Parallel computing with WHIZARD using MPI and the upcoming PYTHIA 8 interface

Improving WHIZARD

Simon Braß

June 26, 2018

Universität Siegen – Theoretische Physik I
http://www.tp.nt.uni-siegen.de/+brass/

Outline

WHIZARD - General Purpose Event Generator

Parallel Multi-channel Monte Carlo integration

(Upcoming) Pythia8 Interface

WHIZARD - General Purpose Event Generator

WHIZARD

- General purpose event generator for multi-particle states [W. Kilian 2001; Wolfgang Kilian, Ohl, and Jürgen Reuter 2011]
    - Hadron and lepton collider
- Under active development:
    - Current version: 2.6.3, `http://whizard.hepforge.org/`
    - Contact/help: whizard@desy.de
    - Team: Wolfgang Kilian, Thorsten Ohl, Jürgen Reuter
        - *SB*, Vincent Rothe, Christian Schwinn, Marco Sekulla, So Young Shim, Pascal Stienemeier, Zhijie Zhao + Master students
- Programming languages:
    - Fortran 2008 (gfortran ≥ `4.8.4`)
    - OCaml (≥ `3.12.0`)
- Standard installation:
    - `configure [FLAGS]`, `make`, `[make check]`, `make install`
- Large self test suite, unit tests [module tests], regression testing
    - Continous integration system (`gitlab CI` @Siegen)

WHIZARD technical overview

- optimized tree matrix element generator `O'Mega` [Moretti, Ohl, and Jurgen Reuter 2001]
- Generator/simulation tool for *lepton collider* beam spectra `CIRCE1/2` [Ohl 1997]
- Interfaces to external packages for Feynman rules, hadronization, tau decays, event formats, analysis, jet clustering etc.:
  - `FastJet, GoSam, GuineaPig(++), HepMC, HOPPET, LCIO, LHAPDF(5/6), LoopTools, OpenLoops, PYTHIA6 [internal], PYTHIA8 (upcoming), Recola, StdHep [internal], Tauola [internal]`
- Event formats: `LHE`, `StdHEP`, `HepMC`, `LCIO` + several `ASCII`

Steering WHIZARD

- Scattering processes and decays
- Factorized processes with spin correlations [variants: no correlations, definite helicity, predefined branching ratios]
- Scripting language for the steering: SINDARIN
- Beam structure: polarization, asymmetric beams, crossing angle, structured beams, decays

```
beams = e1, E1
beams_pol_density = @(-1), @(+1)
beams_pol_fraction = 80%, 30%
```

```
beams = p, pbar => lhapdf
$lhapdf = “NNPDF3”
```

```
beams = e1, E1 => circe2 => isr => ewa
```

WHIZARD NLO automatation (ALPHA STATUS)

- NLO automatation with interfaces to *external* (virtual) ME provider:
  - RECOLA [Actis et al. 2013; Denner, Lang, and Uccirati 2018]
  - GOSAM [Soden-Fraunhofen 2015]
  - OpenLoops [Cascioli, Maierhofer, and Pozzorini 2012]
- FKS subtraction [Frixione, Kunszt, and Signer 1996]
  - Real and virtual subtraction terms internal
- Resonance-aware treatment [Ježo and Nason 2015]
  - enables consistent matching of fixed-order NLO with parton shower for processes with *intermediate* resonances (e.g. H)
- NLO decays available for the NLO processes
- Fixed order events for plotting (weighted)
- Automated POWHEG damping and matching
- Still under validation

```
alpha_power = 2
alphas_power = 0
process eett = e1, E1 => t, tbar { nlo_calculation = full }
```

Phasespace Parametrization

- Phasespace configuration with `cascades` *very* slow
- *DAG* representation of Feynman diagrams from `O'Mega` → cascades2 [Utsch 2018]
    - Respects restrictions on matrix elements
    - Includes all available models and not limited to SM
    - Improved runtime over `cascades` up to 215x faster

Sindarin

```
?omega_write_phs_output = true
$phs_method = "fast_wood"
```

Improving runtime ii

Thread-based Parallelization with `OpenMP`

- Loop over helicity amplitudes in `O'Mega` [Nejad, Ohl, and Jürgen Reuter 2015]
- Phasespace configuration in *cascades2* and generation [Utsch 2018]
- Computation of *multi-channel* probability density

Configure

- `OpenMP` supported by most compilers
- Enabled by configuration flag

```
./configure --enable-fc-openmp
```

Improving runtime iii

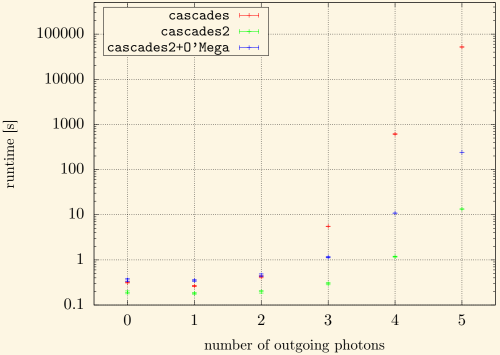


Figure 1: Runtimes of `cascades` and `cascades2`, e.g. $gg \to W^+ t\bar{b} + n\gamma$ [Utsch 2018].

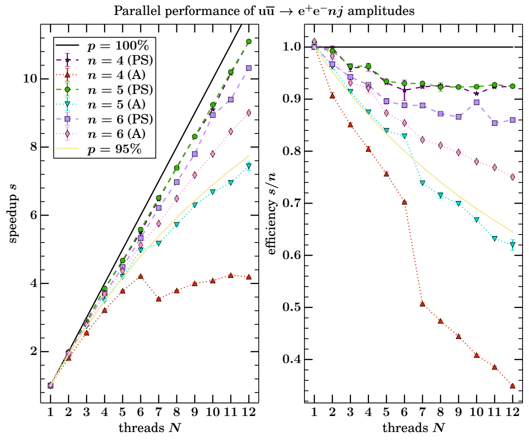


Figure 2: Speedup for *thread-based* parallelization with **`OpenMP`**, e.g. Drell-Yan [Nejad, Ohl, and Jürgen Reuter 2015].

What's next?

- Event generation? → parallelization is trivial, still done
- Monte Carlo integration? → parallelization is non-trivial
  - Twofold-adaptive *multi-channel* Monte Carlo integration
  - Independent random number generation
  - ...

Aspects of *thread-based* Parallelization

- Limited to shared systems, e.g. single machine with multi-core → $N_{\text{Proc}} \leq \mathcal{O}(10)$
- `OpenMP` based on *directives* and implementation is *compiler-dependent*

Further improving? ii

Aspects of *message-passing* Parallelization

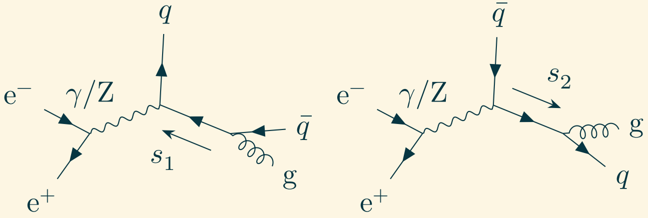
- Limited by physical resources only
- Implemented *directly* in to the source code
- Abstraction of underlying system and topology → Communicators
  - Implementation more complicated than that with `OpenMP`
- Simplification of run management by `mpirun`

Parallel Multi-channel Monte Carlo integration

Multi-channel integration

- Matrix elements have different analytical structures and interference terms, e.g. resonances, final and initial state radiation, ...
- Phasespace parametrization of physical manifold Ω on the unit hypercube U_i → channel $\phi_i : U_i \mapsto \Omega$
- *Partition of unity* $f(x) = \sum_i \alpha_i f(x)$ and $\sum_i \alpha_i = 1$

$$I = \sum_i \alpha_i \int_0^1 f\left(\phi_i(x)\right) \phi_i'(x) \, \mathrm{d}x$$



Process $e^+e^- \to q\bar{q}g$ with Z-resonance and soft/collinear singularities in $s_{1,2}$.

Monte Carlo integration

- Expectation value for $f(x)$ over a *uniform probability density*

$$E(f) = \int f(x)\mathbb{1}_x \,\mathrm{d}\mu(x) \quad \text{and} \quad V(f) \propto 1/N$$

- Importance sampling with $x = G^{-1}(r)$ and $g(x) \propto f(x)$
  - large $f(x)$, large $g(x)$
- Stratified sampling adapts to variance $V(f)$
  - large and rapidly changing $f(x)$, more dense sampling
- Stratified sampling converges *faster* than importance sampling
  - Stratified sampling is unpractical for higher dimensions

VEGAS

- Adaptive algorithm for *factorizable* probability density $g(x)$ [Lepage 1978]

$$g(x) = \prod_i \frac{1}{N \Delta x_i}, \; \Delta x_i = x_{i+1} - x_i \quad \text{and} \quad 0 \leq x_0 \leq \dots \leq x_N \leq 1$$

- Different implementations [Lepage 1980], GSL, VAMP, …
- Importance and Stratified Sampling
    - Automatic swichting between both sample modes
- Integration grid with superficial stratification grid → match under stratified sampling
    - Phasespace division: $\Omega = \cup \Omega_i$
    - Sample each division Ω_i with $N \geq 2$
    - $I = \sum_{i \in I} I_i$

VAMP

- Sample each *channel* with probability density $g_i(\phi_i(x))$
- Adapt $g_i(x)$ with **VEGAS** → Vegas AMPlified
- Overall probability density

$$g(p) = \sum_i \alpha_i g_i(\phi_i^{-1}(p))\phi_i^{-1\prime}(p)$$

- Channel weights α_i are adapted by channel variances

$$\alpha_i \to \alpha_i' = \frac{\alpha_i V_i(f, \vec{\alpha})^\beta}{\sum \alpha_i V_i(f, \vec{\alpha})^\beta}, \quad \beta > 0$$

Parallelization of Integration

VEGAS

- Paralllelization of $\sum_{i \in I} I_i$ with phasespace division $\Omega = U\Omega_i$
- Number of stratas *depends* on number of calls → no division possible
- Workflow
  1. Broadcast grid to all workers
  2. Compute assigned divisions
  3. Master collects results and adapts grid → conserves numerical properties

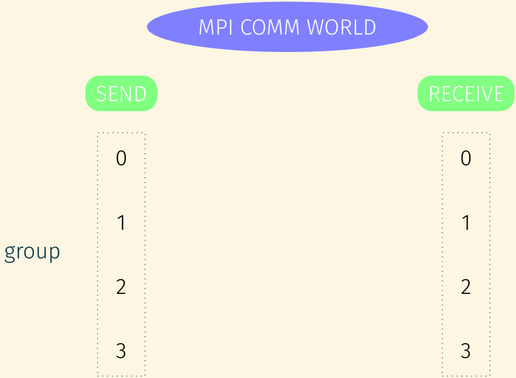
VAMP

- Parallelization over $\sum_i \alpha_i \int \frac{f(\phi_i^{-1}(x)))}{g((\phi_i^{-1}(x)))} \phi_i^{-1\prime}(x)\,\mathrm{d}G(x)$
- Number of channels process-dependent
- Workflow
  1. Broadcast all grids to all workers
  2. Compute and adapt assigned channels
  3. Master collects all grids and results → conserves numerical properties

Message-Passing Interface

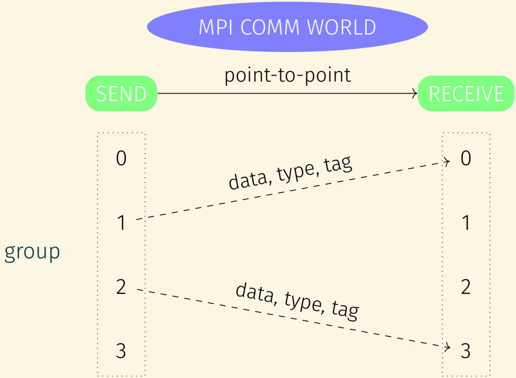
- MPI is a message-passing library interface specification
    - MPI is not a programming language
    - MPI is not an implementation
    - MPI defines only the interface for the communication procedures
    - MPI usage is more abstract and more complicate than OpenMP
- MPI-1 in 1994, MPI-2 in 1997, MPI-3 in 2012 and MPI-3.1 in 2015
- MPI-3.1 supports Fortran 2008 language bindings and *non-blocking* collective communications

Communication Basics



Communicators

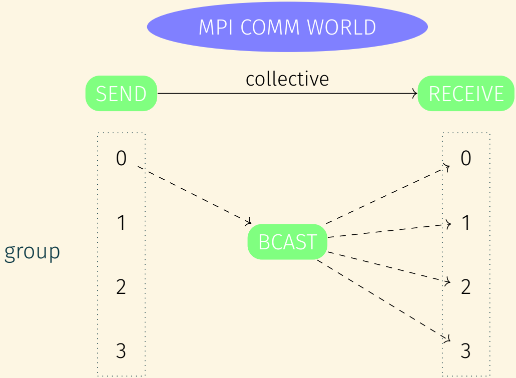
- Ordered set of process identifiers
- Tagged with an integer rank
- Communicator combines group and contex
- Predefined: `MPI_COMM_WORLD`



Point-to-Point Communication

- Sender and receiver
- Data + data type + tag = message
- `MPI_SEND`, `MPI_RECV`, …

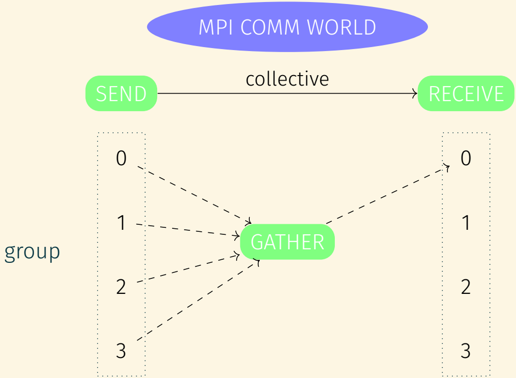
Communication Basics



Collective Communication

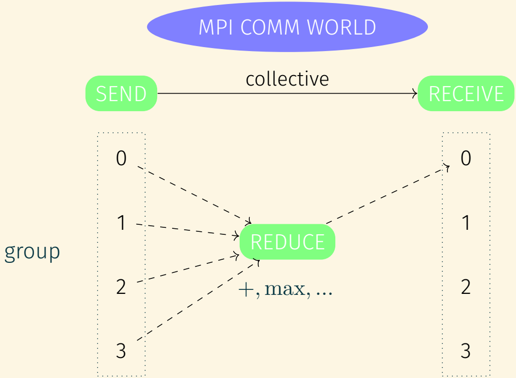
- One-to-all, all-to-one, all-to-all
- Data + data type
- `MPI_BCAST`, `MPI_GATHER`, `MPI_REDUCE`

Communication Basics



Collective Communication

- One-to-all, all-to-one, all-to-all
- Data + data type
- `MPI_BCAST`, `MPI_GATHER`, `MPI_REDUCE`



Collective Communication

- One-to-all, all-to-one, all-to-all
- Data + data type
- `MPI_BCAST`, `MPI_GATHER`, `MPI_REDUCE`

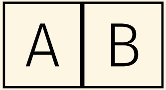
Blocking vs. Non-blocking Communication

Blocking program flow halts until communication completes

Non-blockiung directly return to calling function, communication in background

Increased parallel portion

by usage of non-blocking communication



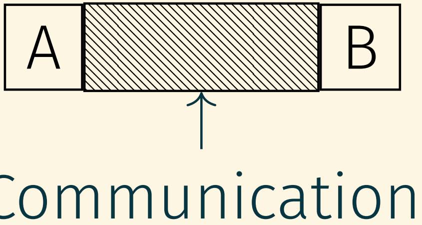
Blocking vs. Non-blocking Communication

Blocking program flow halts until communication completes

Non-blockiung directly return to calling function, communication in background

Increased parallel portion

by usage of non-blocking communication



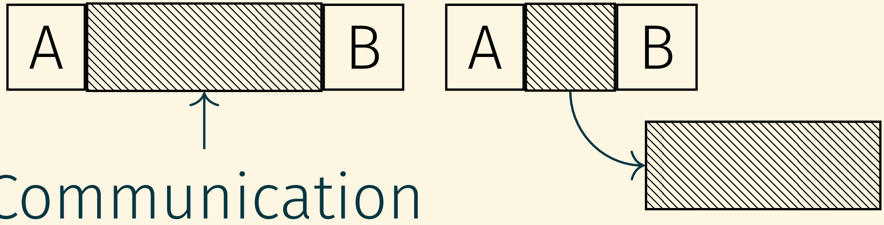
Blocking vs. Non-blocking Communication

Blocking program flow halts until communication completes

Non-blockiung directly return to calling function, communication in background

Increased parallel portion

by usage of non-blocking communication



Details of Implementation

- Reimplementation of VAMP in `Fortran 2008` → VAMP2
    - Fully-functional VAMP replacement, supports channel equivalences
- `Master/Slave` ansatz
    - Only `master` has `I/O`
    - Preserves numerical properties
- Prefer VEGAS over *multi-channel* parallelization
    - Number of *stratas* $N_{\text{boxes}} \geq 2^d \approx \mathcal{O}(100)$ to $\mathcal{O}(1000)$ or higher for stratified sampling
    - Number of *channels* $N_c \approx \mathcal{O}(10)$ to $\mathcal{O}(100)$ depending on process
- Automatically switch between parallelization modes
- Fix *numerical properties* of serial and parallel runs with `RNGstream` [L'ecuyer et al. 2002]
    - Independent random number stream for each channel/strata
- VEGAS parallelization over $d_{||} = \lfloor d/2 \rfloor$ with subspace $d_{\perp} = \lceil d/2 \rceil$ [Kreckel 1997]
- *Collective* and non-blocking communication: MPI_BCAST, MPI_REDUCE

Using WHIZARD with MPI i

- Prerequisite:
  - MPI library: *OpenMPI* (≥ `2.0.0`), MPICH → ask your administrator
  - `./configure FC=mpifort --enable-fc-mpi`
  - Prepare a hostfile → not required on a cluster

```
mpirun --hostfile myhosts -np 20 --output-filename mpi.log whizard
↪  process.sin
```

Using WHIZARD with MPI ii

Hostfile

```
$ cat myhosts
caesium slots=12
xenon   slots=12
neon    slots=5
```

Sindarin

```
$integration_method = "vamp2"
$rng_method = "rng_stream"
```

Ahmdal's Law

- Parallel portion p of a programm
- Speedup for N workers

$$S(p; N) = \frac{T(p; N)}{T(0; 1)} = \frac{1}{(1-p) + p/N}$$

- Parallelization efficiency depends heavily on parallel portion

$$\lim_{N \to \infty} S(p; N) = \frac{1}{1-p}$$

Results

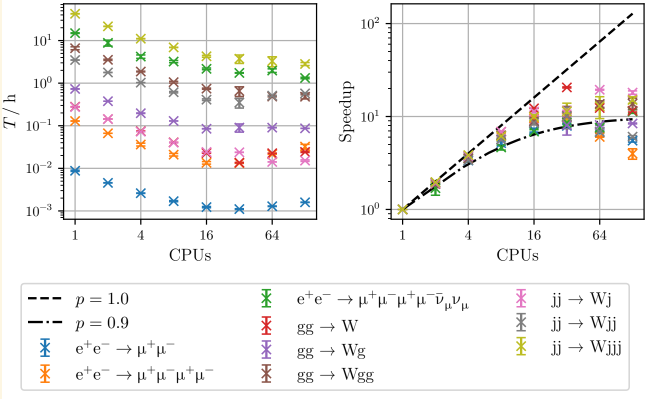


Figure 3: Benchmark for different complex processes. For reference Ahmdal's law is shown.

Discussion

- Speedup of $\mathcal{O}(10)$
  - not optimized for cluster topology
- Saturation after $N = 32$
- Parallel portion 90 %

Flavour Benchmark

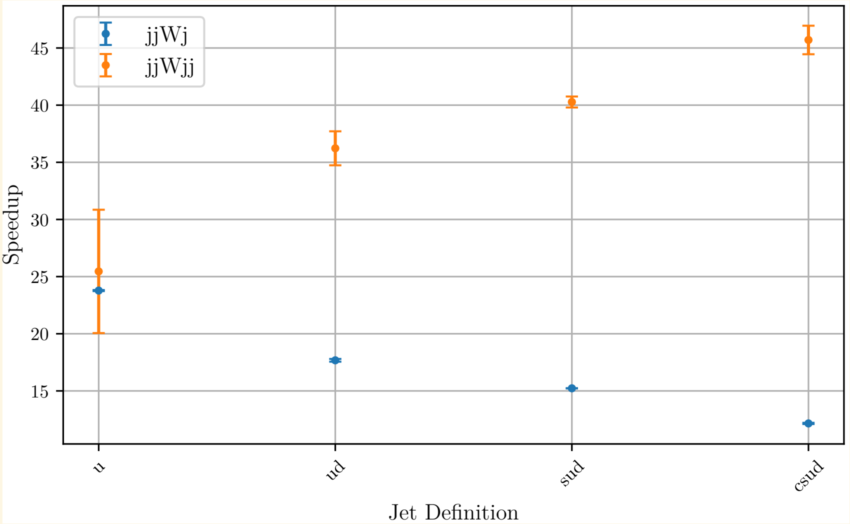


Figure 4: Benchmark for increasing jet flavours → increasing ME evaluation time.
Optimized on cluster topology with 60 workers → $45\times$ **faster**.

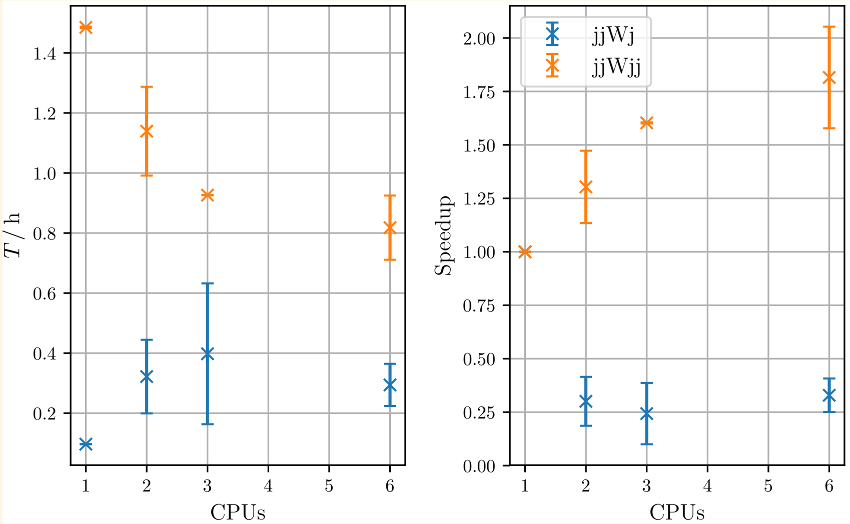


Figure 5: 20 workers with increasing number of threads → doubled speedup, but only *favorable* for **more complicated** processes.

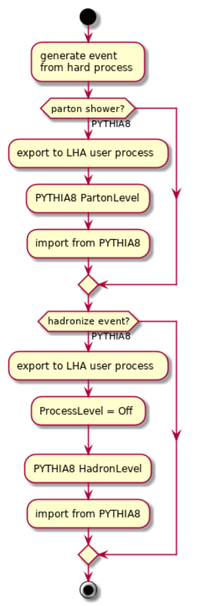
(Upcoming) Pythia8 Interface

Parton shower and hadronization

- Parton shower and hadronization
  - Analytic shower [Wolfgang Kilian, Jurgen Reuter, et al. 2012], PYTHIA6, PYTHIA8
- WHIZARD shipped with final PYTHIA6 release, version v6.427
  - Support due to backwards-compatibility to WHIZARD 1.97
  - Tightly integrated within WHIZARD
  - Export single event record export to PYTHIA6 by HEP *common block*
  - Direct event handling → no event file detour
- PYTHIA8 [Torbjorn Sjöstrand, Mrenna, and Skands 2008; Torbjörn Sjöstrand et al. 2015] successor of PYTHIA6, complete rewrite in C++
  - Improved and expanded physic models
  - Own detailed τ-decay handling [Ilten 2014; Ilten 2013] → indepdendent of TAUOLA

Event transformation

- Retrieve event record
  - Generate weighted or unweighted event from hard process
  - Read from event file → HepMC, LHEF, LCIO, StdHEP,...
- Apply event transformation
  - ISR handler
  - Parton Shower → Matching (CKKW, MLM, POWHEG)
  - Hadronization
- Write event record to file



WHIZARD and PYTHIA8

- PYTHIA8 supports external user processes by the `Les Houches Accord` [Boos et al. 2001]
    - Implement the base class `LHAup` for `WHIZARD` → `C++` to `Fortran` binding ✓
- Export random number generator to PYTHIA8
    - Implement the base class `RndmEngine` ✓
    - Stochastic independence → parallelization runs with `RNGstream`
- Implementation of a `Fortran` interface to PYTHIA8 ✓ (partially)
    - Porting of the PYTHIA6 settings
- Workflow → no need for *inbetween* event file
    - Export event to LHA user process
    - Shower with PYTHIA8
    - Import event back to WHIZARD
- (Upcoming) validation with Drell-Yan

Steering PYTHIA8

- PYTHIA8 configuration file
- PYTHIA8 configuration string → several options must be separeted by ;
- WHIZARD configurations are preferred → only the last setting count

```
$shower_method = "PYTHIA8"
$ps_PYTHIA8_STRING = "[option1]; [option2]"
! WHIZARD splits the string along the ";"
$ps_PYTHIA8_FILE = "pythia8.cfg"
```

PYTHIA8 Shower and Hadronization

- Separate parton shower and hadronization
- `PYTHIA8` allows to separate parton shower and hadronization
  - Deactivate hadronization for parton shower `HadronLevel = off`
  - Deactivate hard process for hadronization
    - `ProcessLevel = off` skips also `PartonLevel` → no beam information available
- `SINDARIN` option to activate hadronization

```
$hadronization_method = "PYTHIA8"
```

Conclusion i

Improving runtime

- Improved phasespace configuration with `cascades2` → $215\times$ faster
- Fully-validated `VAMP` reimplementation → `VAMP2` with `RNGStream`
- Message-passing based parallelization in `VAMP2` → $40\times$ faster
- Hybrid-parallelization `OpenMP` + `MPI` → doubling the speedup
- First publication using `VAMP2` and `MPI` parallelization [Ballestrero et al. 2018]

Conclusion ii

PYTHIA8

- Les Houches interface for user process
- Parton shower and hadronization → TAUOLA independent
- Steerable by PYTHIA configuration file
- Available in next WHIZARD release

Outlook

- VAMP2 and MPI are ready for producation runs!
- *Upcoming* PYTHIA8 validation and first production runs.
    - We need input from experimental side!

The End

Thank you for your attention!

Adaptive Integration

- Adpative and iterative **VEGAS** algorithm based on Importance and Stratified sampling
- Normalized step-function $p(x)$ as probability density function $g(x)$

$$g(x) = \frac{1}{N\Delta x_i}, \quad x_i - \Delta x_i \leq x_i < x_{i+1}, \quad i \in \{1, ..., N\}$$

- Steps x_i define bin length $\Delta x_i = x_i - x_{i-1}$ with $0 = x_0 < ... < x_N = 1$
- Integrate and collect (squared) weights
- Adapt bin width with weights

Implementation

Stratification Grid

- Additioanl stratifiaction grid
- Divide unit hypercube in *equal-sized* subcubes
- Access stratification grid with (box-)coordinates $\vec{c} = \{c_1, c_2, ..., c_N\}, \quad c_i \in \mathbb{N}$
- Switch between importance and (genuine) stratified sampling

Adaptive Algorithm

- Access bins with (bin-)coordinates $\vec{b} = \{b_1, b_2, ..., b_M\}, \quad b_i \in \mathbb{N}$
- Calculate x_i from bin and box coordinates
- Transform box coordinates $\vec{c}$ and a random number $\vec{r} \in U^{\otimes d}$ to actual position on the adaptive grid $\vec{x}$

$$\vec{x} = \phi\left(\vec{c} + \vec{r}\right)$$

Example

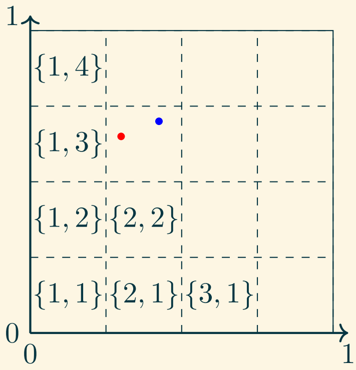


Figure 6: Stratified grid with two sample points $\{2, 3\}$.

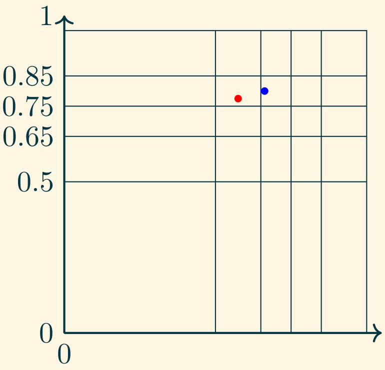


Figure 7: Adpative grid with five bins and transformed points from the stratified grid.

Parallelization over the Stratified Grid

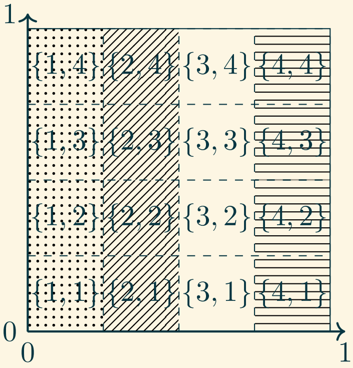


Figure 8: Stratified grid is divided for parallelization.

- Broadcast adaptive Grid non-blocking to all worker
- Each worker gets a division of the strtatifed grid assigned $\{c_1, ..., c_{D_\parallel}\}$ and $D_\parallel = \lfloor D/2 \rfloor$
- Each worker computes its division with its own (independent) stream of random numbers
- Master reduces non-blocking the results from all workers

MPI example

Point-to-Point Communication

```
if (rank == sender) then
   call MPI_send (data(:n), n, MPI_INTEGER, receiver, tag, comm, ierr)
else if (rank == receiver) then
   call MPI_recv (data(:n), n, MPI_INTEGER, sender, tag, comm, ierr)
end if
```

List of Publications

General WHIZARD reference EPJ C71 (2011) 1742, arXiv:0708.4241

O'Mega (ME generator) LC-TOOL (2001) 040; arXiv:hep-ph/0102195

VAMP (MC integrator) CPC 120 (1999) 13; arXiv:hep-ph/9806432

CIRCE (beamstrahlung) CPC 101 (1997) 269; arXiv:hep-ph/9607454

Parton shower JHEP 1204 (2012) 013; arXiv:1112.1039

Color flow formalism JHEP 1210 (2012) 022; arXiv:1206.3700

NLO capabilities JHEP 1612 (2016) 075; arXiv: 1609.03390

Parallelization of MEs CPC 196 (2015) 58; arXiv:1411.3834

POWHEG matching EPS-HEP (2015) 317; arXiv: 1510.02739

References

- Actis, S. et al. (2013). "Recursive generation of one-loop amplitudes in the Standard Model". In: *JHEP* 04, p. 037. DOI: 10.1007/JHEP04(2013)037. arXiv: 1211.6316 [hep-ph].
- Ballestrero, Alessandro et al. (2018). "Precise predictions for same-sign W-boson scattering at the LHC". In: arXiv: 1803.07943 [hep-ph].
- Boos, E. et al. (2001). "Generic user process interface for event generators". In: *Physics at TeV colliders. Proceedings, Euro Summer School, Les Houches, France, May 21-June 1, 2001*. arXiv: hep-ph/0109068 [hep-ph]. URL: http://lss.fnal.gov/archive/preprint/fermilab-conf-01-496-t.shtml.

References ii

- Cascioli, Fabio, Philipp Maierhofer, and Stefano Pozzorini (2012). “Scattering Amplitudes with Open Loops”. In: *Phys. Rev. Lett.* 108, p. 111601. DOI: `10.1103/PhysRevLett.108.111601`. arXiv: `1111.5206 [hep-ph]`.
- Denner, Ansgar, Jean-Nicolas Lang, and Sandro Uccirati (2018). “Recola2: REcursive Computation of One-Loop Amplitudes 2”. In: *Comput. Phys. Commun.* 224, pp. 346–361. DOI: `10.1016/j.cpc.2017.11.013`. arXiv: `1711.07388 [hep-ph]`.
- Frixione, S., Z. Kunszt, and A. Signer (1996). “Three jet cross-sections to next-to-leading order”. In: *Nucl. Phys.* B467, pp. 399–442. DOI: `10.1016/0550-3213(96)00110-1`. arXiv: `hep-ph/9512328 [hep-ph]`.

- Ilten, Philip (2013). “Electroweak and Higgs Measurements Using Tau Final States with the LHCb Detector”. PhD thesis. University Coll., Dublin. arXiv: 1401.4902 [hep-ex]. URL: https://inspirehep.net/record/1278195/files/arXiv:1401.4902.pdf.
- – (2014). “Tau Decays in Pythia 8”. In: *Nucl. Phys. Proc. Suppl.* 253-255, pp. 77–80. DOI: 10.1016/j.nuclphysbps.2014.09.019. arXiv: 1211.6730 [hep-ph].
- Ježo, Tomáš and Paolo Nason (2015). “On the Treatment of Resonances in Next-to-Leading Order Calculations Matched to a Parton Shower”. In: *JHEP* 12, p. 065. DOI: 10.1007/JHEP12(2015)065. arXiv: 1509.09071 [hep-ph].
- Kilian, W. (2001). “WHIZARD manual”. In: pp. 1924–1980.

Kilian, Wolfgang, Thorsten Ohl, and Jürgen Reuter (Sept. 2011). "WHIZARD—simulating multi-particle processes at LHC and ILC". In: *The European Physical Journal C* 71.9. DOI: `10.1140/epjc/s10052-011-1742-y`. URL: `https://doi.org/10.1140%2Fepjc%2Fs10052-011-1742-y`.

Kilian, Wolfgang, Jurgen Reuter, et al. (2012). "An Analytic Initial-State Parton Shower". In: *JHEP* 04, p. 013. DOI: `10.1007/JHEP04(2012)013`. arXiv: `1112.1039 [hep-ph]`.

Kreckel, Richard (Nov. 1997). "Parallelization of adaptive MC integrators". In: *Computer Physics Communications* 106.3, pp. 258–266. DOI: `10.1016/s0010-4655(97)00099-4`. URL: `https://doi.org/10.1016%2Fs0010-4655%2897%2900099-4`.

L'ecuyer, Pierre et al. (2002). "An object-oriented random-number package with many long streams and substreams". In: *Operations research* 50.6, pp. 1073–1075.

- Lepage, G. Peter (1978). "A New Algorithm for Adaptive Multidimensional Integration". In: *J. Comput. Phys.* 27, p. 192. DOI: 10.1016/0021-9991(78)90004-9.
- – (1980). "VEGAS: AN ADAPTIVE MULTIDIMENSIONAL INTEGRATION PROGRAM". In:
- Moretti, Mauro, Thorsten Ohl, and Jurgen Reuter (2001). "O'Mega: An Optimizing matrix element generator". In: pp. 1981–2009. arXiv: hep-ph/0102195 [hep-ph].
- Nejad, Bijan Chokoufe, Thorsten Ohl, and Jürgen Reuter (Nov. 2015). "Simple, parallel virtual machines for extreme computations". In: *Computer Physics Communications* 196, pp. 58–69. DOI: 10.1016/j.cpc.2015.05.015. URL: https://doi.org/10.1016%2Fj.cpc.2015.05.015.
- Ohl, Thorsten (1997). "CIRCE version 1.0: Beam spectra for simulating linear collider physics". In: *Comput. Phys. Commun.* 101, pp. 269–288. DOI: 10.1016/S0010-4655(96)00167-1. arXiv: hep-ph/9607454 [hep-ph].

- Sjöstrand, Torbjorn, Stephen Mrenna, and Peter Z. Skands (2008). "A Brief Introduction to PYTHIA 8.1". In: *Comput. Phys. Commun.* 178, pp. 852–867. DOI: `10.1016/j.cpc.2008.01.036`. arXiv: `0710.3820 [hep-ph]`.
- Sjöstrand, Torbjörn et al. (2015). "An Introduction to PYTHIA 8.2". In: *Comput. Phys. Commun.* 191, pp. 159–177. DOI: `10.1016/j.cpc.2015.01.024`. arXiv: `1410.3012 [hep-ph]`.
- Soden-Fraunhofen, Johann Felix von (2015). "GoSam 2.0: a tool for automated one-loop calculations". In: *J. Phys. Conf. Ser.* 608.1, p. 012061. DOI: `10.1088/1742-6596/608/1/012061`. arXiv: `1409.8526 [hep-ph]`.
- Utsch, Manuel (Apr. 2018). "Parametrisierung des Phasenraums im Monte-Carlo-Eventgenerator WHIZARD". MA thesis. Unversität Siegen.