Event generator benchmarking update

HSF WG meeting

May 9th, 2019

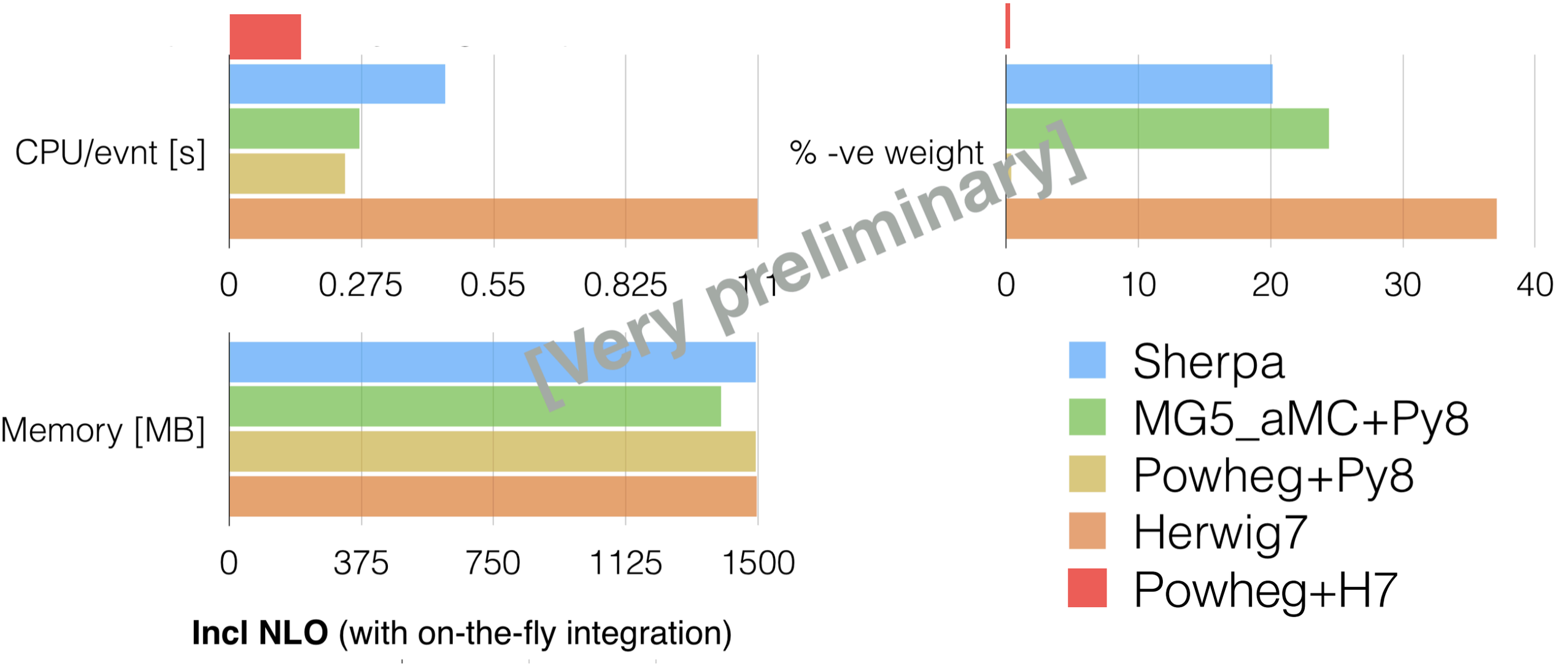
S. Amoroso, F.Siegert
For ATLAS

BENCHMARKING GENERATORS

- For the HSF workshop we prepared an ATLAS benchmarking of generator configurations across different codes
  - Monitoring both CPU and memory usage and the negative weights fraction
  - All tests run on a bare-metal machine using atlas software
  - 10 runs of 5-10K events which are then averaged
  - Still few inconsistencies in the setups used (e.g. MG5_aMC@NLO runs with internal PDFs as could not use LHAPDF)

- Aim to expose how the different physics choices made by different codes impact the resource usage and push the authors to improve
- Will go through few small updates on the numbers we showed, and the lessons learned from this exercise

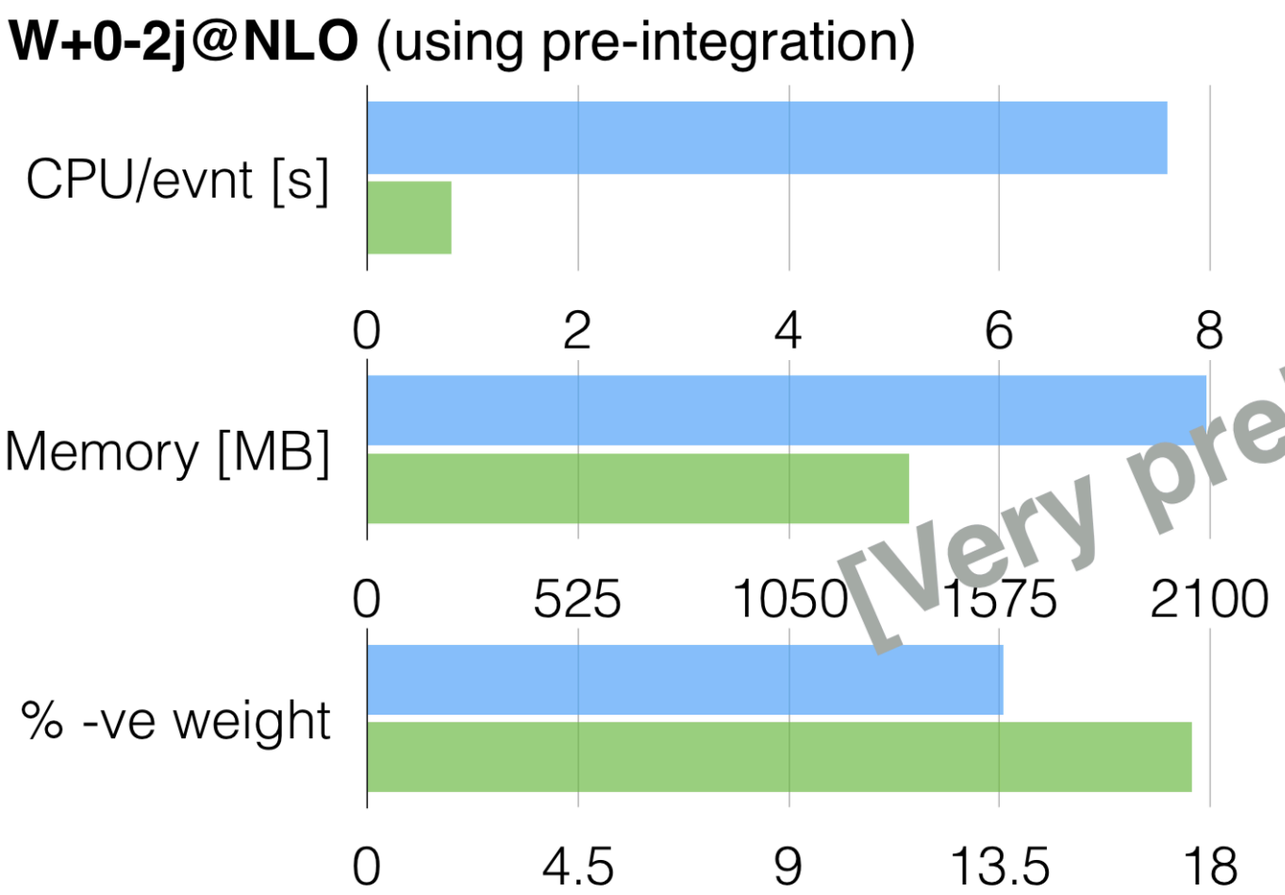
BENCHMARKING - TTBAR

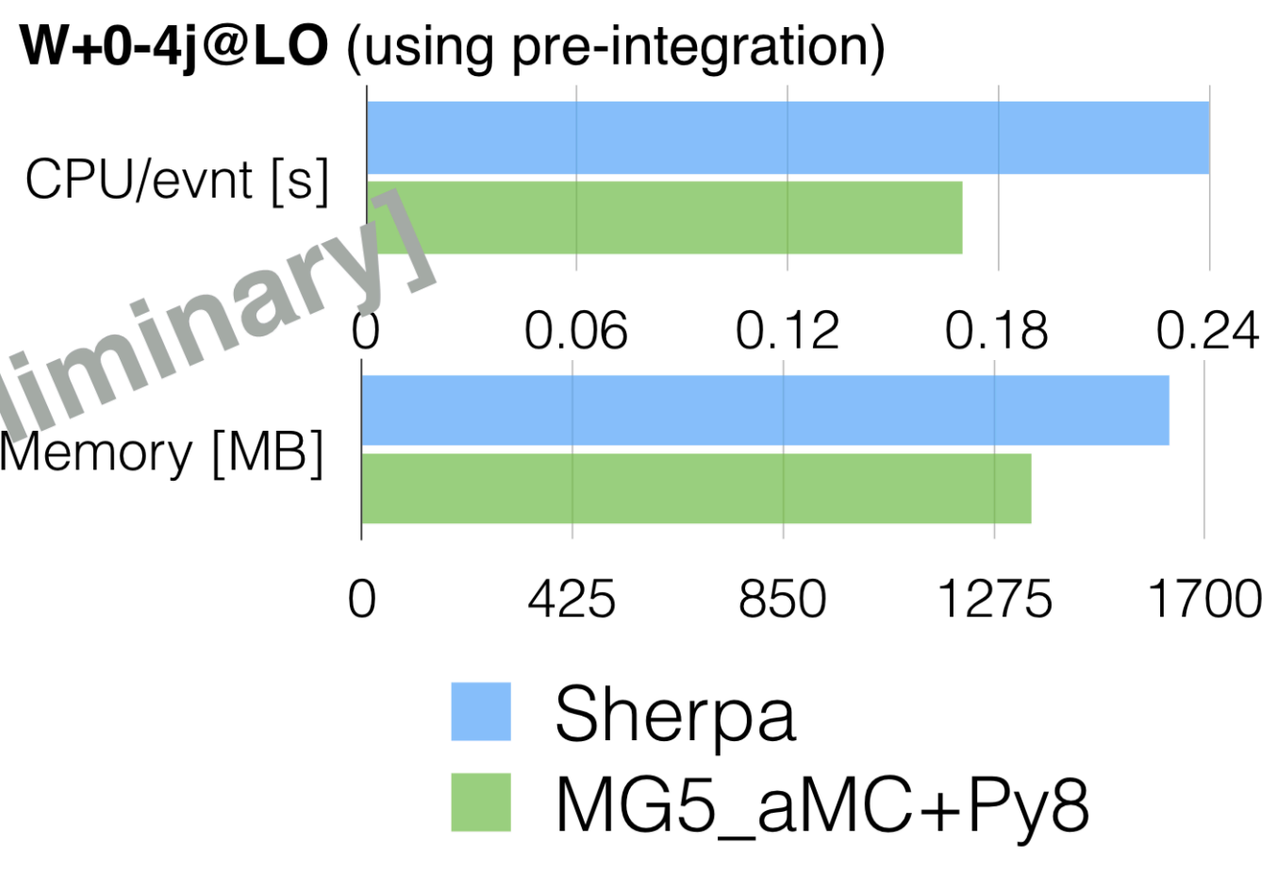


Incl NLO (with on-the-fly integration)

- First comparison is for ttbar@NLO, with all generator settings to their default values
  - Not representative of a realistic use case, as integration accuracies might differ (and are anyhow usually done in a separate step)
  - Powheg extremely fast and with almost no negative weights
  - Herwig7-Matchbox both slow and with a large neg. weight fraction

Benchmarking - W+jets

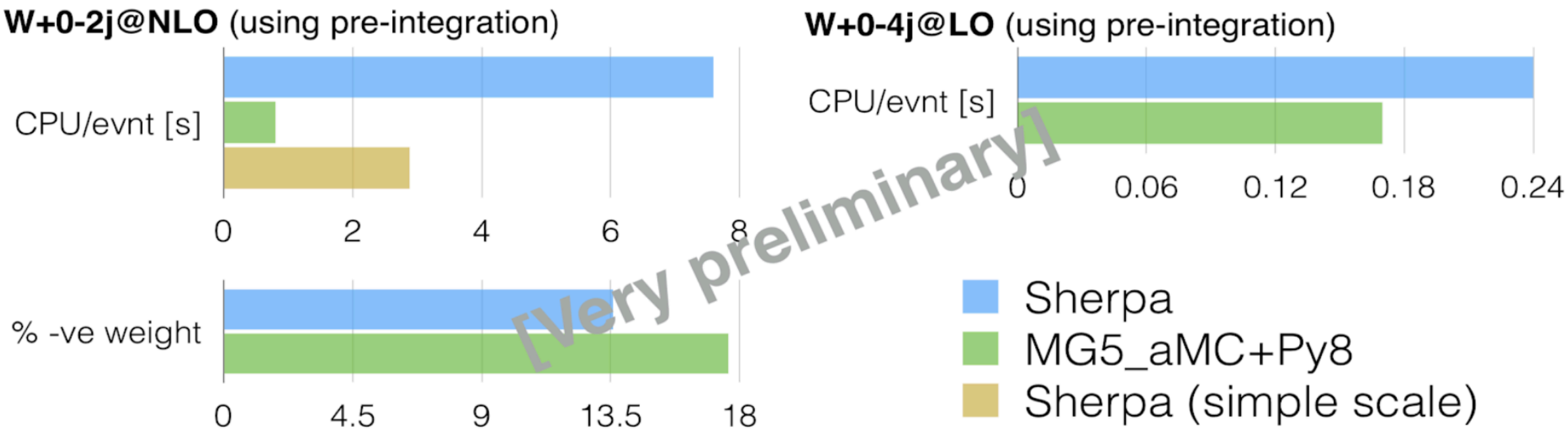




[Very preliminary]

- In NLO-merged configurations we see differences in CPU/evnt between Sherpa and FxFX for the same accuracy
  - From the discussion we had in the workshop this was found to be related to the Sherpa scale choice in W+3jets real-emission ME
  - Sherpa choice is formally more accurate (logarithmically)
  - But using the FxFx scale makes Sherpa faster by more than a factor two -> Does this has any impact on the physics?

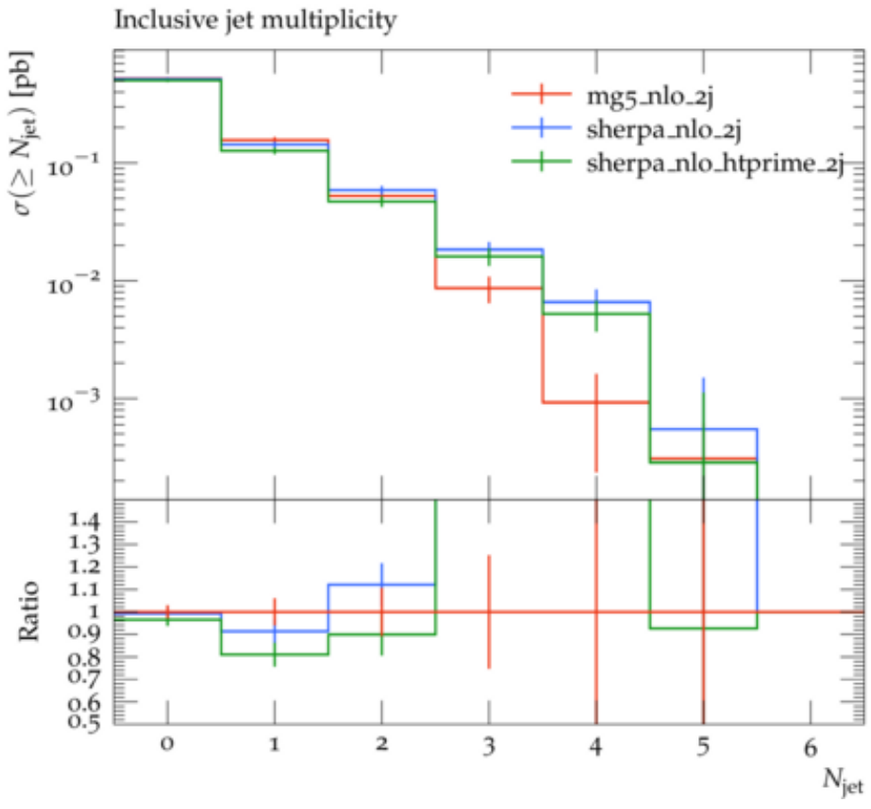
Benchmarking - W+jets

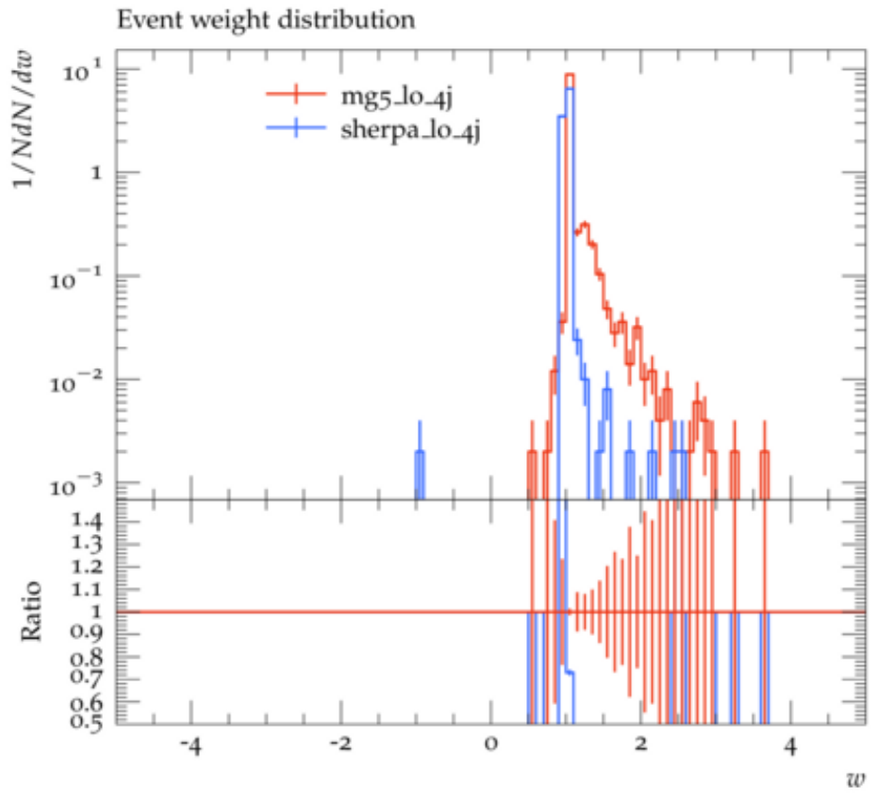


- In NLO-merged configurations we see differences in CPU/evnt between Sherpa and FxFX for the same accuracy
  - From the discussion we had in the workshop this was found to be related to the Sherpa scale choice in W+3jets real-emission ME
  - Sherpa choice is formally more accurate (logarithmically)
  - But using the FxFx scale makes Sherpa faster by more than a factor two -> Does this has any impact on the physics?

Benchmarking - W+jets

- The physics impact of the HTprime scale doesn't seem large
  - But need to check this with a larger scale validation
  - We plan to use this new scale for our future Sherpa productions
  - And I believe the Sherpa authors plan to make it the default one
- Sherpa is found to have a narrower weight distribution than FxFx
  - This is due to the merging not being unweighted in FxFx (to be followed up with the authors)





Negative weights in aMC@NLO

- NLO computations naturally introduce negative weights
  - They can hamper the statistical power of a sample
  - aMC@NLO, H7 and unitarised mergings seen to behave particularly bad
  - Mitigation strategies are possible, but require significant time investment from the generator authors

- In aMC@NLO S-events folding can reduce the negative weight fraction at the expense of increased generation time
  - Tested it using a private branch from the authors and generating samples of ttbar production at NLO

Fold (ξ, y, ϕ)	**Neg. weights**	**Time (10K evts)**
1,1,1	20 %	1 m
2,1,2	16 %	3 m
4,1,4	14 %	7 m
8,1,8	14 %	24 m

  - Reduction of the neg. weight fraction visible but not enough to compensate the CPU increase

Summary

- We looked at the differences in resource usage between different generator codes for the same setup
  - Moving to an HTprime scale in Sherpa might allow us to gain a factor two in CPU for our next big V+jet samples
  - Strategies to reduce the negative weight fraction are being implemented in aMC@NLO and might soon be available

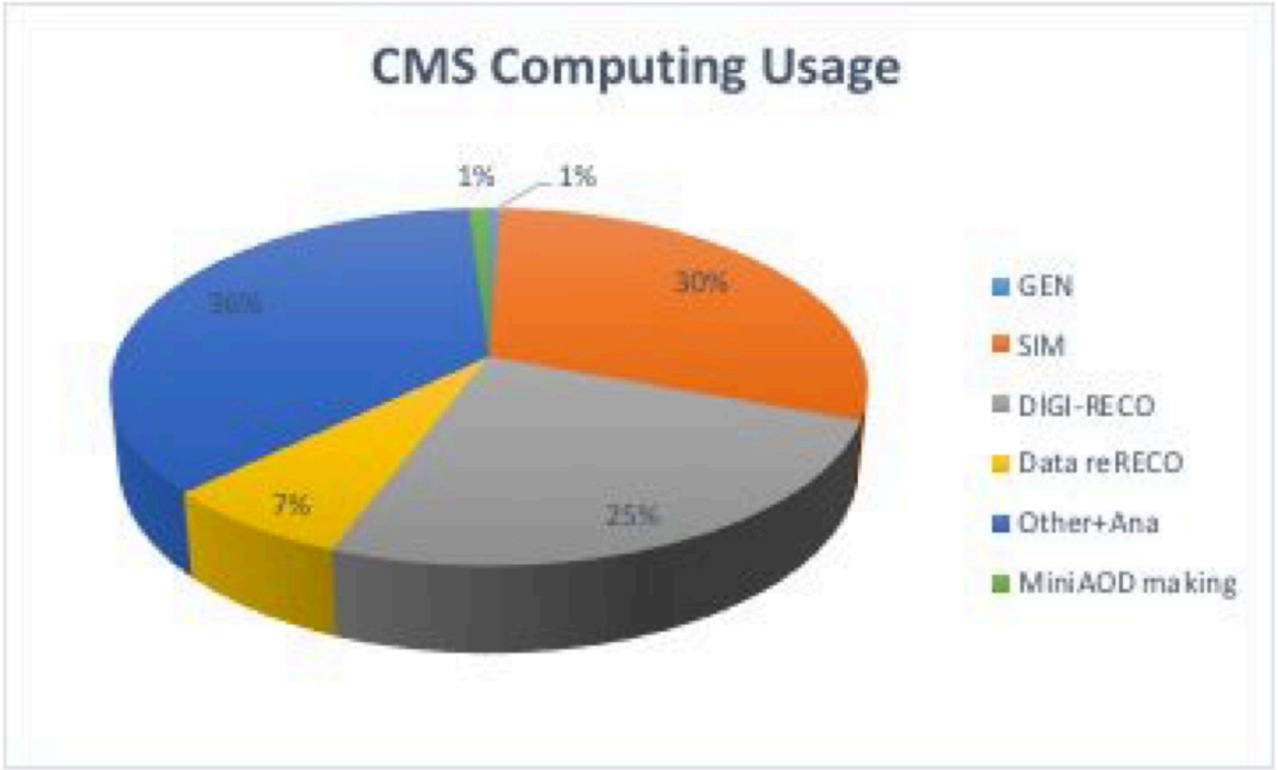
- We plan to finalise the benchmarking exercise and document it in an ATLAS public note
  - Add the at the impact of EvtGen, LHAPDF, and ME and shower weights on the overall timing
  - Include H7 to the LO/NLO-merged comparisons
  - Look at the impact of compiler optimisation flags

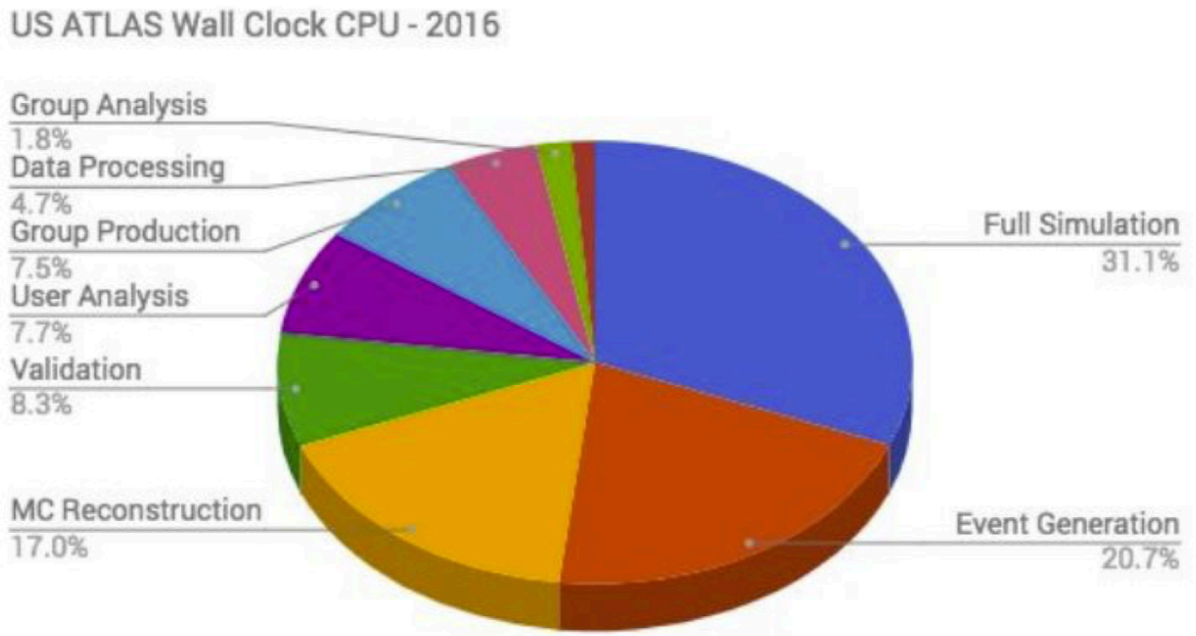
- Will allow some more (and public) discussion with the generator authors about differences in resource usage

Backup

ATLAS/CMS COMPUTING BUDGET

- Very different fraction of even CPU usage between ATLAS and CMS
  - CMS claims ~3% of the total vs ~20% for ATLAS

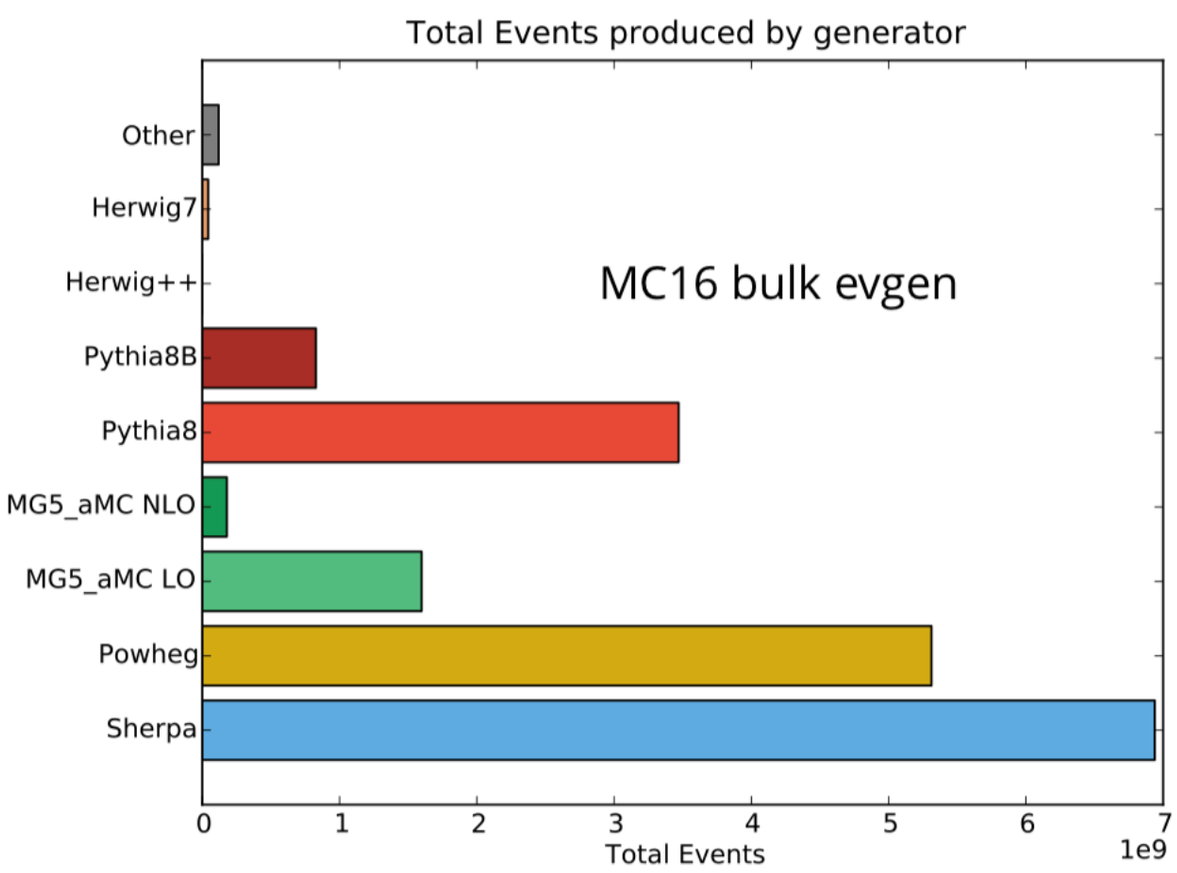


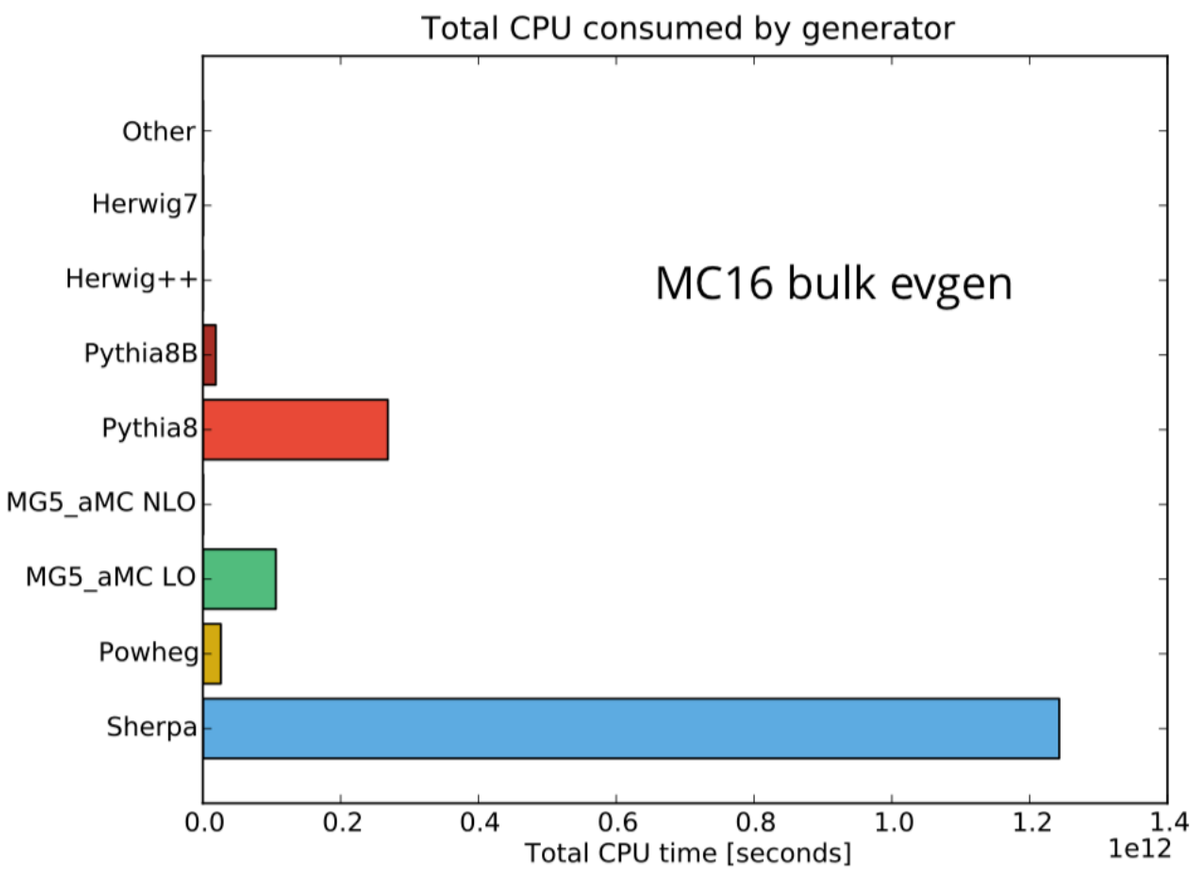


- Discussed both at the HSF workshop and in follow-up meetings
- (mostly) Tracked down to be a combination of two effects
  - ATLAS numbers are from 2016, when the Run2 evgen production was in full-swing. For 2018 evgen is 11% of total
  - CMS relying mostly on LO-merged (MLM) samples (FxFx used for V+jets, ttbar and few other bulk samples)

Current ATLAS Generator Usage

- Sherpa (NLO-merged) samples are currently taking most of our event generation CPU consumption
  - The baseline V+jets sample is the largest (3.2B) and most precise sample we produce (V+2jets@NLO, V+4jets@LO)
- Computational load grows ~factorially with the number of particles and virtual loops.



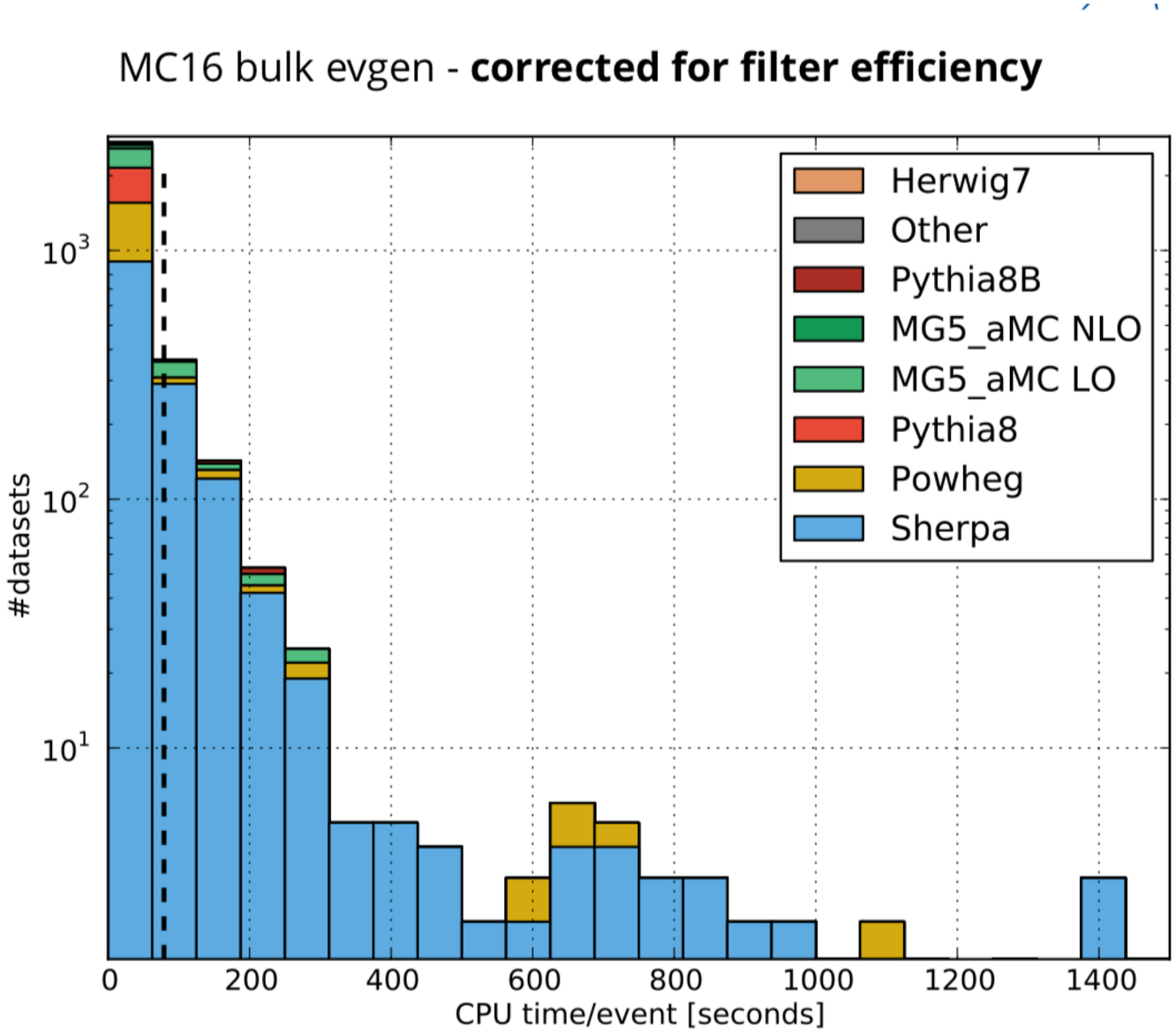


Current ATLAS Generator Usage

- The average CPU/event for all of the mc16 datasets
  - Each entry in the plot representing a single dataset

- The average across all samples is ~80s/event
- But for many samples the event generation is much slower than the full simulation

	Avg CPU/evt
Evgen	~80 s
FullSim	~245 s
FastSim	~45 s
Reco	~60 s



Sample	Fraction of events with neg. weights [%]
Sherpa (lepton+jets)	20.5
Sherpa (lepton+jets)	20.4
Sherpa (dilepton)	20.4
Sherpa ttbb (lepton+jets, CSSKIN, 4FS)	24.4
Sherpa ttbb (lepton+jets, CMMPS, 4FS)	25.7
aMC@NLO+Py8 (lepton+jets)	23.7
aMC@NLO+Py8 (dilepton)	23.7
aMC@NLO+Py8 (FxFx, 70 GeV)	28.4
aMC@NLO+H++ (4FS, ttbb)	37.2
Powheg+Herwig7 (lepton+jets)	0.4
Powheg+Herwig7 (dilepton)	0.4