Towards efficient and sustainable event generation for the HL-LHC

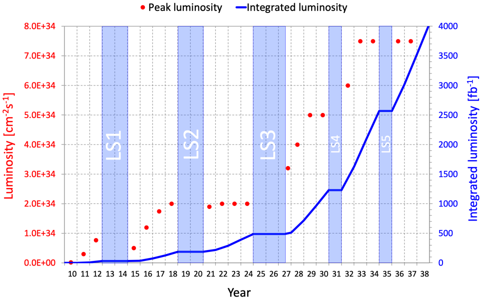
Christian Gütschow

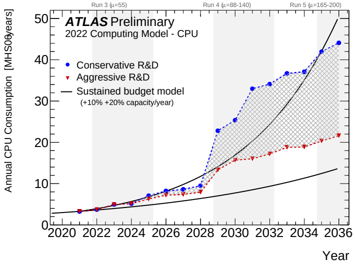
Swift-HEP/ExaTEPP, Warwick

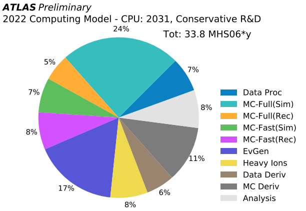
12 November 2024

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



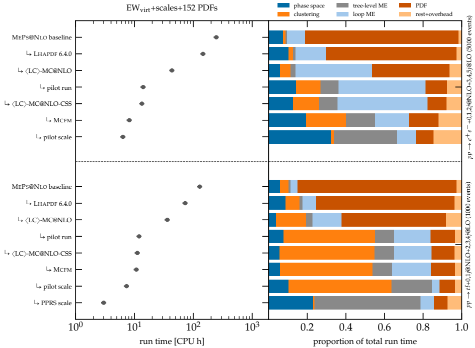




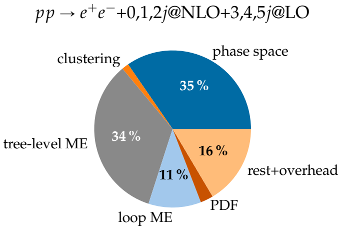
[CERN-LHCC-2022-005]

Targeted optimisation of CPU-based event generation

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



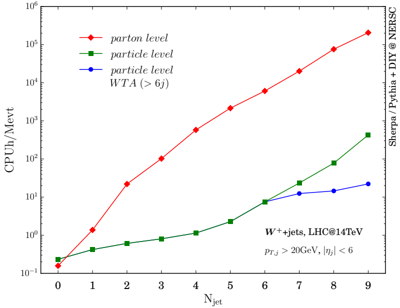
[**EPJ C82 (2022) 12**]

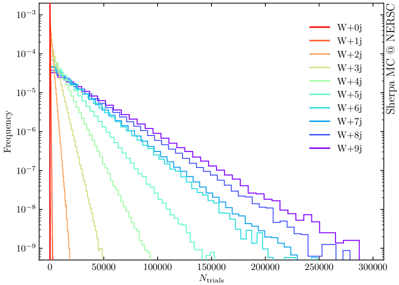


- CPU consumption **overall improved by factors of ×39 and ×43** for V+jets and $t\bar{t}$+jets

Parton vs particle level

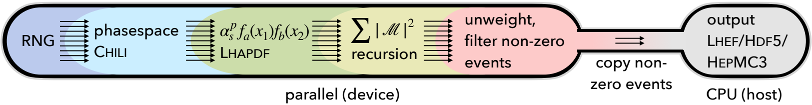
- Scaling of parton- and particle level analysed in [**PRD 100 (2019) 1**]
- cost of showering matrix elements with extra emissions **dominated by parton level**
  - number of diagrams grows factorially with every additional emission (at best exponentially when exploiting recursions a la COMIX)
- low-multiplicity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





Introducing Pepper: a portable ME generator for the HL-LHC

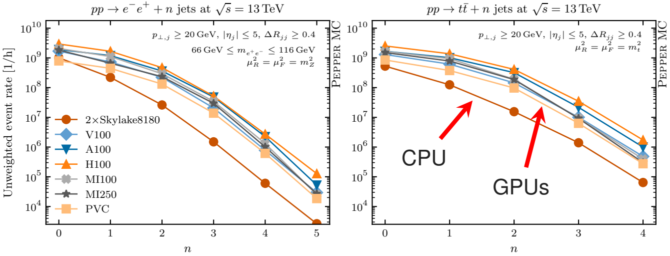
- Focus on highest multi (e.g. $e^+e^- + 5$, $t\bar{t} + 4$)
  this is beyond small scale computing → WLCG / HPC
- 10–20 years ago: Homogeneous CPU+RAM architectures
- This is undergoing a big change (partly due to AI trends)
  - HPC moves to exascale era → scalability
  - GPU acceleration → portability
- PEPPER addresses both aspects with **MPI**, **HDF5** and **Kokkos**
- PEPPER **parallelises** the entire parton-level event generation:



- Tested Xeon CPU, Intel/AMD/Nvidia GPU, HPC systems
  - Covers all (pre-)exascale architectures on previous slide
  - Scalable from a laptop to a Leadership Computing Facility

Comparing runtimes on relevant architectures

- Excellent performance across a wide range of architectures
- Portability provided by Kokkos: one code-base compiled for different architectures

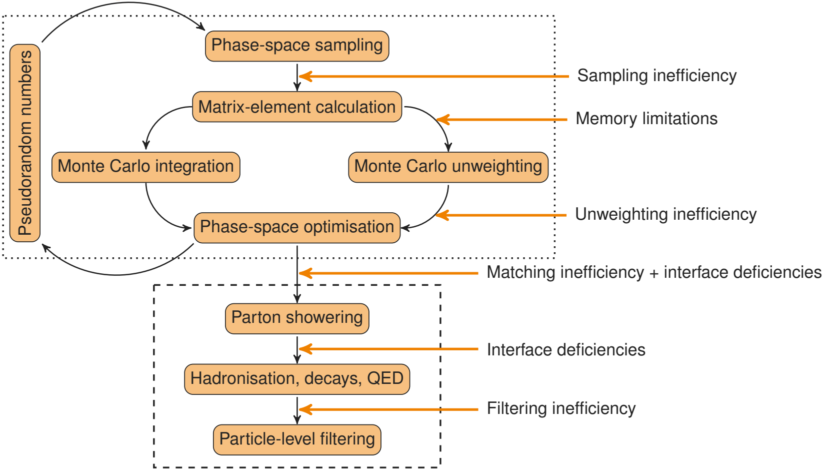


MEvents / hour	2×Skylake8180	V100	A100	H100	MI100	MI250	PVC
$pp \to t\bar{t} + 4j$	0.06	0.5	1.0	1.7	0.4	0.3	0.3
$pp \to e^- e^+ + 5j$	0.003	0.03	0.05	0.1	0.03	0.03	0.02

Portability: Aurora example

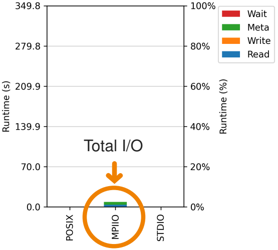
- Estimate "roughly 330 billion [leptonically decaying V+jets] events" required for HL-LHC [arXiv:2112.09588]
  - "**SHERPA 2.2.11 setup would exceed budget by 16%**"
- Assuming all 330 billion events are $Z + 4j$, production cost at parton-level would be:
  - 240M CPUh COMIX @ Intel E5-2650 v2 CPU
  - 380k GPUh PEPPER @ Nvidia A100 → **This would be 8h on Aurora (with PVC)**

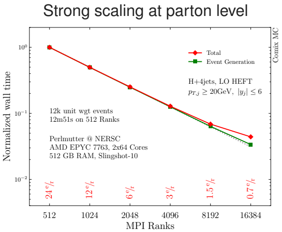
Typical MC production chain

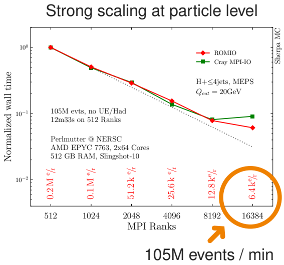


→ Lack of active development on infrastructure tools (LHE, HepMC, ...) set to become next major bottleneck going forward

New LHE-H5 format





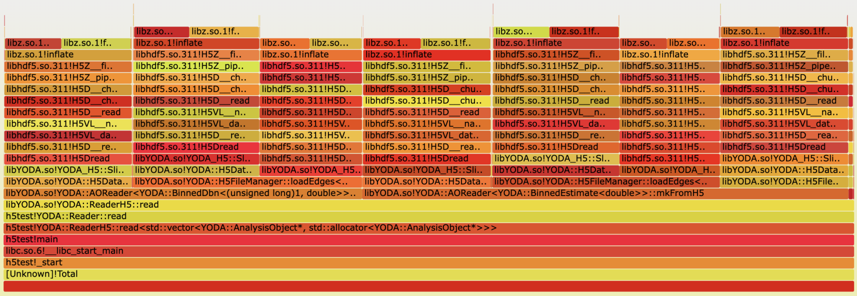


- new efficient LHE-like data format based on HDF5+HighFive proposed in [**PRD 109 (2024) 1**]
- overall I/O time reduced to below 1s per rank
  - time spent in I/O operations less than 5% when reading 128.85 GiB
- ideal for accessing back-fill queues at large computing centres
- already support by Sherpa and Pythia

New YODA-H5 format

- ➜ provides an alternative for applications where ASCII is currently a bottleneck
- ➜ integer-factor speed-ups for I/O operations; nicely balanced profiles

	ASCII/H5 exec times		
Raw ASCII size	excl. compression	incl. compression	gz vs raw h5
$\mathcal{O}$(5kB)	0.36	0.35	0.51
$\mathcal{O}$(3MB)	2.69	1.03	6.34
$\mathcal{O}$(33MB)	2.15	0.41	4.80



Software releases and dissemination

- new YODA 2.0 major release + publication [arxiv:2312.15070]
- new Rivet 4.0 major release + publication [SciPost Phys Codebases 2024 36]
- new Sherpa 3.0 major release + publication [arxiv:2410.22148]
- all including various new features to improve HPC friendliness
- Swift-HEP contributions also presented at ACAT and ICHEP this year
- over the summer: [IPPP workshop] on Monte Carlo support tools
- upcoming: follow-up workshop at DESY
- next year: more MC-support-tools-oriented efforts (supported by CoSeC)

Outlook

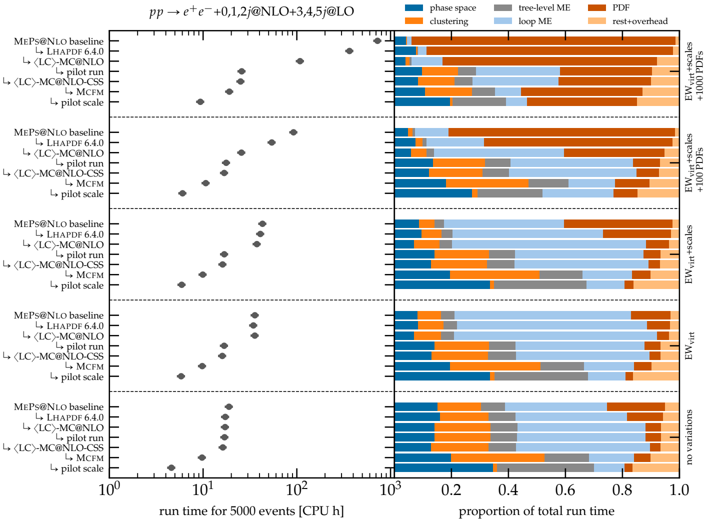
- Lots of discussion points that could feed into Swift-HEP2 efforts:
  - LHAPDF: support for 2D spline fits (first promising results in Python, now being propagated to C++), reduce memory and look-up times by optmising grid binning and many more!
  - SHERPA: adapt the codebase to better support modern computing architectures and workflows with RSE support
  - HEPMC: needs systematic profiling in light of HPC usage; extend to efficiently support factorised parton-shower systematics
  - Showers: extend recent reweighting efforts to reduce cost of assessing non-perturbative uncertainties (hadronisation, MPI, decays) [**arXiv:2308.13459**]
  - Hadron-decay modelling: no reward for theory careers, but existing implementations will need to be maintained
  - Spin-density propagation: avoid mixing of showers via decay modules by providing minimal standard interface in terms of array representation of the matrices and spinors
- General theme: make event generation maximally HPC friendly and develop the MC infrastructure to deliver the best possible calculations and robust uncertainties

Summary

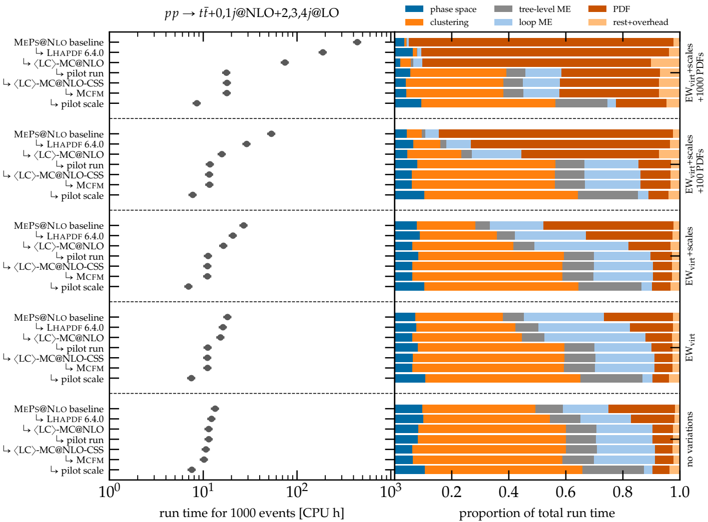
- computing cost of traditional CPU-based multi-leg event generation significantly reduced by factor 40–80 following dedicated profiling [**EPJ C82 (2022) 12**]
- first production-ready portable LO event generator allows to incorporate GPU resources into high-precision simulations [**SciPost Phys 15 (2023) 4**]
- this constitutes the first realistic and sustainable approach towards large scale event generation in the HL-LHC era
- well positioned to exploit Leadership Computing Facilities across the world
  - ideally suited for high-precision calculations if quotas can be obtained
- focus now shifting towards better calculations and more robust uncertainty estimates

Backup

Breakdown of CPU budget in V+jets



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

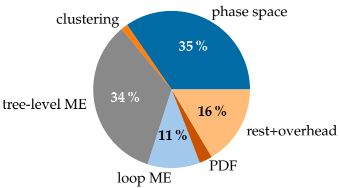


Parton-level event generation

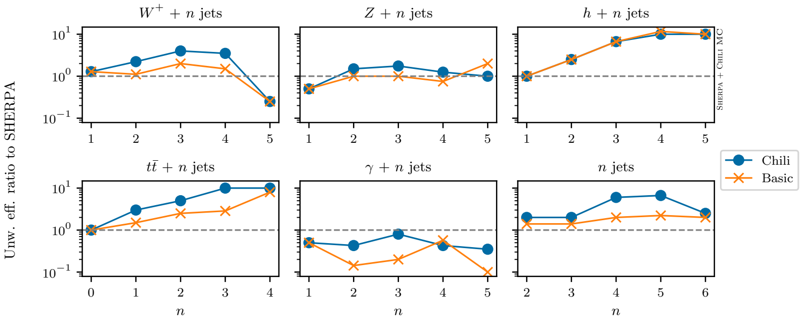
$$\sigma_{pp\to X_n} = \sum_{ab} \int dx_a dx_b d\phi_n \times f_a(x_a,\mu_F^2) f_b(x_b,\mu_F^2) \times |\mathcal{M}_{ab\to X_n}|^2 \times \Theta(p_1,\ldots,p_n)$$

- Large portion of MC time spend in ME+PS
- Components we need to consider:
  - Phase space generation → CHILI
  - PDFs → LHAPDF
  - Tree-level matrix elements → PEPPER

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



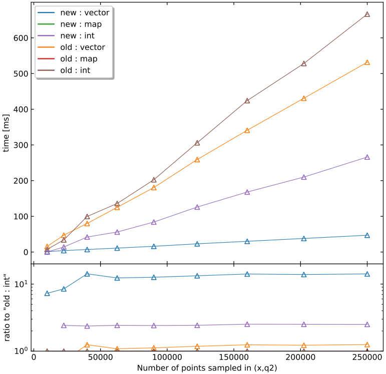
Chili performance compared to SHERPA default



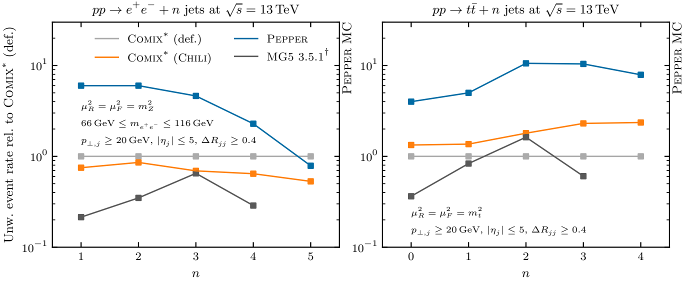
- CHILI phase-space generator uses simple (MCFM-inspired) structure: one t-channel + adjustable number of s channels [**SciPost Phys 15 (2023) 4**]
  - Portable (ported built-in CHILI in PEPPER)
  - Efficiency on par with recursive COMIX phase-space

Making LHAPDF GPU-ready

- Byproduct of porting exercise
  - PDF evaluations not critical
  - copying of data is
- Huge performance gains
  - 3-10x speedup per PDF evaluation
  - more possible by changing MC workflows
- OpenMPI used for efficient initialisation
  - constant init time vs. infinite init time [PRD 100 (2019) 1]
- Added CUDA + Kokkos interface/version
  - excellent computing performance / accuracy
  - portable version used for the remaining talk



PEPPER + CHILI: baseline CPU performance



- First complete CHILI + PEPPER benchmark
  - Unweighted event throughput compared to COMIX*
  - Constitutes baseline single-threaded performance of currently available competitive algorithms
  - Standalone PEPPER performs better than COMIX, but PEPPER's real goal is portability

Numbers generated on Intel Xeon E5-2650 v2
* Partonic processes split into to *g*/*q* groups (not SHERPA default)
† Modified to match efficiency convention of [PRD 101 (2020) 7]

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

Super-computing for the HL-LHC

- PEPPER runs on all leading systems in EU and U.S.
- ATLAS estimated roughly 330 billion leptonically decaying V+jets events would be required for HL-LHC [JHEP 08 (2022) 089]
- Time for V+j sample:
  - 4h Frontier
  - 6h Aurora
  - 8h Leonardo
  - 15h Lumi

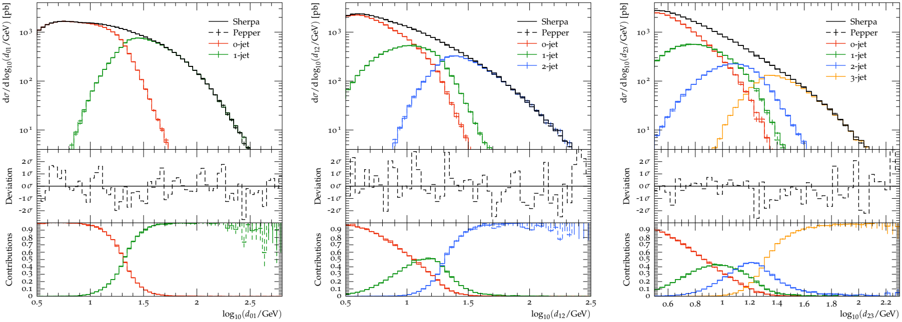
Introducing LHEH5

- 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

Tool-chain integration

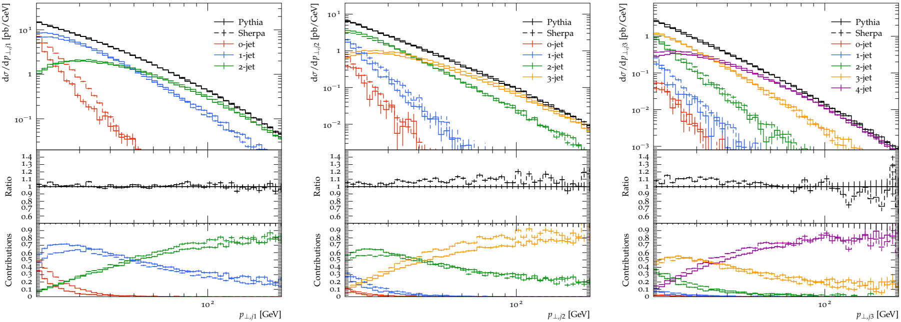
- ➜ validated for standard candle processes (Z+jets shown) at various multiplicities
- ➜ can mix and match generators to reduced computing time to the absolute minimum required for event simulation



[PRD 109 (2024) 1]

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 (✗)