Efficiency of Event Generators

Enrico Bothmann

Institute for Theoretical Physics, University of Göttingen

QCD@LHC 2024
7—11 October, Freiburg, Germany

MCnet

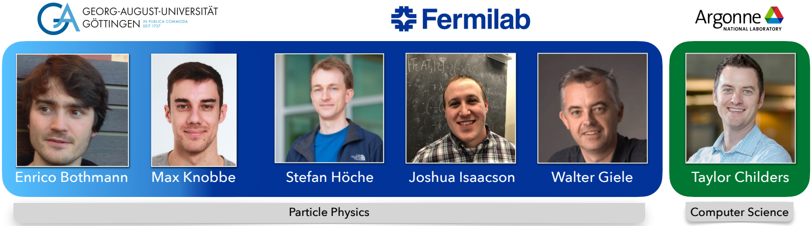
Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

Prologue

Disclaimer

- Ignore recent ∼20–40× speed-ups in Sherpa 2.2.12, 2.2.13 and 3.0.0
  [EB et al.] arXiv:2209.00843
- Ignore ML developments
  → already discussed by Steffen on Monday
- Focus on recent developments in the novel Pepper event generator

- The Pepper team:



1. Introduction
2. Efficiency & portability with Pepper
3. Numerical stability for higher-order calculations
4. Summary

1 Introduction

2 Efficiency & portability with Pepper

3 Numerical stability for higher-order calculations

4 Summary

Introduction

Why improve event generation efficiency?

- High statistics at HL-LHC & excellent detector performance

  → Need for accurate & expensive simulated event samples

  → Poor event generation efficiency can limit experimental success

  [HSF Physics Event Generator WG] arXiv:2004.13687 arXiv:2109.14938

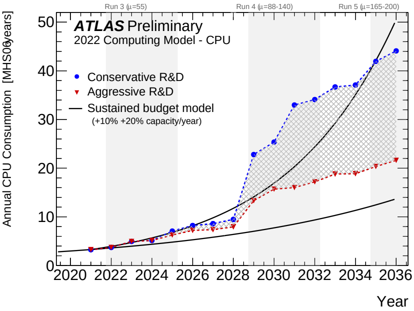
What dominates the computing budget?

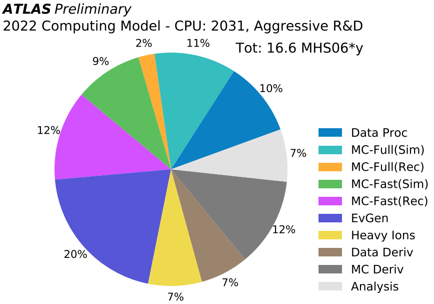
- Which physics processes?
- Parton or particle level?
- Which final-state jet multiplicities?

In contrast to computers, human resources are scarce.

We can't afford to make incremental improvements.

LHC projected CPU consumption for event generation





- ATLAS: 14 % CPU for event generation in 2022
- expect ca. 20 % during HL-LHC ("Aggressive R&D" scenario 2031)

Miha's slide on background uncertainties in VH production

Impact of background theory uncertainties in H→bb/cc

ATLAS

$\mu_{VH}^{bb} = 0.91 \pm 0.10\,(\text{stat.}) \pm 0.12\,(\text{syst.})$

$\mu_{VH}^{cc} = 1.0 \pm 4.0\,(\text{stat.}) \pm 3.5\,(\text{syst.})$

Source of uncertainty		$VH, H \to b\bar{b}$	$VH, H \to c\bar{c}$
Total		0.151	5.29
Statistical		0.097	3.94
Systematic		0.116	3.53
Statistical uncertainties			
Data statistical		0.089	3.70
$t\bar{t}$ $e\mu$ control region		0.009	0.06
Background floating normalisations		0.034	1.23
Other VH floating normalisation		0.007	0.24
Simulation samples size		0.023	1.61
Experimental uncertainties			
Jets		0.028	1.00
E_T^{miss}		0.009	0.24
Leptons		0.004	0.23
b-tagging	b-jets	0.020	0.30
	c-jets	0.013	0.73
	light-flavour jets	0.006	0.67
Pile-up		0.009	0.24
Luminosity		0.006	0.08
Theoretical and modelling uncertainties			
Signal		0.073	0.56
Z + jets		0.039	1.76
W + jets		0.055	1.41
$t\bar{t}$ and Wt		0.018	1.03
Single top quark (s-, t-ch.)		0.010	0.15
Diboson		0.032	0.51
Multi-jet		0.006	0.57

ATLAS-CONF-2024-010

- Background modeling related uncertainties are among the largest in the ATLAS and CMS H→bb/cc searches
- MC statistical uncertainty also poses an issue due the computational complexity in generating the multi-leg NLO samples
- At HL-LHC it will be extremely challenging to get background predictions with systematic uncertainty matching the data statistics

CMS $\mu_{VH}^{bb} = 1.15 \pm 0.21$

	$\Delta\mu$	
Background (theory)	+0.043	−0.043
Signal (theory)	+0.088	−0.059
MC sample size	+0.078	−0.078
Simulation modeling	+0.059	−0.059
b tagging	+0.050	−0.046
Jet energy resolution	+0.036	−0.028
Int. luminosity	+0.032	−0.027
Jet energy scale	+0.025	−0.025
Lepton ident.	+0.008	−0.007
Trigger ($\vec{p}_T^{miss}$)	+0.002	−0.001

PRD 109 (2024) 092011

CMS $\mu_{VH}^{cc} = 7.7 \pm 3.7$

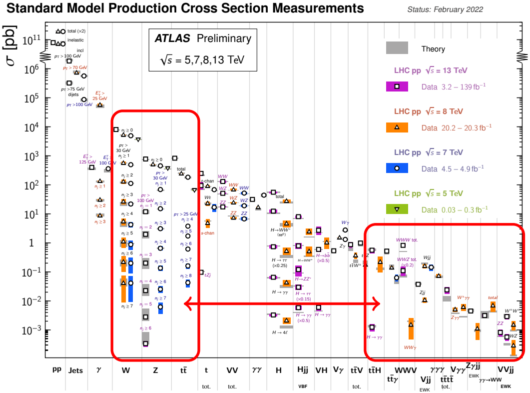
Uncertainty source	$\Delta\mu / (\Delta\mu)_{tot}$
Statistical	85%
Background normalizations	37%
Experimental	48%
Sizes of the simulated samples	37%
c jet identification efficiencies	23%
Jet energy scale and resolution	15%
Simulation modeling	11%
Integrated luminosity	6%
Lepton identification efficiencies	4%
Theory	22%
Backgrounds	17%
Signal	15%

PRL 131 (2023) 061801

October 10, 2024 Miha Muškinja 11

Let's make sure this does not become a common finding.

Which physics processes?



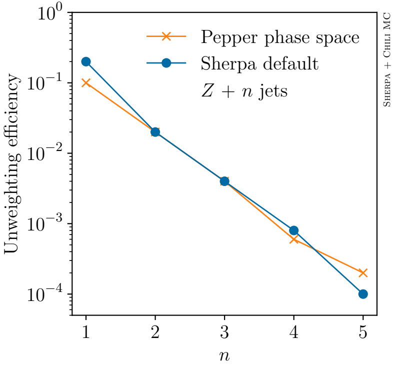
[ATLAS] https://twiki.cern.ch/twiki/bin/view/AtlasPublic/StandardModelPublicResults

- Signals: High multiplicity but comparably low complexity
- Main backgrounds: High multiplicity and high complexity

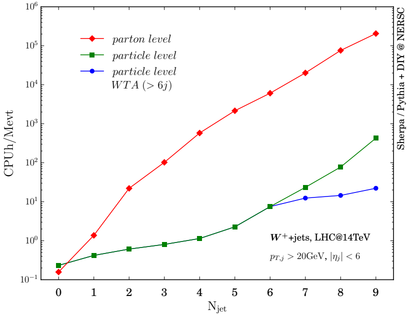
Heavy hitter background simulations

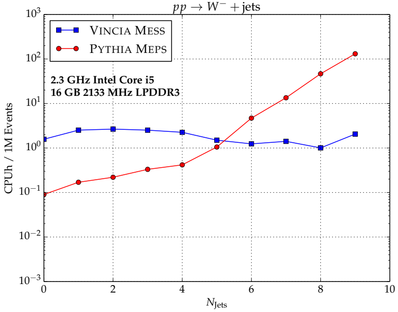
- ATLAS' state-of-the-art Sherpa samples
  - $pp \to e^+e^- + 0, 1, 2j@NLO + 3, 4, 5j@LO$
  - $pp \to t\bar{t} + 0, 1j@NLO + 2, 3, 4j@LO$
  - Unweighted events needed for downstream processing

- 60–80 % time spent in LO matrix elements (ME) and phase space
  optimized & using analytic loop MEs
  [EB et al.] arXiv:2209.00843
- Reason: low unweighting efficiencies and expensive ME for high jet multiplicities
  [Höche,Prestel,Schulz] arXiv:1905.05120



Timing distribution: scaling with multiplicity





- Hard scattering simulation much more demanding than particle-level remainder [Höche,Prestel,Schulz] arXiv:1905.05120
- Complexity of multi-jet merging can be reduced to achieve linear scaling using sector showers [Brooks,Preuss] arXiv:2008.09468
  $\rightarrow$ not a problem in principle

1 Introduction

2 Efficiency & portability with Pepper

3 Numerical stability for higher-order calculations

4 Summary

Pepper aims and preconditions

With the identified bottleneck & multiplicity scaling, we set our . . .

Figure of merit for efficient event generation

unweighted parton-level event throughput
for highest relevant jet multiplicity of heavy-hitter processes
e.g. $pp \to e^+e^- + 5j$, $pp \to t\bar{t} + 4j$ (more for HL-LHC era?)

In devising optimal algorithms, consider current computational trends:

- 10–20 years ago: homogeneous CPU+RAM architectures
- Most modern HPC uses GPU accelerators enabling large throughput
- Many computing vendors, heterogeneous architectures

→ Portability required to achieve efficiency on wide range of hardware

Pepper amplitudes

- Berends–Giele recursion for best multi-jet scaling behaviour

  Based on early 2000s performance studies:

  [Dinsdale, Ternick, Weinzierl] arXiv:hep-ph/0602204

  [Duhr, Höche, Maltoni] arXiv:hep-ph/0607057
- Colour summing to allow for lockstep GPU evaluation
- Use minimal colour basis for general QCD amplitudes

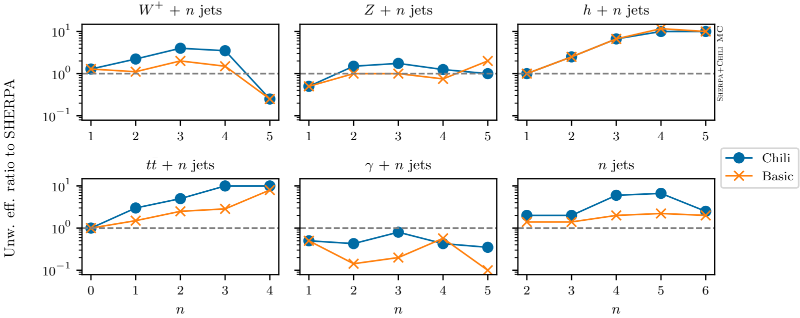
$$\mathcal{O}((n-1)!^2) \to \mathcal{O}((n-2)!^2)$$

  [Melia] arXiv:1304.7809 arXiv:1312.0599 arXiv:1509.03297

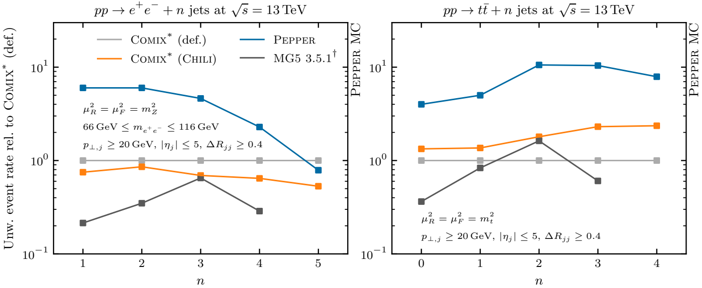
  [Johansson,Ochirov] arXiv:1507.00332
  - Combined for the first time with Berends-Giele recursion
  - Generalised in our implementation for e^+e^-+jets amplitudes
- Helicity sampling to avoid additional 2^n scaling

Pepper phase space

- Chili phase-space generator uses simple MCFM-inspired structure: one t channel + adjustable number of s channels
  [EB et al.] arXiv:2302.10449
  - Simple and thus easily ported
  - Rambo-like speed
  - Efficiency on par with complex recursive Sherpa phase space
  - Full Chili vs. "Basic" Chili (minimum number of s channels):



Baseline performance comparison (single-threaded)



- Unweighted event throughput compared to Sherpa (Comix*)
- Constitutes baseline single-threaded performance of currently available competitive algorithms
- Novel Pepper generator performs better than Comix, but Pepper's real goal is portability [EB et al.] arXiv:2311.06198

Numbers generated on Intel Xeon E5-2650 v2

* Partonic processes split into to g/q groups (not Sherpa standard)

† Modified to match efficiency convention of [Gao et. al] arXiv:2001.10028

Why portability?

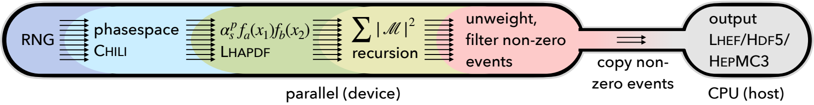
- Many computing vendors, heterogeneous architectures
- (Pre-)Exascale computing systems intentionally diverse

→ Portable ME generator projects: Pepper, madgraph4gpu, MadFlow

[EB et al.] arXiv:2311.06198 [Hageböck et al] arXiv:2312.02898 [Carrazza et al] arXiv:2106.10279

Portability is baked into Pepper

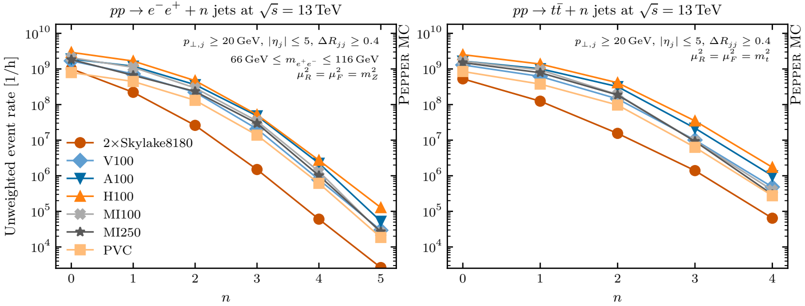
- Focus on highest multi (e.g. $e^+e^- + 5$, $t\bar{t} + 4$)
  this is beyond small scale computing → WLCG / HPC
- Computing is undergoing a big change (partly due to AI trends)
  - HPC moves to exascale era → scalability
  - GPU acceleration → portability
- Pepper addresses both aspects with MPI, HDF5 and CUDA & 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
- Tested scaling on up to 1024 Nvidia GPU on Argonne's Polaris cluster

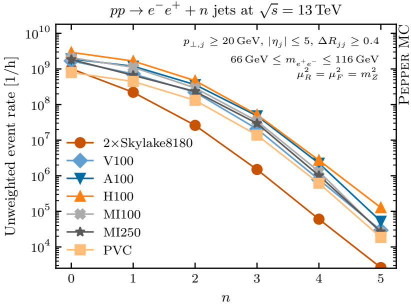
Comparing runtimes on relevant architectures

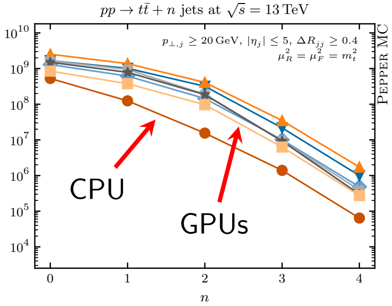
- Excellent performance across a wide range of architectures
- One code-base compiled for different architectures



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

- Excellent performance across a wide range of architectures
- One code-base compiled for different architectures

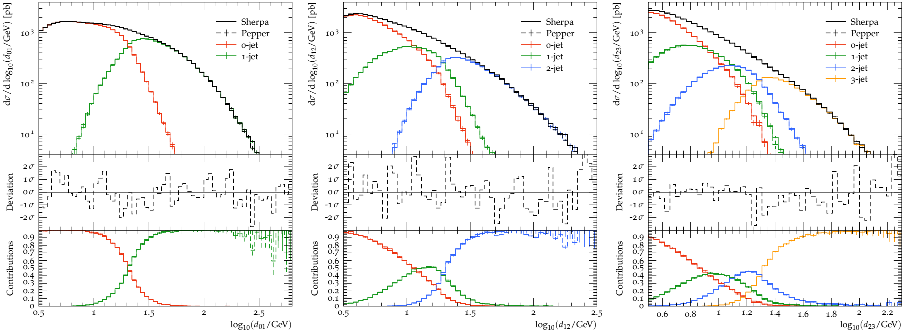




MEvents / hour	2×Skylake8180	V100	A100	H100	MI100	MI250	PVC
$pp \to t\bar{t} + 4j$	0.06	0.5	1.0	1.7	0.4	0.3	0.3
$pp \to e^- e^+ + 5j$	0.003	0.03	0.05	0.1	0.03	0.03	0.02

Toolchain integration

- Pepper writes out reusable HDF5 based LHEH5 parton-level events
- Particle-level simulation via Sherpa or Pythia
  $\rightarrow$ Validated LHEH5-based framework [EB et al.] arXiv:2309.13154



- 3–5$\times$ speed-up for heavy-hitter ATLAS Sherpa MEPS@NLO set-ups

1 Introduction

2 Efficiency & portability with Pepper

3 Numerical stability for higher-order calculations

4 Summary

Numerical stability for higher-order calculations

- Numerically stable scattering ME near infrared (IR) limit
- Important for success of (future) collider experiments
- (N)NLO subtraction methods
  - $\rightarrow$ cancellation of large numbers between real & subtraction terms
  - $\rightarrow$ naive algorithms at double precision (DP) give instable results
- Example of a slicing calculation at NNLO
  - Use unphysical IR cutoff τ_{cut} $\rightarrow$ must ensure independence of results
  - Quad precision (QP) rescue system $\rightarrow$ $\mathcal{O}(10\ldots100)$ time penalty
  - Post-processing to remove outliers $\rightarrow$ labor intensive

$\rightarrow$ Cf. e.g. Event generators' and N(n)LO codes' acceleration workshop, CERN, Nov 2023

Numerically stable amplitudes in Pepper

Idea

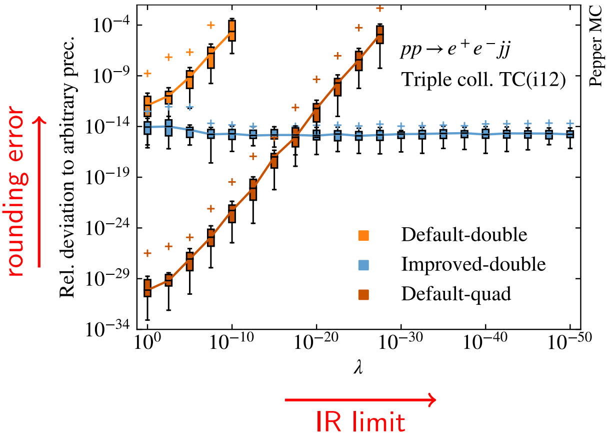
- Use physics knowledge to write propagators, vertices and ext. states in terms of large and small momentum components
- Rewrite expressions to avoid numerically unstable operations

- Novel DP algorithms with smaller rounding errors than naive QP implementations @ smaller computational cost
- Implemented in Pepper $\rightarrow$ perfect source of stable & fast real/subtraction terms for (N)NLO calculations
  - Could drastically reduce NNLO simulation runtimes
  - Real-real contribution can be $\sim 2/3$ of total runtime

Pepper IR stability deep into infrared limit

Study behaviour in IR limits of $pp \to e^+e^- + 2j$ at LO

[EB, Campbell, Höche, Knobbe] arXiv:2406.07671



Pepper IR stability for LHC physics at NNLO

- $u\bar{u} \to e^+e^- + X$ at NNLO with jettiness slicing
- MCFM vs. MCFM+Pepper (= "BG recursion")

[EB, Campbell, Höche, Knobbe] arXiv:2406.07671

$\tau_{\rm cut}$	$\delta\sigma_{\rm NNLO}(u\bar{u} \to e^+e^- + X)$ [pb]		
	naive double	improved double	
		analytic ME	BG recursion
10^{-2}	28.11(5)	28.12(5)	28.10(7)
10^{-3}	27.4(3)	27.0(1)	27.0(1)
10^{-4}	27.2(4)	27.4(3)	27.0(3)
10^{-5}	instable	27.2(9)	27.1(8)
10^{-6}	instable	27.7(18)	27.4(16)

IR limit ↓

- Uncertainty driven by subtracted real-emission corrections
- Must use asymptotic $\tau_{\rm cut} \to 0$, but naive double instable
- Has been combined with projection-to-Born improved subtraction

  [Campbell et al.] arXiv:2408.05265

1 Introduction

2 Efficiency & portability with Pepper

3 Numerical stability for higher-order calculations

4 Summary

Summary

- Current bottleneck in (Sherpa4LHC) event generation:
  - Large MEPS@NLO background samples for V+jets, tt+jets, ...
  - Tree-level matrix elements and phase space at highest jet multis
- Can now be offloaded to GPU using novel Pepper event generator
- Next target: Subtracted real emission terms for NLO and NNLO
  - Excellent numerical stability for (N)NLO subtraction methods
- GPU event generators as provider for on-device ML training data

5 Backup: Additional material on Pepper

6 Backup: Sherpa optimisations

7 Backup: Projected CPU and energy consumption

8 Backup: Machine Learning

9 Backup: Chris' slides

Phase space generator: Chili [SciPost Phys. 15 (2023) 169]

- Differential phase space element for an n-particle final state

$$\mathrm{d}\Phi_n(a,b;1,\dots,n) = \left[\prod_{i=1}^{n} \frac{\mathrm{d}^3\vec{p}_i}{(2\pi)^3\,2E_i}\right]\,(2\pi)^4\delta^{(4)}\left(p_a+p_b-\sum_{i=1}^{n}p_i\right)\,.$$

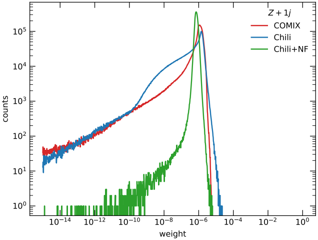
- Standard factorization formula

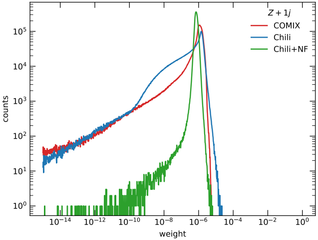
$$\mathrm{d}\Phi_n(a,b;1,\dots,n) =$$
$$\mathrm{d}\Phi_{n-m+1}(a,b;\{1,..,m\},m+1,\dots,n)\,\frac{\mathrm{d}s_{\{1,..,m\}}}{2\pi}\,\mathrm{d}\Phi_m(\{1,..,m\};1,\dots,m)\,.$$

- Use t-channel + adjustable number of s-channels
- Basic strategy: use single t-channel and only add s-channel resonances when required
  $\rightarrow$ easy to combine with Vegas
  $\rightarrow$ lean implementation allows for portability

Simple & portable phase space: applications & NIS

- What to do with Chili? (stand-alone library ↗)
  - Simplicity & portability → used in Pepper v1 as default
  - Speed → public parton-level Sherpa version for HPC ↗
- One/few channels + speed + portability → good fit for ML
- Example: Neural Importance Sampling
  Proof-of-principle [EB et al.] arXiv:2001.05478
  i-Flow+Sherpa [Isaacson et al.] arXiv:2001.10028 arXiv:2001.05486
  MadNIS [Heimel et al.] arXiv:2212.06172 arXiv:2311.01548
- Chili+MadNIS quick'n'dirty [EB et al.] arXiv:2302.10449





- Future: Sherpa v3.x, on-device NN training with Pepper

Portability: Aurora example

- Estimate “roughly 330 billion [leptonically decaying V+jets] events” required for HL-LHC [ATLAS] arXiv:2112.09588
  - “**Sherpa 2.2.11 setup would exceed budget by 16%**”
  - Assume 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 $\rightarrow$

This would be 8h on Aurora (with PVC)

The Color & Helicity Sum [EB, Giele, Höche, Isaacson, Max Knobbe, 2106.06507]

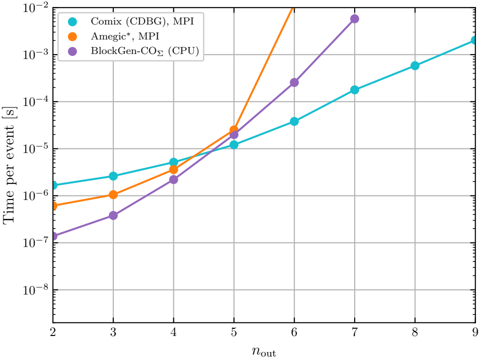
Benchmark performance for gluon-only Color-treatment:

- 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

⇒ **Algorithmic choice: Sum colors**

Helicity-treatment:

- Picture less clear, still allow multiple options



The Color & Helicity Sum [EB, Giele, Höche, Isaacson, Max Knobbe, 2106.06507]

Benchmark performance for gluon-only

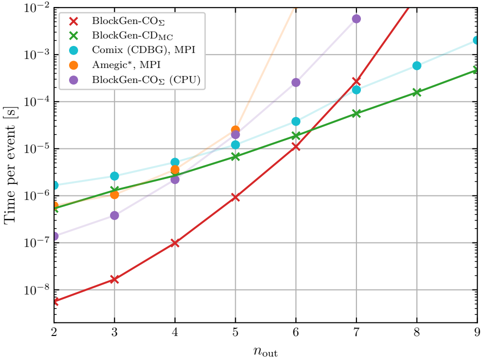
Color-treatment:

- 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

⇒ **Algorithmic choice: Sum colors**

Helicity-treatment:

- Picture less clear, still allow multiple options



The Color & Helicity Sum [EB, Giele, Höche, Isaacson, Max Knobbe, 2106.06507]

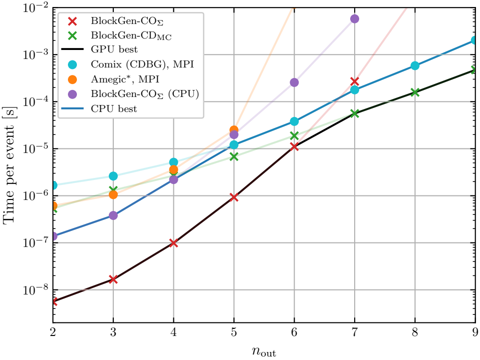
Benchmark performance for gluon-only Color-treatment:

- 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

⇒ **Algorithmic choice: Sum colors**

Helicity-treatment:

- Picture less clear, still allow multiple options

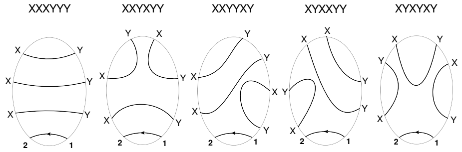


From Gluon-only to V+Jets

- 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 $\{A(1,2,\sigma), \sigma \in \mathrm{Dyck}\}$

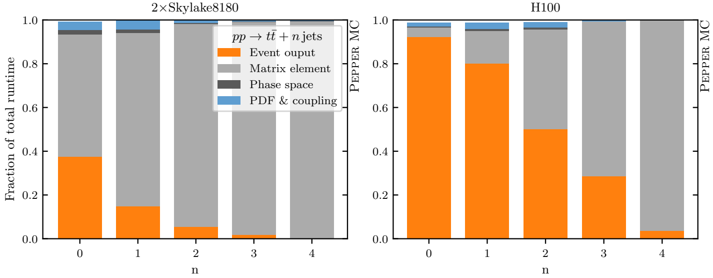
  [Melia] arXiv:1304.7809 arXiv:1312.0599 arXiv:1509.03297 [Johansson, Ochirov] arXiv:1507.00332

  $\rightarrow$ Allows to fix one fermion line, remaining permutations are given by Dyck-Words
  $\rightarrow$ Four particle Dyck Words: ()(), (())
  $\rightarrow$ Significantly fewer amplitudes to compute
- Include EW particles after QCD basis has been set up



arXiv:1304.7809

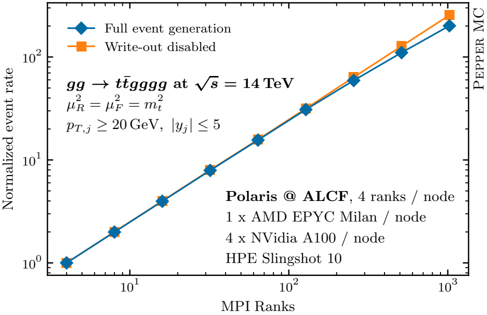
Timing details



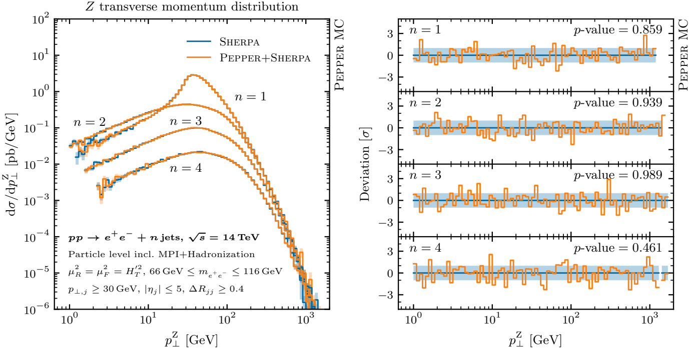
- Lower multiplicities are limited by write-out speed
  $\to$ No more need for computing improvements, but faster I/O
- Computing becomes relevant component only for large multiplicities

Scalability

- Test scalability for up to 1024×A100's
- Equivalent technology to [2309.13154]
  → established scaling to up to 16k threads at NERSC
- Scaling problems might not show up for a couple of nodes or low data volume
  → important benchmark



Validation + Pipeline into existing tools

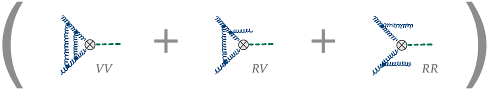


- Validated against Sherpa
  → for $V + j$, $t\bar{t}$+j, single multiplicity and multi-jet merged
- Writeout of HDF5 files, processable via Sherpa & Pythia

NNLO

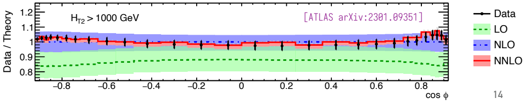
Slide by Alexander Huss
@ CERN generator & code acceleration workshop Nov' 2023

SUBTRACTIONS – NNLO



- typical runtime for $2 \to 2$ processes: $\mathcal{O}$(100k) CPU core hours
  - V+jet, di-jet, … ⇝ VV:RV:RR ∼ 1:20:100 (CPU hours)
- an extreme $2 \to 3$ example: $\mathcal{O}$(100M) CPU core hours
  - tri-jet ⇝ VV:RV:RR ∼ 1:100:200 (CPU hours)

$$\frac{1}{\sigma}\,\frac{\mathrm{d}\Sigma}{\mathrm{d}\cos\phi}$$



14

5 Backup: Additional material on Pepper

6 Backup: Sherpa optimisations

7 Backup: Projected CPU and energy consumption

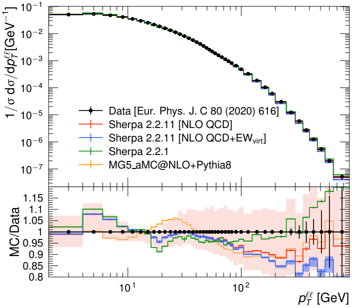
8 Backup: Machine Learning

9 Backup: Chris' slides

On-the-fly uncertainty calculation

- On-the-fly param./th./algo. uncertainties encoded by rel. event weights
- Easy for prefactors, but even changing the underlying probability distribution of accept/reject algorithms can be accounted for by a single weight factor
  [Höche,Schumann,Siegert] arXiv:0912.3501, [Giele,Kosower,Skands] arXiv:1102.2126, [EB,Schönherr,Schumann] arXiv:1606.08753, [Bellm et al.] arXiv:1605.08256, [Mrenna,Skands] arXiv:1605.08352

- Standard for Pythia, Herwig, Sherpa samples in ATLAS/CMS production
- E.g. alternative event weights for 7-point $\mu_{F,R}$ scale variations, PDF/α_S variations in ME, shower and, newly, hadronisation
  [Bierlich et al.] arXiv:2308.13459
- New in Sherpa 3: merging cut variations



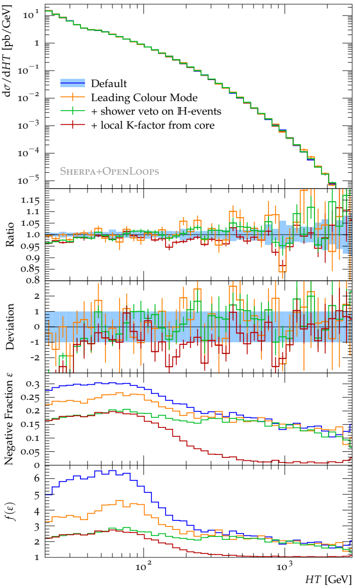
⇝ Uncertainty estimates at a fraction of the cost

Negative event weights

- Negative-weight event fraction ϵ reduces the statistical power of a simulated sample by

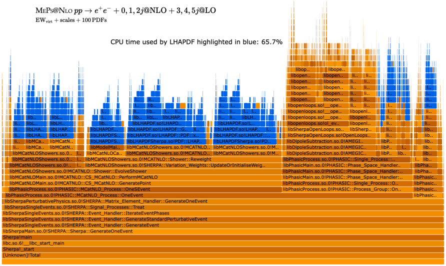
$$f(\epsilon) = (1 - 2\epsilon)^{-2}$$

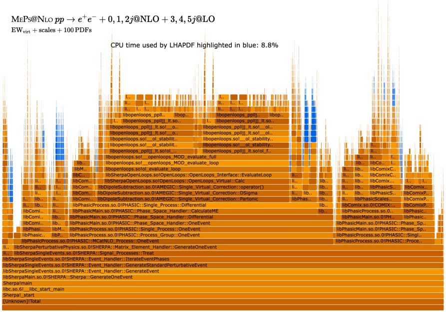
- Negative events still need expensive detector simulation
- Mitigations and ideas available but no complete solution yet [Danziger Höche Siegert] arXiv:2110.15211 [ATLAS] arXiv:2112.09588 [Frederix et al.] arXiv:2002.12716 arXiv:2310.04160 [Maier et al.] arXiv:2303.15246



Pilot runs

- Low efficiencies can be mitigated by pilot runs:
  1. run minimal pilot run to only calculate acceptance probability
  2. if a proposal is accepted, revert RNG state and run again with full shebang
- applicable to any accept/reject algorithm with low efficiencies
- e.g. 5× speed-up in Sherpa simulation:

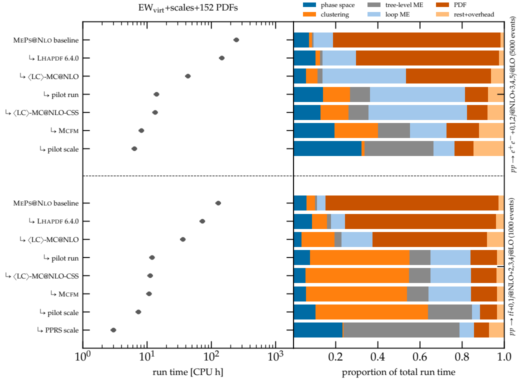


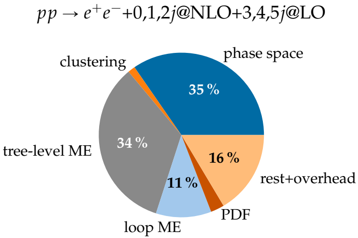


[EB et al.] arXiv:2209.00843

Pilot runs: part of significant performance improvements

Case study: ATLAS baseline configuration





- CPU consumption overall improved by factors of $\times$**39** and $\times$**43** for V+jets and $t\bar{t}$+jets [EB et al.] arXiv:2209.00843
- After optimisation, more than two thirds of CPU time spent in phase space sampling and (tree-level) matrix elements

5 Backup: Additional material on Pepper

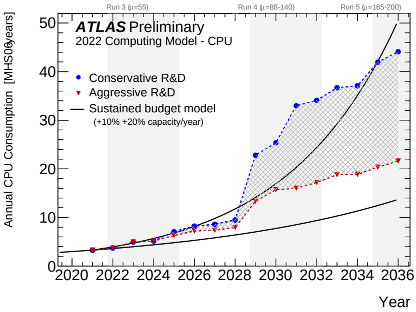
6 Backup: Sherpa optimisations

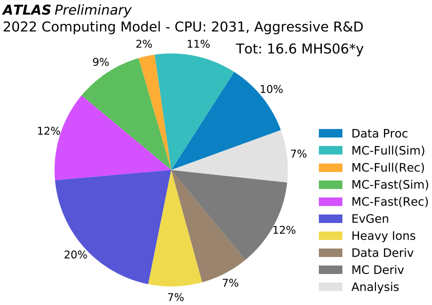
7 Backup: Projected CPU and energy consumption

8 Backup: Machine Learning

9 Backup: Chris' slides

LHC projected CPU consumption for event generation





- ATLAS: 14 % CPU for event generation in 2022
- expect ca. 20 % during HL-LHC ("Aggressive R&D" scenario 2031)
  - High statistics at HL-LHC & excellent detector performance

    → Need for precise & efficient event generator simulations

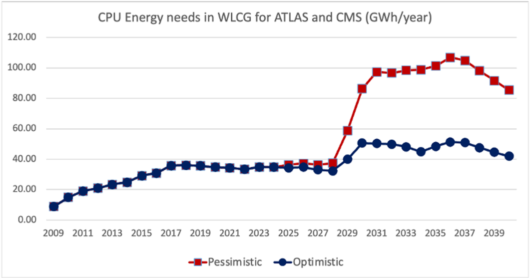
    → Poor performance can limit experimental success

    [HSF Physics Event Generator WG] arXiv:2004.13687, arXiv:2109.14938

What does this mean in terms of energy consumption?

[Britton, Campana, Panzer-Stradel] CHEP 2023

- "WLCG data centres' power consumption [. . .] generally driven by the CPU needs [. . .], at CERN, 70% [. . .]"



- difficult to convert to CO_2 equivalents, as the conversion factor is strongly country & season dependent
- 2031: 15k–30k households (each 3.5 MWh/year)
  $\rightsquigarrow$ 2k–4k for event generation back-of-the-envelope

5 Backup: Additional material on Pepper

6 Backup: Sherpa optimisations

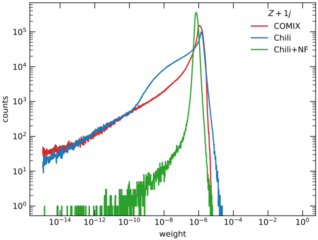
7 Backup: Projected CPU and energy consumption

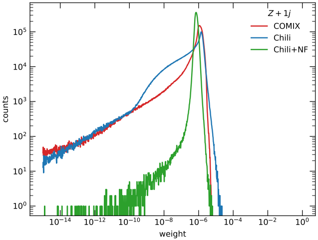
8 Backup: Machine Learning

9 Backup: Chris' slides

Phase space sampling/integration with Normalising Flows

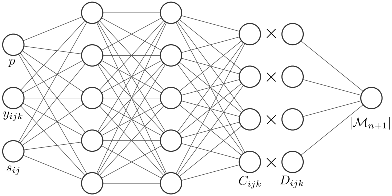
- Use NN trained bijective phase space mappings for improved sampling/integration of your integrand
- Example: Neural Importance Sampling
  Proof-of-principle [EB et al.] arXiv:2001.05478
  i-Flow+Sherpa [Isaacson et al.] arXiv:2001.10028 arXiv:2001.05486
  MadNIS [Heimel et al.] arXiv:2212.06172 arXiv:2311.01548
- Might work best with simple phase space such as Chili, easy to train due to low number of channels and ported as part of Pepper to run on GPU [EB et al.] arXiv:2302.10449
- Chili+MadNIS quick'n'dirty [EB et al.] arXiv:2302.10449

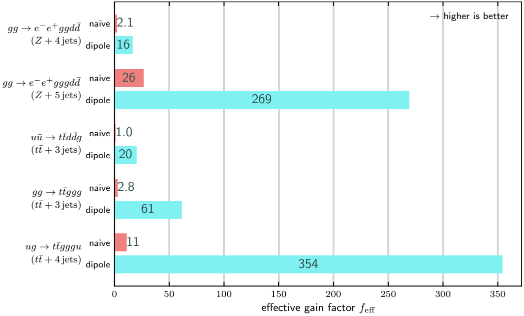




More ML: Surrogate Unweighting

- Replace $|\mathcal{M}|^2$ with fast ML surrogate
  Daniel Maître's and Simon Badger's talks @ CERN generator & code acceleration workshop Nov' 2023, LC surrogate: [Frederix, Vitos] arXiv:2409.12128
- Use second unweighting step to correct to exact $|\mathcal{M}|^2$
  [Danziger et al.] arXiv:2109.11964
- Train linear coefficients C_{ijk} of dipole terms D_{ijk}
  [Janßen et al.] arXiv:2301.13562





5 Backup: Additional material on Pepper

6 Backup: Sherpa optimisations

7 Backup: Projected CPU and energy consumption

8 Backup: Machine Learning

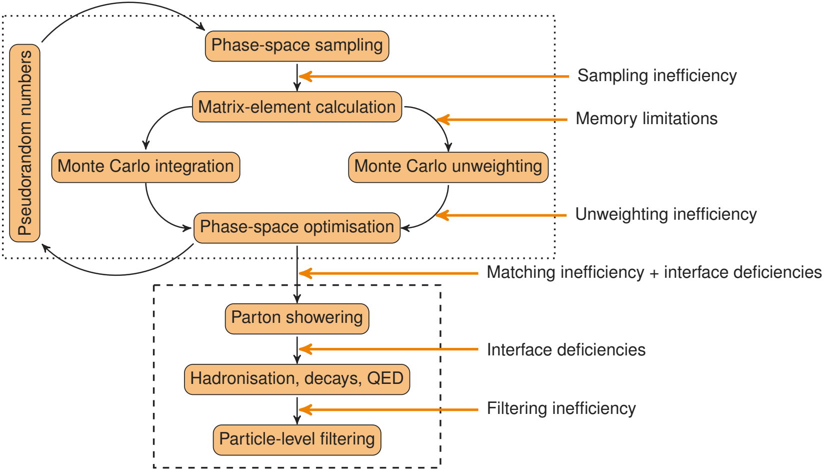
9 Backup: Chris' slides

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×

Other bottlenecks



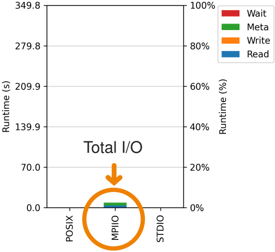
➜ Lack of active development on infrastructure tools (LHE, HepMC, ...) set to become a major bottleneck going forward

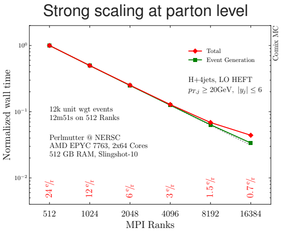
Introducing LHEH5

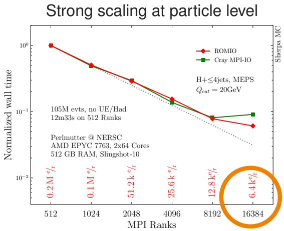
- established LHEF format is based on XML
  - flexible enough to add any desired feature
  - poses a challenge for I/O operations at scale
- new efficient LHE-like data format based on HDF5+HighFive proposed in [PRD 109 (2024) 1]

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





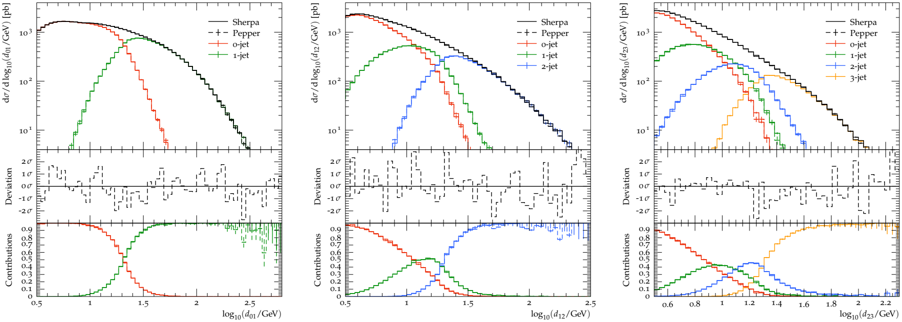


105M events / min

- 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

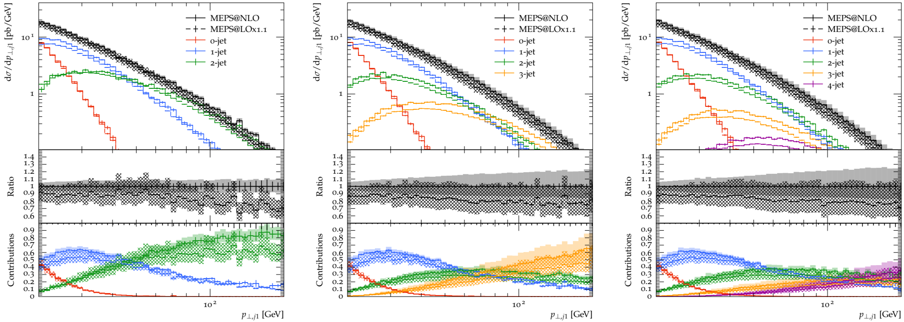
Comparison of parton-level event generators

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



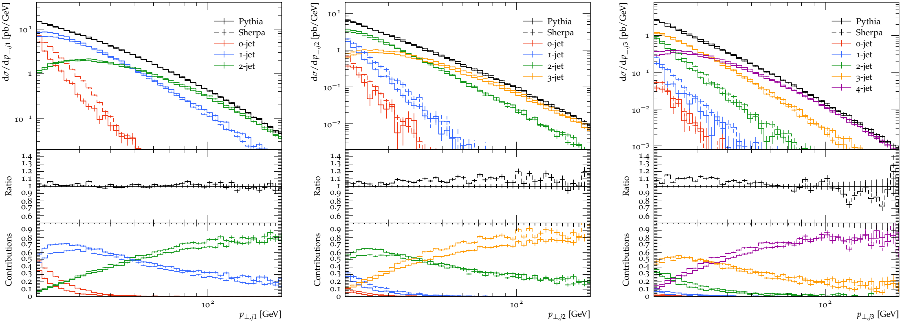
Improved modelling through high-multiplicity final states

- 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



More robust uncertainty estimates

- LHEH5 enables efficient substitution of various parts in the event generation chain
  - already supported by Pepper, 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 (✗)
- 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 (✗)