GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Institut für Theoretische Physik

Development of novel, portable matrix-element + phase space methods

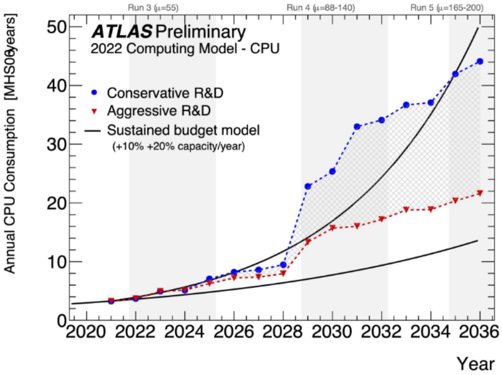
Max Knobbe
MCnet meeting 2023

In collaboration with:
E. Bothmann, T. Childers, W. Giele, S. Höche, J. Isaacson, R. Wang

MCnet

Gefördert durch
DFG Deutsche Forschungsgemeinschaft

Introduction / Motivation



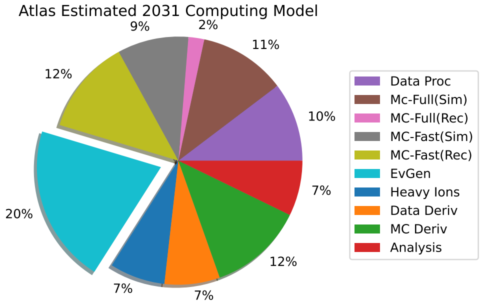


Figure and numbers taken from [CERN-LHCC-2022-005]

- Computing needs are predicted to grow faster than available resources
  $\rightarrow$ Computing budget might limit physics outcome
- Sizeable part of CPU budget spend on event generation (roughly 20%)
- Our strategy, dedicated rewrite to tackle major bottle-necks
  $\rightarrow$ Identify major bottle-necks first!

Boiling down the Problem

Expensive MC Samples: V+Jets with many jets

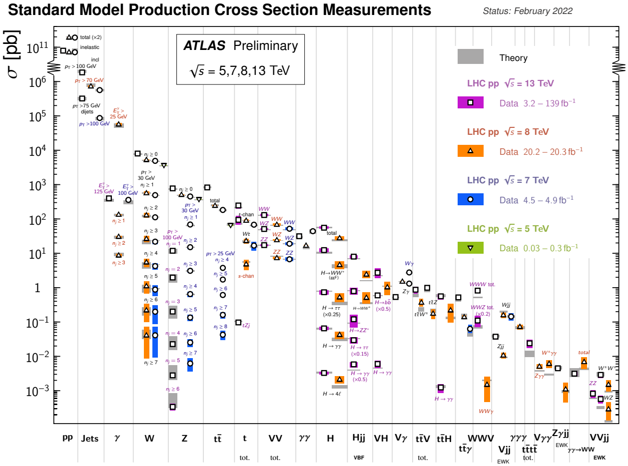
- Background to essential analysis(es), e.g. Higgs-boson and top-quark measurements

  cf. [2112.09588]

- Large production cross-section → large MC samples
- To reduce significant portion of MC budget, ensure to be efficient for these processes

Guiding Principles for the following discussion:

1. Good performance for bottleneck processes (for now: V+Jets,ttbar), in particular for many jets!
2. Deployable in modern architectures
3. Useful integration in existing MC tool-chain



twiki.cern.ch/twiki/bin/view/AtlasPublic/StandardModelPublicResults

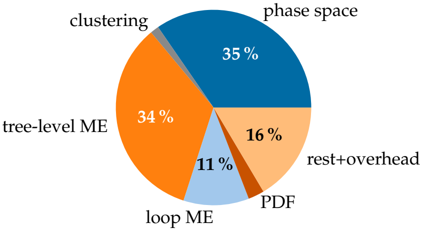
(Some) Components of a MC Computation

$$\sigma_{pp\to X_n} = \sum_{ab}\int dx_a dx_b d\phi_n f_a(x_a,\mu_F^2) f_b(x_b,\mu_F^2) \times |\mathcal{M}_{ab\to X_n}|^2 \Theta(p_1,...,p_n)$$

- After some optimisation: Large portion of MC time spend in ME + PS
  cf. [2209.00843]
- In this talk: Re-think ME stratgy
- Goal: Develop efficient strategy for the different components
- Naive treatment of helicity/color sum scales terribly with increase of multiplicity
- $$|\mathcal{M}_{ab\to X_n}(1,...,n)|^2 = \sum_{\text{helicity}}\sum_{\text{color}} A(p_1,...,p_n)A(p_1,...,p_n)^\dagger$$

  Major components we have to take care of are
  1. The Phase-Space generation
  2. The helicity sum
  3. The amplitudes
  4. The color sum

$pp \to e^+e^-$+0,1,2j@NLO+3,4,5j@LO

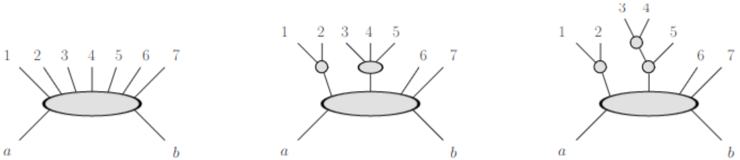


[2209.00843]

A new Phase-Space generator: Chili [Bothmann et al. 2302.10449]

Basic building blocks:

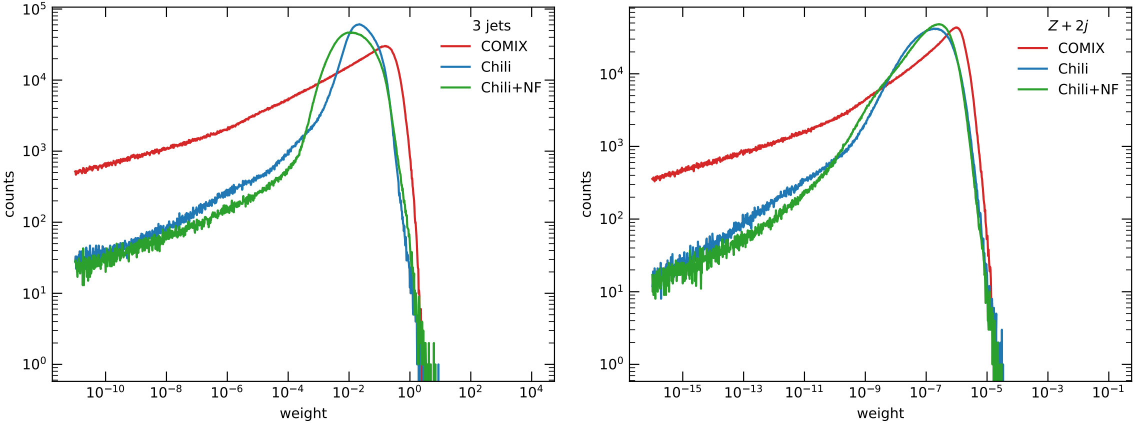
- T-channel phase-space generator from MCFM [Figy, Giele 1806.09678]
- S-channel phase-space factorization and decays [Byckling, Kajantie NPB9(1969)568]
- Variable number of s-channel decays (user-defined) to eliminate exponential scaling



(a) (b) (c)

- Very simple phase space generator, but fast and portable

Chili performance compared to Sherpa defaul



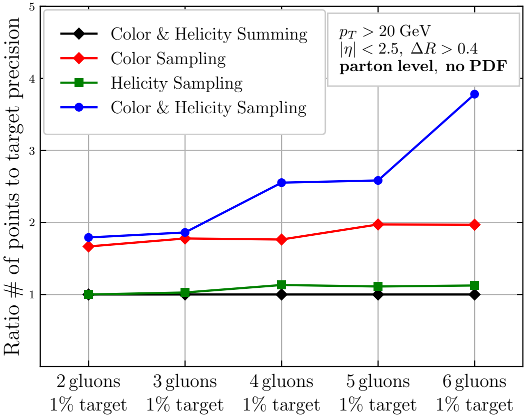
→ Tested across a large range of processes ($W + 5j,\ Z + 5j,\ h + 5j,\ t\bar{t} + 4j,\ \gamma + 5j,\ 6j$)
→ Overall good performance

Investigate impact of helicity-summing vs. sampling on convergence

$\rightarrow$ How many points to we have to compute to make up for MC estimate?

- Compare convergence in realistic setup
  $\rightarrow$ Recursive PS-Sampler, ...
- Helicity sampling comes with close to zero additional points
  $\rightarrow$ This takes analytic amplitude knowledge into account
- Loss in precision increases with multiplicity for color and helicity sampling

$\Rightarrow$ **Algorithmic choice: Sample helicities**



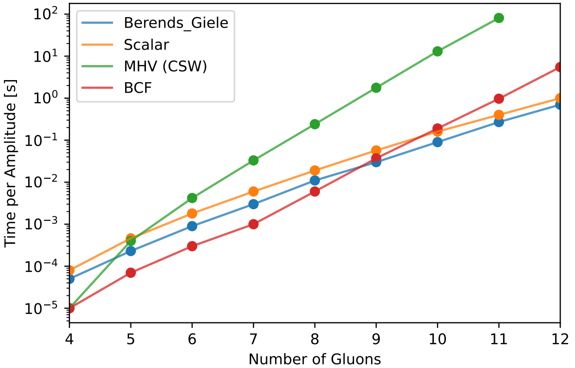
[Bothmann, Giele, Höche, Isaacson, MK, 2106.06507]

The Amplitudes

- Different strategies to compute tree-level amplitudes efficiently
  1. Berends-Giele like recursion
  2. Scalar
  3. MHV (CSW)
  4. BCF
- Rely on performance studies from early 2000's
  hep-ph/0602204,hep-ph/0607057
- We are interested in best scaling behaviour / performance for multi-jet processes (guiding principles)

Best scaling option is the Berends-Giele recursion
⇒ **Algorithmic choice: Berends-Giele recursion**

NB: BCF-like recursions show potential for intermediate multiplicites, dominant effect from (N)MHV amplitudes that can be included in current generator

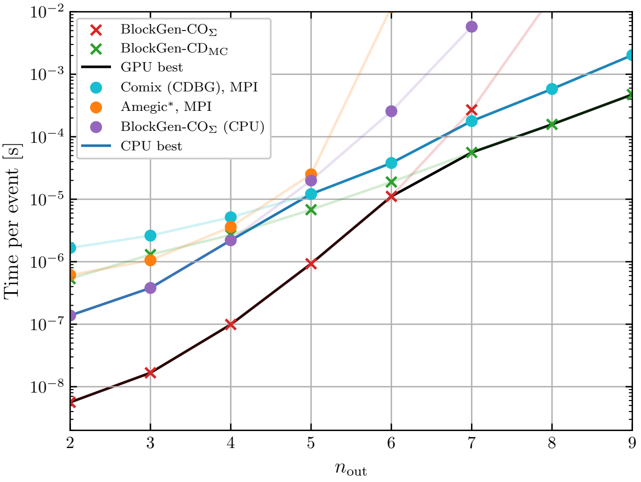


Based on numbers from hep-ph/0602204

The Color Sum [Bothmann, Giele, Höche, Isaacson, MK, 2106.06507]

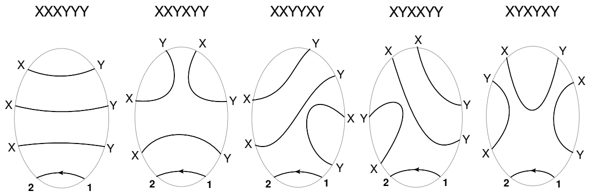
- Benchmark performance for gluon-only process
- Relevant test, since as-many-gluon-as-possible amplitudes make up largest portion of computing time for jet-processes
- Compare different color treatments: color-dressing/summing/sampling
- Color-sampled algorithms scale similar to color-summed approaches
- Color-summing scales worse than color-dressing, but faster up to roughly 5-6 outgoing jets
- Caveat: Color-sampling comes with penalty factor from slower convergence

$\Rightarrow$ **Algorithmic choice: Sum colors**



From Gluon-only to V+Jets

- Introduce spinors (Weyl for massless, Dirac for massive particles)
- Add more general QCD three point vertices
- Straight-forward for helicity-sum and Berends-Giele recursion
- First time in a code aimed for production: use minimal QCD color-basis $\{A(1,2,\sigma), \sigma \in \text{Dyck}\}$
  [T. Melia 1304.7809 & 1312.0599 & 1509.03297; H. Johansson, A. Ochirov, 1507.00332]
  $\rightarrow$ Allows to fix one fermion line, remaining permutations are given by Dyck-Words
  $\rightarrow$ For particle Dyck Words: ()(), (())
  $\rightarrow$ Significantly less amplitudes to compute
- Include EW particles after QCD basis has been setup

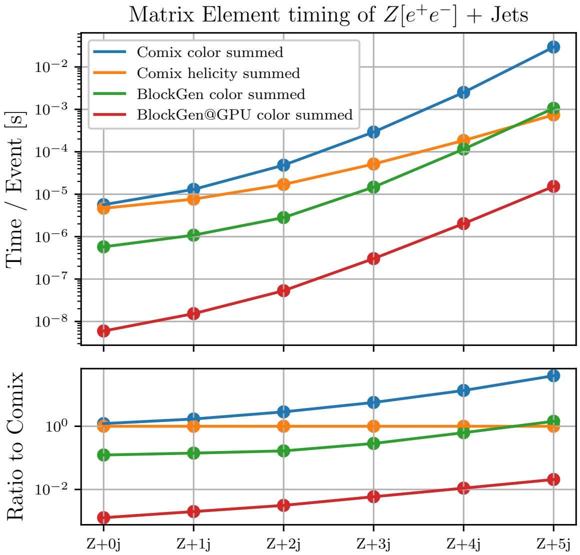


[1304.7809]

Results

Relevant benchmark process: $pp \to Z[ee] + nj$

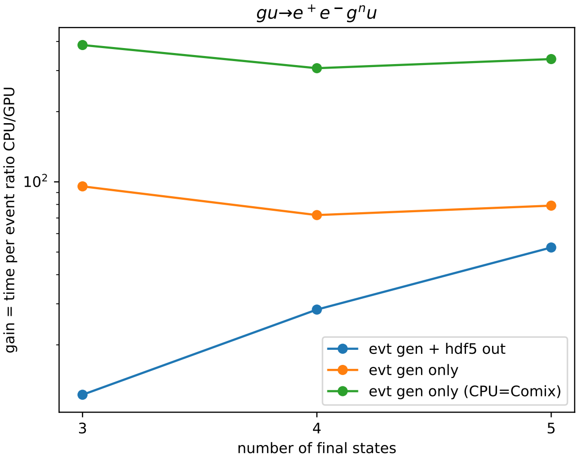
- Realistic setup for ME computations
  $\to$ Include all sub-processes, no PDF
  $\to$ Dominated by as-many-as-possible
- Compare dedicated C++ Version with dedicated Cuda version
- PS: Rambo
- CPU: i3-8300 CPU @ 3.70GHz, GPU: Tesla V100S
- Excellent performance compared to current Sherpa-Standard
- Almost three orders of magnitude for lower multiplicities (single CPU core vs full GPU)



Results cont'd: PRELIMINARY Performance of the whole tool-chain*

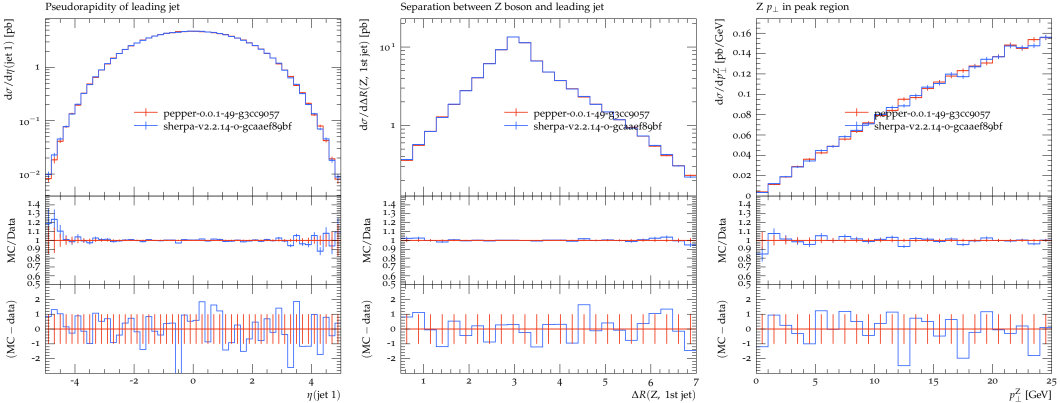
Best comparison we can currently do, but details hiding in the '*':

- Still lacking Chili@GPU
  → The only real GPU number we can compute are the ME timings
  → Remaining numbers are estimate from the CPU performance
- The gain here is measured per trial event
  → Comix' worse cut-efficiency helps
  → Eventually should measure per unweighted event
- LHAPDF is avaiable on the GPU, but not used yet
  → Will be added once PS is there
- hdf5 can probably be hidden by asynchronous writes



Results cont'd: How to make this usable

1. Compute Phase-Space and Matrix-Element
2. Write out **unweighted** Phase-Space points
3. Do the remaining MC magic the usual way



Conclusion / Outlook

- Matrix-elements show excellent performance on both CPU and GPU
  - Helps to reduce CPU time consumption but also enables more/complicated computations and use of modern GPU data-centers
- CPU Speed-up $\mathcal{O}(10)$, GPU Speed-up $\mathcal{O}(100)$
  - Color-summing/helicity sampling good choice for intermediate to high multiplicities
  - No process-specific optimisations $\rightarrow$ straight-forward extension to more processes
  - Potentially need some more fine-tuning of Cuda version
- Eventual goal: parton-level generator that delivers seed events to SHERPA for further processing
  - Proposed workflow, (ME,PS)@GPU, (Shower,Hadronisation,MPI,..,)@CPU using HDF5 I/O [Höche, Prestel, Schulz 1905.05120]
    $\rightarrow$ Nearing usable product, natural inclusion in current toolchain