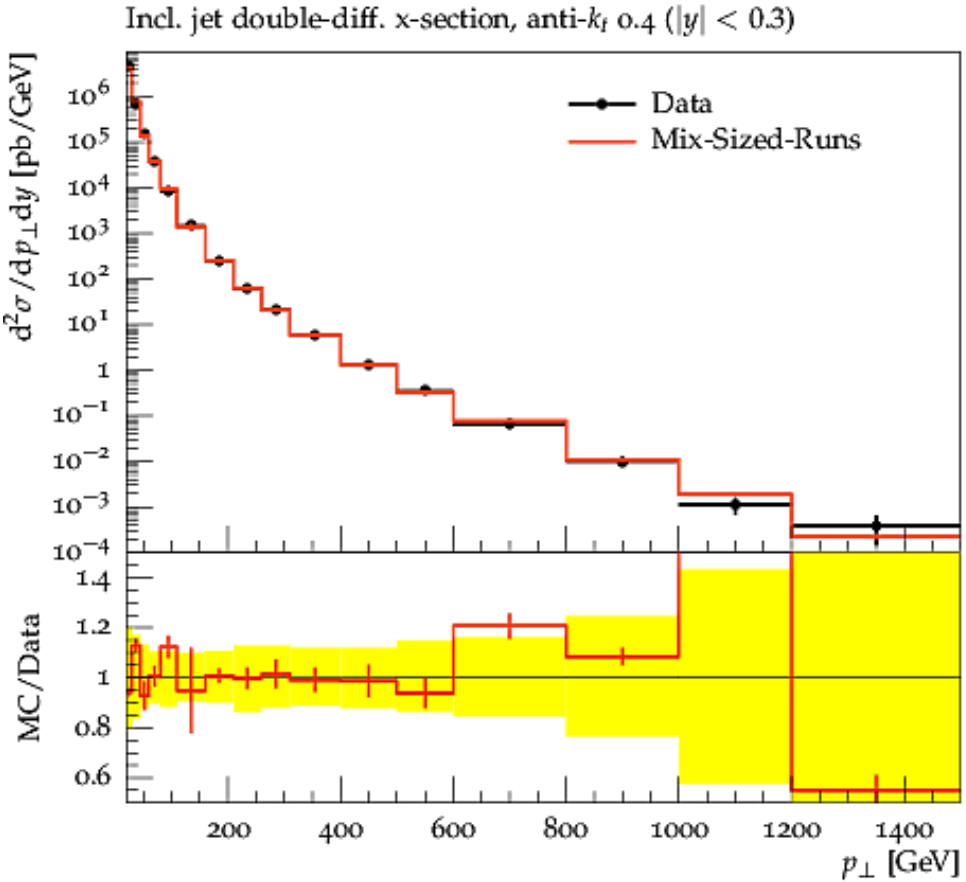
```
====================================
HERWIG-PARALLEL: Status of run Example
====================================

--------------
run information
--------------
read step:    complete (10/10 integration jobs finished, 0 queued, 0 running)

location:          /itp/fast/radek/Example/Example
infile:            /itp/fast/radek/Example/LHC-Example.in
setupfile:
build script:      /itp/fast/radek/Example/herwig-parallel-build.sh
integrate script:  /itp/fast/radek/Example/herwig-parallel-integrate.sh
run script:        /itp/fast/radek/Example/herwig-parallel-run.sh

------------------------
status of individual jobs
------------------------
job #1 with id 175640 (itpalbatros20):   7.0% done, xs =  307585040.920000 pb +/- 6017997.605600 pb (739087532.174054)
job #2 with id 175641 (itpalbatros15):  11.0% done, xs =  298983520.860000 pb +/- 4280640.386400 pb (659829856.187220)
job #3 with id 175642 (itpalbatros20):   7.0% done, xs =  307452873.540000 pb +/- 5354574.072400 pb (658873757.866654)
job #4 with id 175643 (itpalbatros15):  11.0% done, xs =  293273397.550000 pb +/- 4015289.029600 pb (620683684.706105)
job #5 with id 175644 (itpalbatros16):   9.0% done, xs =  296602274.750000 pb +/- 4771302.367900 pb (668799768.888952)

-----------------
consistency checks
-----------------
median cross section:        298983520.860000 pb
median integrand variance:   659829856.187220

criteria for consistency limits:
- standard deviations wrt. to median cross section:   4.0
- error factor wrt. median integrand variance:        2.0

=> no inconsistent jobs found with given criteria

-------
summary
-------
total number of jobs:                                5
- unprocessed jobs (not sent to cluster):            0
- queued jobs (not yet started):                     0
- running jobs (but no events generated yet):        0
- running jobs (generating events):                  5   [1 2 3 4 5]
- finished jobs:                                     0
- jobs crashed/aborted in read phase                 0
- jobs crashed/aborted in run phase                  0
- jobs with unknown status:                          0

---------------------------------
combination of total cross sections
---------------------------------
current progress:   9.0% (45000 of 500000 events done)
current result:   299749883.774179 pb +/- 2129972.858553 pb ( = 0.710583%)
```

Example

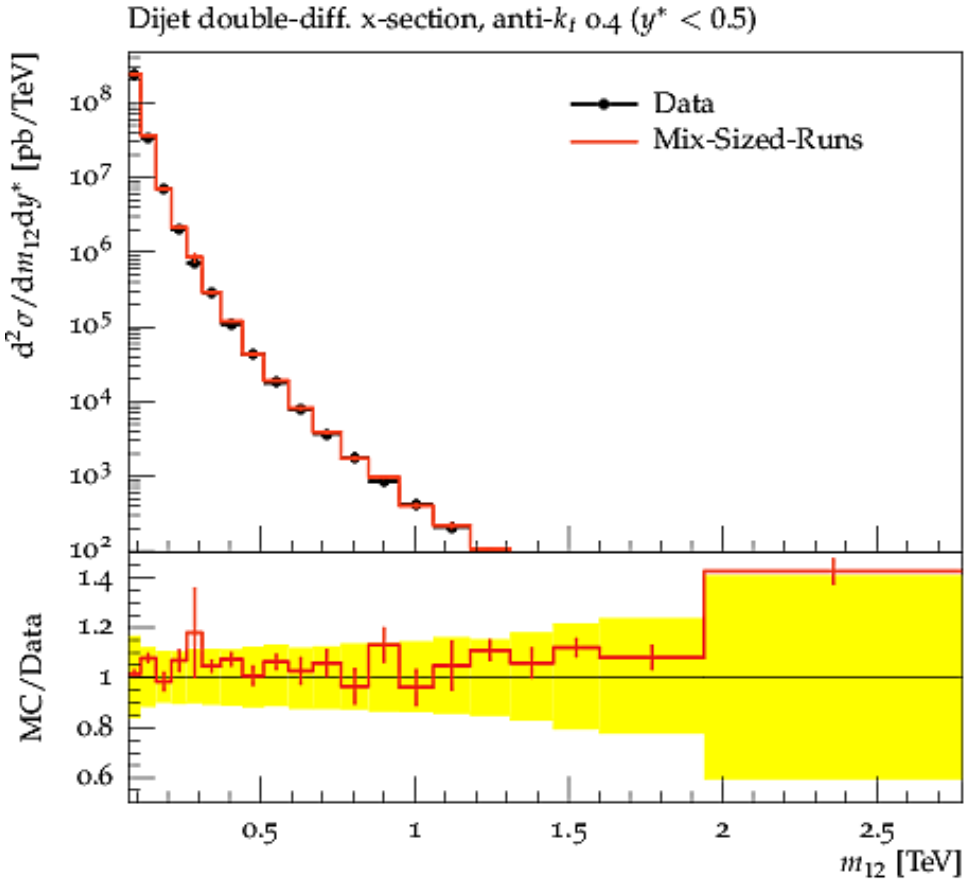
- Last step - after all runs are finished we combine them

```
herwig-parallel-combine Example/
```

Example


Incl. jet double-diff. x-section, anti-k_t 0.4 ($|y| < 0.3$)
Data
Mix-Sized-Runs
$d^2\sigma/dp_\perp dy$ [pb/GeV]
MC/Data
$p_\perp$ [GeV]

Example


Dijet double-diff. x-section, anti-k_t 0.4 ($y^* < 0.5$)
Data
Mix-Sized-Runs
$d^2\sigma/dm_{12}dy^*$ [pb/TeV]
10^8
10^7
10^6
10^5
10^4
10^3
10^2
MC/Data
1.4
1.2
1
0.8
0.6
0.5
1
1.5
2
2.5
m_{12} [TeV]

Conclusions

- Herwig-Parallel was presented
- Formula for proper error calculation was derived
- Our patch fixes error calculations in Rivet