Efficient precision simulation of processes with many-jet final states at the LHC

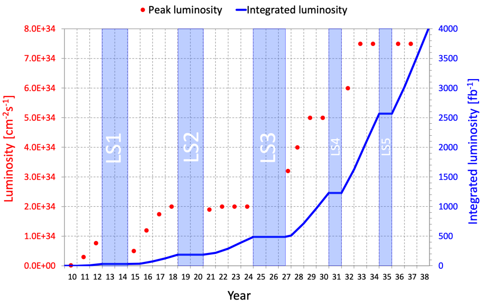
Enrico Bothmann (Göttingen), Taylor Childers (Argonne),
Christian Gütschow (UCL), Stefan Höche (FNAL),
Paul Hovland (Argonne), Joshua Isaacson (FNAL),
Max Knobbe (Göttingen), Robert Latham (Argonne)

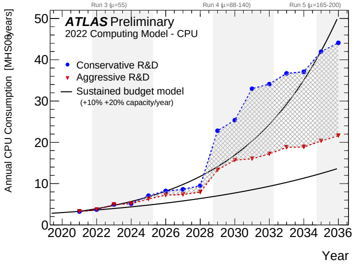
ACAT2024, Stony Brook

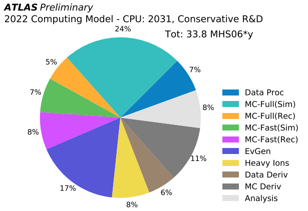
13 March 2024

Expected computing requirements

- projected evolution of computing resources sees cost of event generation on par with detector simulation
- LHC measurements in danger of being limited by Monte Carlo statistics







[CERN-LHCC-2022-005]

Targeted optimisation of CPU-based event generation

- Most event generation CPU spent on multi-leg NLO calculations [JHEP 08 (2022) 089]
  - used for main Standard Model processes: extremely large event sample sizes
  - relevant to measurements and searches alike

Targeted optimisation of CPU-based event generation

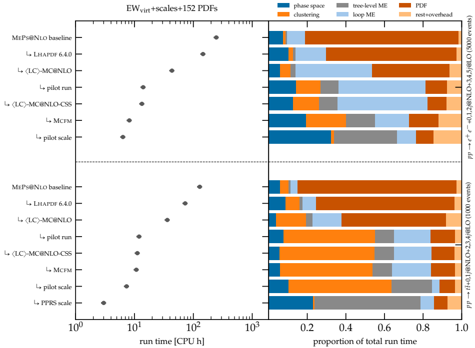
- Most event generation CPU spent on multi-leg NLO calculations [JHEP 08 (2022) 089]
  - used for main Standard Model processes: extremely large event sample sizes
  - relevant to measurements and searches alike
- Study CPU performance of Sherpa MEPS@NLO calculations for $e^+e^- + 0, 1, 2j$@NLO$+3, 4, 5j$@LO and $t\bar{t} + 0, 1j$@NLO$+2, 3, 4j$@LO
  - introduction of pilot run in Sherpa brings a factor 5 improvement
  - using analytic QCD loop amplitudes in the unweighting brings another factor 1.5
  - detailed write-up presented in [EPJC 82 (2022) 12]

Targeted optimisation of CPU-based event generation

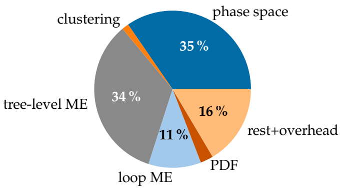
- Most event generation CPU spent on multi-leg NLO calculations [JHEP 08 (2022) 089]
  - used for main Standard Model processes: extremely large event sample sizes
  - relevant to measurements and searches alike
- Study CPU performance of Sherpa MEPS@NLO calculations for $e^+e^- + 0,1,2j$@NLO$+3,4,5j$@LO and $t\bar{t} + 0,1j$@NLO$+2,3,4j$@LO
  - introduction of pilot run in Sherpa brings a factor 5 improvement
  - using analytic QCD loop amplitudes in the unweighting brings another factor 1.5
  - detailed write-up presented in [EPJC 82 (2022) 12]

cumulative speed-ups for:	$pp \to e^+e^-$ + jets			$pp \to t\bar{t}$ + jets		
	runtime [CPU h/5k events]			runtime [CPU h/5k events]		
setup variant	old	new	speed-up	old	new	speed-up
no variations	20 h	5 h	4×	15 h	8 h	2×
EW_{virt}	35 h	5 h	6×	20 h	8 h	2×
EW_{virt} +scales	45 h	5 h	7×	25 h	8 h	4×
EW_{virt} +scales+100 PDFs	90 h	5 h	15×	55 h	8 h	7×
EW_{virt} +scales+1000 PDFs	725 h	8 h	78×	440 h	9 h	51×

Case study: ATLAS baseline configuration

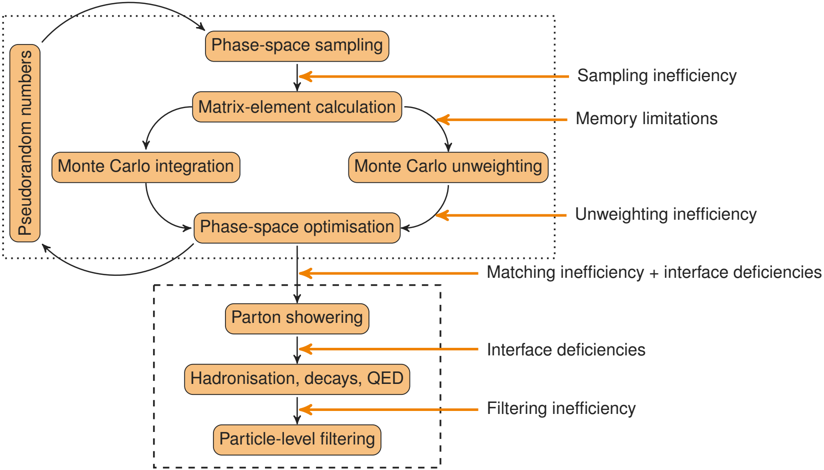


$pp \to e^+e^-$+0,1,2j@NLO+3,4,5j@LO



➜ CPU consumption **overall improved by factors of $\times$39 and $\times$43** for V+jets and $t\bar{t}$+jets

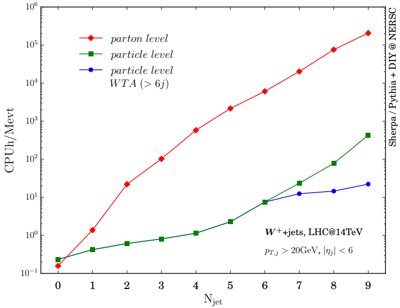
Other bottlenecks

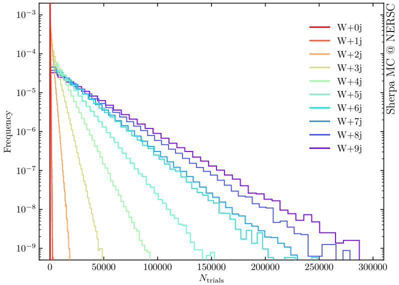


→ Lack of active development on infrastructure tools (LHE, HepMC, ...) set to become a major bottleneck going forward

Parton vs particle level

- Scaling of parton- and particle level analysed in [**PRD 100 (2019) 1**]
- cost of showering matrix elements with extra emissions **dominated by parton level**
  - number of diagrams grows factorially with every additional emission (at best exponentially when exploiting recursions a la `COMIX`)
- low-multiplicity matrix elements cheaper to regenerate entirely than to store on disk



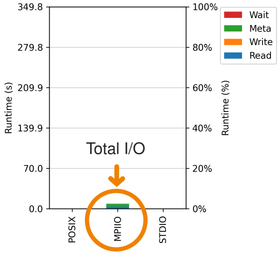


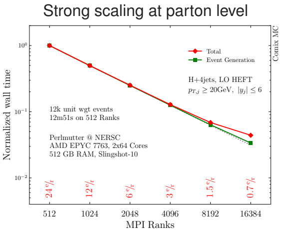
Introducing LHEH5

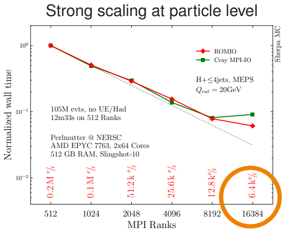
- established LHEF format is based on XML
  - flexible enough to add any desired feature
  - poses a challenge for I/O operations at scale
- new efficient LHE-like data format based on HDF5+HighFive proposed in [PRD 109 (2024) 1]

Name	Data type	Contents
VERSION	$3 \times$ int	Version ID
INIT	$10 \times$ double	beamA, beamB, energyA, energyB, PDFgroupA, PDFgroupB, PDFsetA, PDFsetB, weightingStrategy, numProcesses
PROCINFO	$6 \times$ double	procId, npLO, npNLO, xSection, error, unitWeight
EVENTS	$9 \times$ double	pid, nparticles, start, trials, scale, fscale, rscale, aqed, aqcd
PARTICLES	$13 \times$ double	id, status, mother1, mother2, color1, color2, px, py, pz, e, m, lifetime, spin
CTEVENTS	$9 \times$ double	ijt, kt, i, j, k, z1, z2, bbpsw, tlpsw
CTPARTICLES	$4 \times$ double	px, py, pz, e

I/O performance





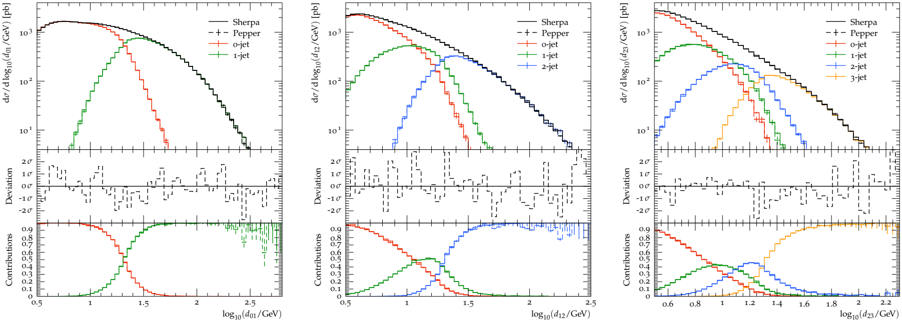


105M events / min

- overall I/O time reduced to below 1s per rank
  - time spent in I/O operations less than 5% when reading 128.85 GiB
- ideal for accessing back-fill queues at large computing centres

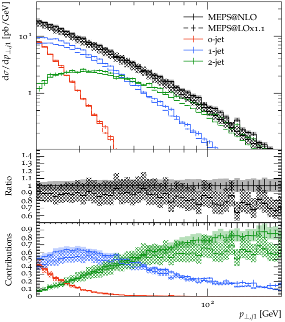
Comparison of parton-level event generators

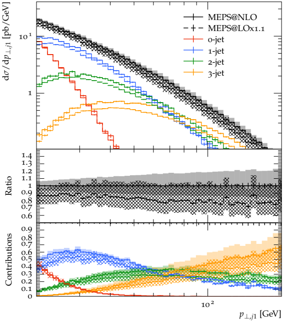
- validated for standard candle processes (Z+jets shown) at various multiplicities
- can mix and match generators to reduced computing time to the absolute minimum required for event simulation

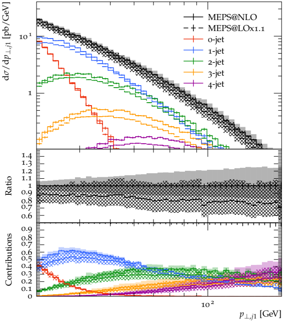


Improved modelling through high-multiplicity final states

- simulation of additional radiation at tree level clearly necessary for proper physics modelling of high-multiplicity final states
- hatched bands indicate the scale uncertainties from 7-point scale variations at LO, solid bands represent the corresponding band at NLO
- uncertainties inevitably increase with additional jet multiplicities as more of the phase space is systematically varied

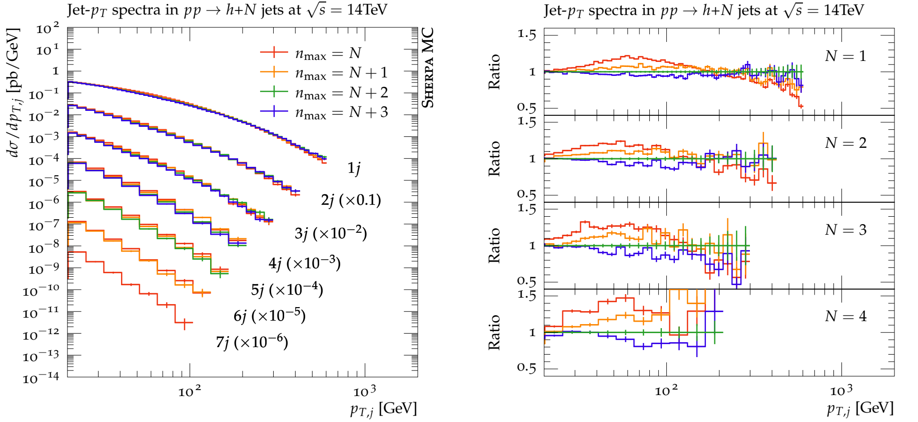






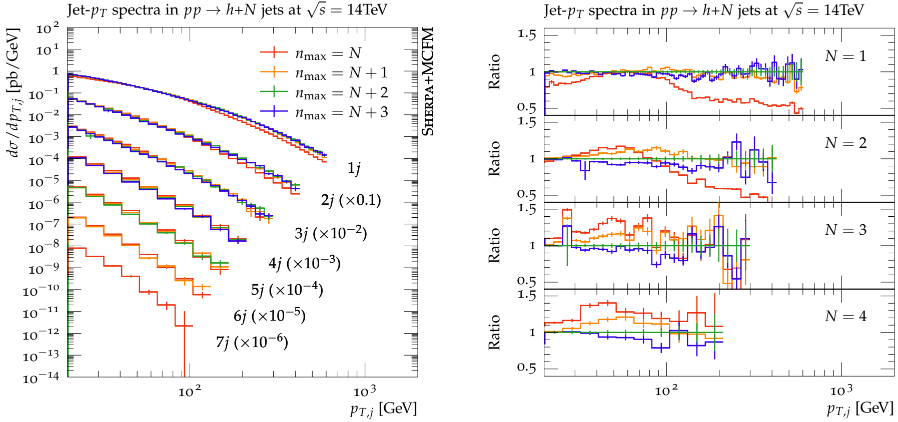
Case study: Higgs plus many jets at LO

- $H + 0, 1, 2, 3, 4, 5, 6, 7j$@LO, ratios normalised to $n_{\max} = 2$
- maximum jet multiplicity ($n_{\max}$) set to the number of measured jets, N (red), $N + 1$ (green), $N + 2$ (blue) and $N + 3$ (purple)



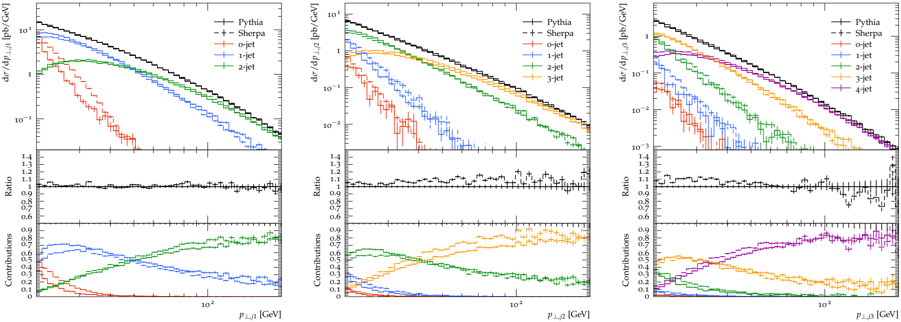
Case study: Higgs plus many jets at NLO

- $H + 0, 1, 2j$@NLO$+3, 4, 5, 6, 7j$@LO, ratios normalised to $n_{\max} = 2$
- maximum jet multiplicity ($n_{\max}$) set to the number of measured jets, N (red), $N + 1$ (green), $N + 2$ (blue) and $N + 3$ (purple)



More robust uncertainty estimates

- LHEH5 enables efficient substitution of various parts in the event generation chain
  - already supported by both Sherpa and Pythia!
- 10 % uncertainty seen in Z+jets due to different algorithmic choices in the parton showers



Future event generation workflows

- Approach 1: produce parton-level samples centrally with input from the MC developers, provide them in a shared space for all experiments
  - experiments run their preferred shower setup (✔)
  - allows for affordable plug & play between different models (✔)
  - lowers cost threshold for reproducing larger setups after some time if need be (✔)
  - requires more storage for parton-level events (✘)
  - new infrastructure needs to be set up and maintained (✘)

Future event generation workflows

- Approach 1: produce parton-level samples centrally with input from the MC developers, provide them in a shared space for all experiments
  - experiments run their preferred shower setup (✔)
  - allows for affordable plug & play between different models (✔)
  - lowers cost threshold for reproducing larger setups after some time if need be (✔)
  - requires more storage for parton-level events (✗)
  - new infrastructure needs to be set up and maintained (✗)
- Approach 2: run everything in one go, harnessing heterogeneous resources, possibly with in-memory transfer of GPU-accelerated calculation components
  - no intermediate storage for parton level events needed (✔)
  - minimal infrastructure changes required (✔)
  - parton-level events continue to cost twice as strictly necessary (✗)
  - regenerating larger setups from scratch will become painful (✗)

Summary

- computing cost of traditional CPU-based multi-leg event generation significantly reduced by factor 40–80 following dedicated profiling
- first production-ready portable LO event generator (cf. Enrico's talk) allows to incorporate GPU resources into high-precision simulations
- new `LHEH5` format allows for efficient parton-level event generation
  - excellent I/O performance for massive MPI applications
  - additional factor 3–6 speed-up for traditional grid resources
- facilitates more robust uncertainty estimates of parton-shower effects
- allows better exploitation/recycling of existing resources in subsequent large-scale production campaigns
- seeing latest performance improvements reflected in up-to-date projections from the experiments paramount for defining appropriate objectives going forward

Backup

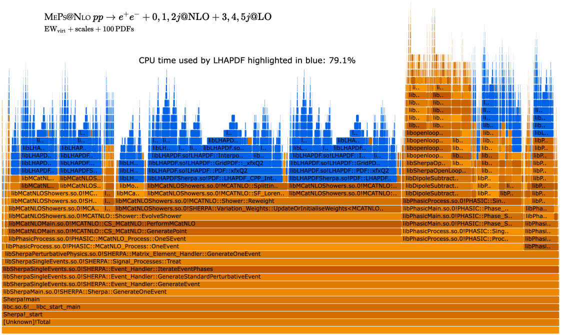
Initial profiling exercises

- ➜ first generator CPU profiling done by Tim Martin suggested per-event CPU dominated by LHAPDF

- ➜ graph shows PDF calls highlighted in blue (using LHAPDF 6.2.3)

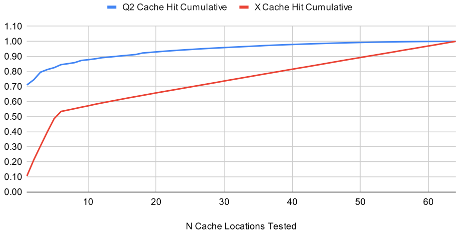
- ➜ maybe not completely surprising: multiweights originally not designed with hundreds of variations in mind [**EPJC 76 (2016) 11**]

MEPS@NLO $pp \to e^+e^- + 0,1,2j$@NLO $+ 3,4,5j$@LO
EW_{virt} + scales + 100 PDFs



- ➜ explore two approaches in parallel: make LHAPDF faster and rework LHAPDF call strategy

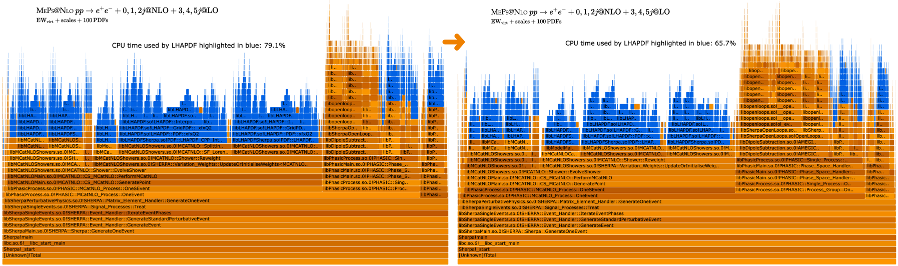
Improving LHAPDF



- first PDF-grid cache introduced in v6.3.0
  - rendered ineffective by PDF-call strategy used in Sherpa
  - nevertheless useful as case study
- follow-up release v6.4.0 with improved interpolation logic
  - revised cache implementation with improved memory layout (but well-matched call strategy in the generator still crucial)
  - pre-computation of shared coefficients of the interpolation polynomial along (x, Q^2) grid lines
  - results in factor 3 speed-up for single flavour computations
  - can achieve factor 10 speed-up when combining with multi-flavour caching

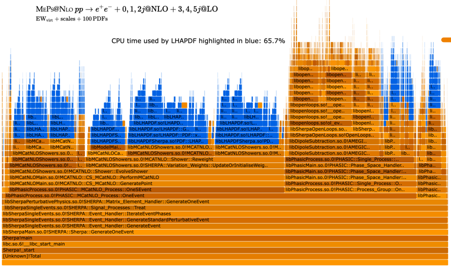
Impact of new LHAPDF

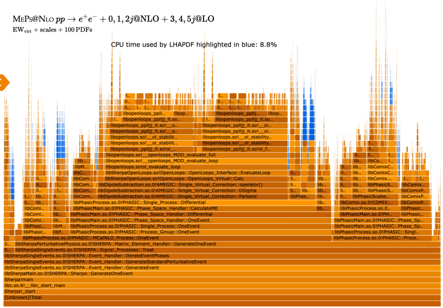
- ➜ ATLAS V+jets setup **overall 30% faster** using new LHAPDF release
  - ➜ switching from old ATLAS production default v6.2.3 to new v6.4.0 release



Internal restructuring and pilot run

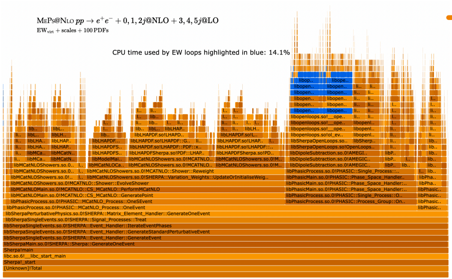
- perform the unweighting using a minimal setup and once an event is accepted, rewind RNG state and re-calculate accepted event using all the bells and whistles
- **achieves factor 5 speed improvement** for ATLAS setup (using LHAPDF 6.4.0 yields additional 6% speed-up)
- pilot run reduces CPU spent on evaluating PDFs to below 10%



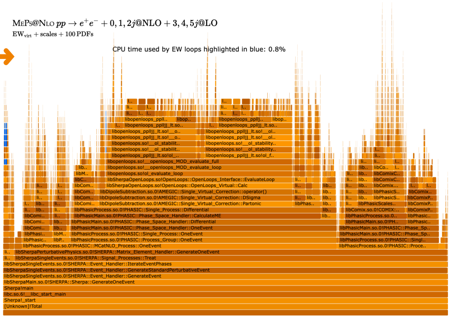


Internal restructuring in Sherpa 2.2.12: the pilot run

- ➜ CPU spent on calculating EW one-loop amplitudes going from 19% down to 0.8% when using the pilot run with the ATLAS V+jets setup
- ➜ nevertheless, $\sim$40% of the CPU still spent on calculating QCD loops

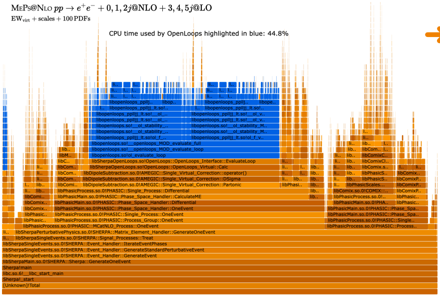


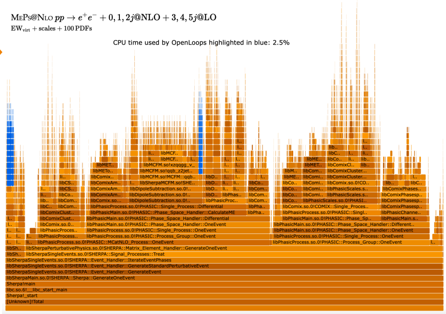
➜



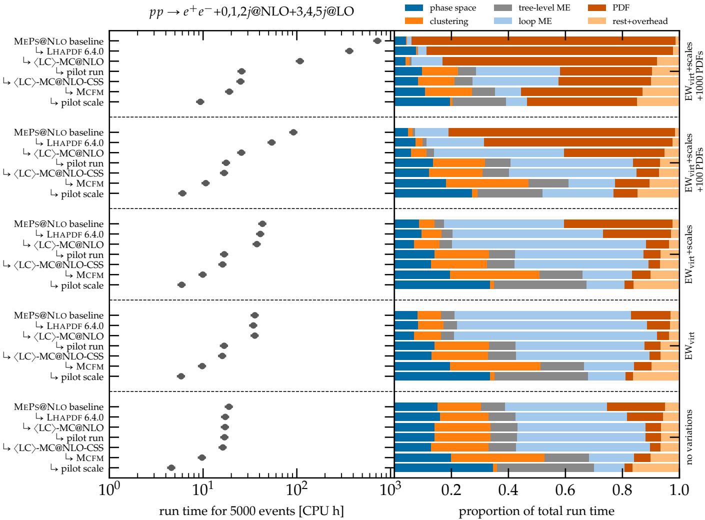
Analytic vs numerical QCD loop amplitudes

- employ analytic one-loop amplitudes (if available) in the pilot run using Sherpa-MCFM interface [**EPJC 81 (2021) 12**]
- yields **additional ∼35% speed improvement** for the V+jets setup

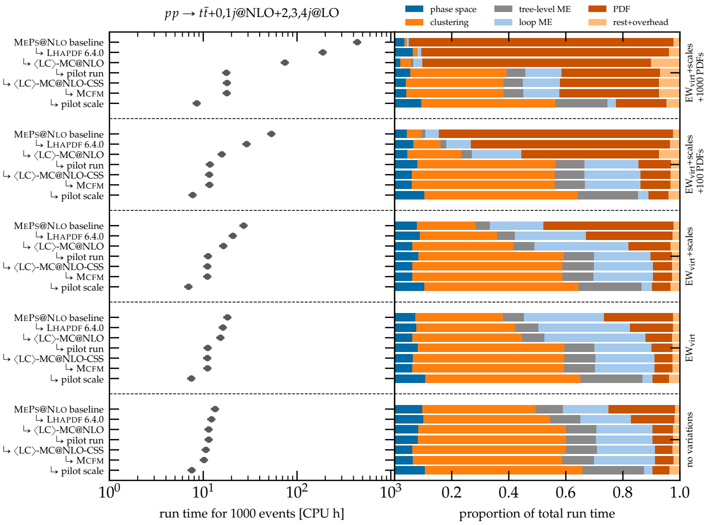




Breakdown of CPU budget in V+jets



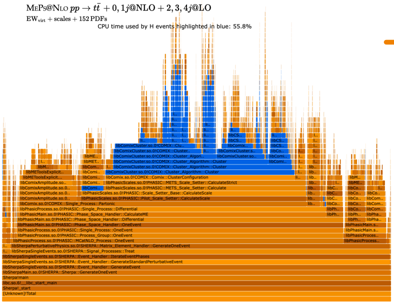
Breakdown of CPU budget in $t\bar{t}$+jets



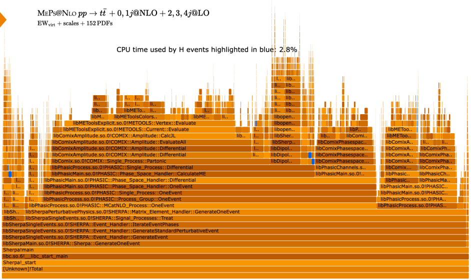
Cluster-independent scale definition

➜ employ clustering-independent scale definition ($H'_{\mathrm{T}}/2$) for $\mathcal{H}$-events in $t\bar{t}$+jets (already used in V+jets baseline setup)

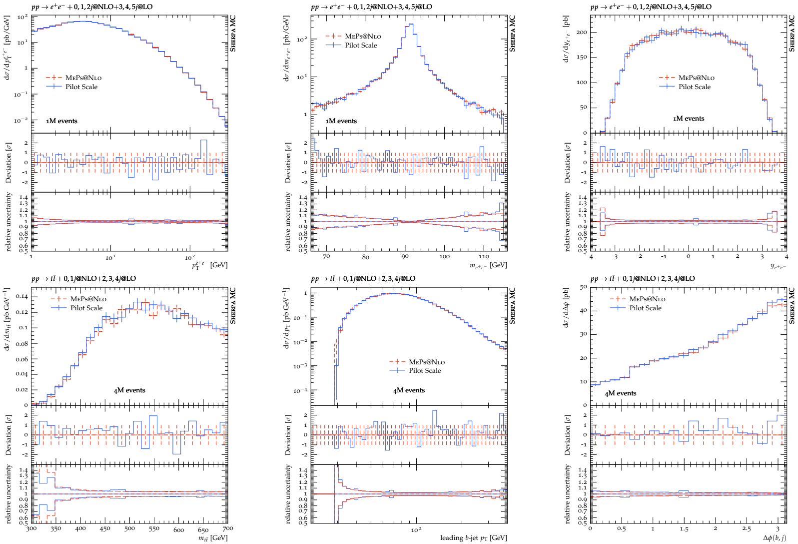
➜ yields **additional factor 2 speed-up** of the overall run time



➜

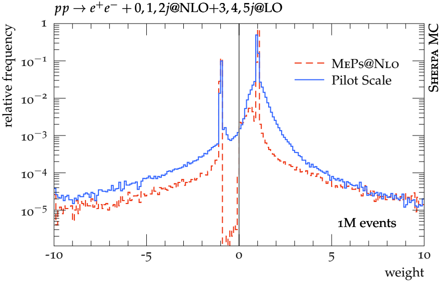


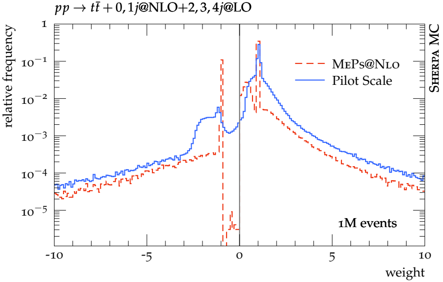
Comparison of MEPS@NLO vs Pilot Scale strategy



Weight distribution for pilot scale

- weight distributions for partially unweighted events after matching and merging:





- second unweighting would reduce the efficieny by less than factor 2 for large N_{events}

Benchmarking with state-of-the-art event generators

- comparison of Sherpa's COMIX with PEPPER+CHILI, on a single core Intel(R) Core(TM) i3-8300 CPU at 3.70GHz with 8MB L3 cache.
- samples generated with a given target for the total cross section uncertainty ("Tot. unc.")
- "Speed-up" gives the walltime gain factor of PEPPER+CHILI vs. COMIX
- PEPPER+CHILI: $Z + 0, 1j$ generated using helicity summing, while the higher ones use helicity sampling, thereby achieving the best possible performance in each case
- factorial scaling in PEPPER causes COMIX to win at very high multiplicities

Process	Tot. unc. [%]	Sherpa (Comix)			Pepper+Chili			Speed-up
		Walltime [s]	Mem. (USS) [MB]	Eff. [%]	Walltime [s]	Mem. (USS) [MB]	Eff. [%]	
Z+0j	0.089	68	62	22	10	40	43	6.8
Z+1j	0.19	76	66	5.3	31	33	10	2.5
Z+2j	0.99	92	64	0.28	10	35	1.4	9.2
Z+3j	3.8	95	65	0.037	36	43	0.097	2.6
Z+4j	14	122	115	0.0050	71	133	0.016	1.7