SHERPA: performance and statistics

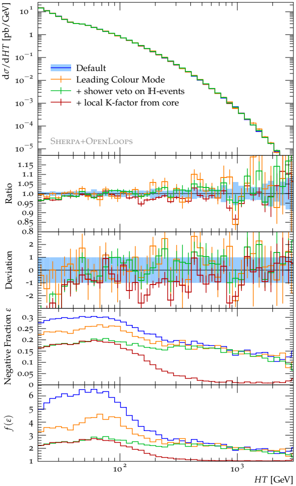
Marek Schönherr

IPPP, Durham University

Negative weight fractions

Danziger, Höche, Siegert, arXiv:2110.15211, ATLAS arXiv:2112.09588

- explored three methods to improve the neg. weight fraction in SHERPA
  1) reduce matching accuracy to leading colour, neglect spin-correlations
  2) include jet veto on $\mathbb{H}$-events, as originally formulated in arXiv:2012.5030
  3) use local K-factor in NLO→LO merging from core configuration instead of highest multiplicity
- public since SHERPA-2.2.8 (Sep '19)



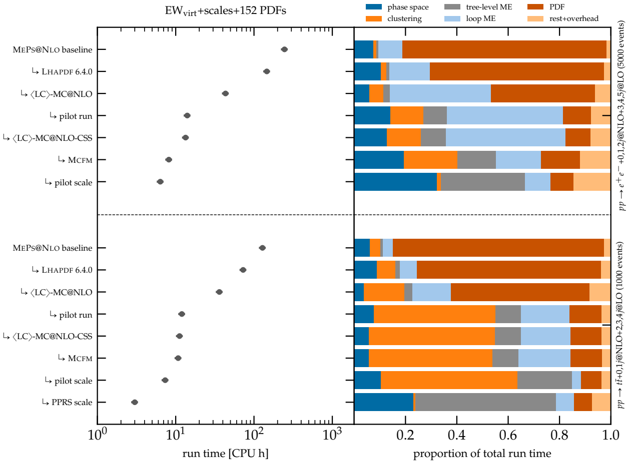
Simplified pilot runs

Bothmann, et.al., arXiv:2209.00843

explored how to reduce the CPU footprint for the heaviest use cases, e.g. $Z + 0, 1, 2j$ @NLO $+ 3, 4, 5j$ @LO (ATLAS default)

1) improvements in **LHAPDF** (internal grid handling)
2) ⟨**LC**⟩-**Mc@Nlo**, reduce S-Mc@Nlo to traditional Mc@Nlo
3) **pilot run**
   use minimal setup to find accepted phase space point,
   recompute with all bell and whistles
4) ⟨**LC**⟩-**Mc@Nlo-CSS**, move ⟨LC⟩-Mc@Nlo to shower
5) replace more versatile loop library like OpenLoops with analytical single-purpose loop library like **Mcfm**
6) compute the pilot run with a simplified **pilot scale** definition, e.g. H_{T} instead of scale defined through clustering
   → incurs a weight by reverting to correct scale in final event
   → small weight spread, no significant reduction of stat. power

Simplified pilot runs

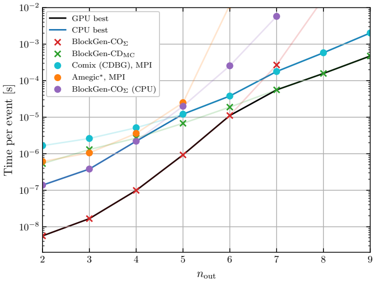


Matrix-element generators on GPUs

Bothmann, et.al., arXiv:2106.06507, to appear

- new generator BLOCKGEN (now PEPPER) to explore suitable a algorithms for GPU computations
  → process and multiplicity dependent
- write out to `Hdf5`, read-in to SHERPA proper for showering, merging, ...

$gg \to ng$



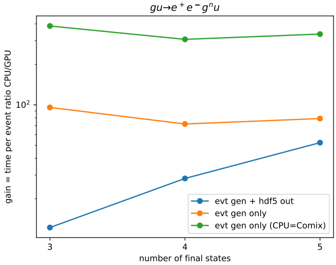
16 threaded CPUs vs. 1 GPU

Matrix-element generators on GPUs

- new generator BlockGen (now Pepper) to explore suitable a algorithms for GPU computations
  → process and multiplicity dependent
- write out to `Hdf5`, read-in to Sherpa proper for showering, merging, ...

Bothmann, et.al., arXiv:2106.06507, to appear

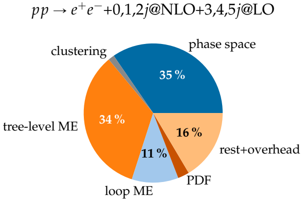
preliminary

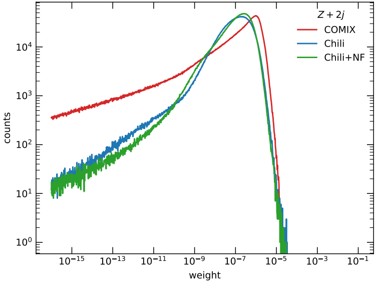


1 CPU vs. 1 GPU

Phase-space generators on GPUs

Bothmann, et.al., arXiv:2302.10449





- phase space generator important part of the story, Chili

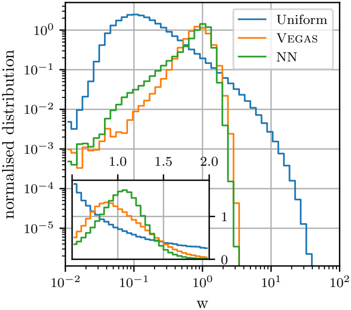
- traditional automatic phase space parametrisation contains too many channels that are not relevant for mundane inclusive phase space region used for main ATLAS/CMS event samples

Machine Learning phase-space integration

Gao, et.al., arXiv:2001.10028, Bothmann, et.al, arXiv:2001.05478

Two approaches using normalising flows

1) learn phase-space distribution of momenta directly
2) learn transformation of random numbers using existing phase space parametrisation, ie. replace VEGAS

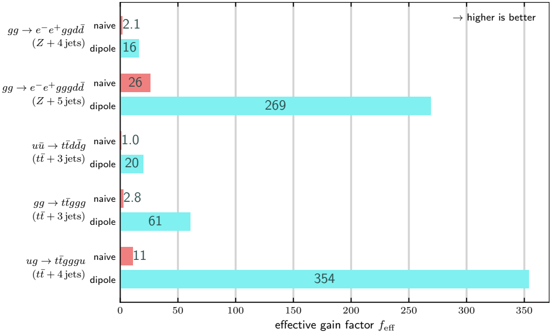


- works beautifully at low multiplicities, no better than VEGAS at higher multis

Machine Learning matrix elements

Danziger, et.al., arXiv:2109.11964, Janßen, et.al., arXiv:2301.13562

- replace ME with fast ML surrogate
- use second unweighting step to correct surrogate to full ME



Conclusions

- reverting to traditional MC@NLO (neglecting $N_c = 3$ colour- and spin-correlations in matching) severely reduces negative weights without impacting the physics description of standard observables with current uncertainties
  $\rightarrow$ available since SHERPA-2.2.8
- pilot runs for unweighted event generation massively reduces generation time per event with no change to physics description
  $\rightarrow$ available since SHERPA-2.2.12
- newly designed matrix element and phase space generators for GPUs will further substantially reduce generation time per event, necessitates intermediate storage format
  $\rightarrow$ to be introduced in SHERPA-3.x
- ML sollutions to phase space integration not yet suitable for high-multiplicities, but ME-surrogates offer working solution
  $\rightarrow$ to be introduced in SHERPA-3.x

Thank you!