Generator issues and expectations, the experience of the experiments: ATLAS

Tetiana Moskalets (Albert-Ludwigs-Universität Freiburg)
on behalf of ATLAS Physics Modelling Group

Event generators' and N(n)LO codes' acceleration workshop
13-14 November 2023, CERN

ATLAS EXPERIMENT

universität freiburg

Emmy Noether-Programm

DFG Deutsche Forschungsgemeinschaft

Outline

- Motivation for computing performance improvements
- Current CPU bottlenecks
- Resource bookkeeping
- Addressing the CPU issue:
  - Improvements in the per-event CPU efficiency
  - Phase-space biasing
  - Negative weights: latest and expected improvements
  - Parallelisation
  - Usage of GPUs
  - Sharing of samples between the experiments
- Usage of the newer generator versions
- Conclusions

- This talk **is** about:
  - technical side of the MC generation
- This talk is **not** about:
  - physics issues

Why improve the computing performance of the MC generators?

- **Event generators are essential software components of the data processing of the LHC experiments, and large consumers of their computing resources.**

- Study ongoing within ATLAS on estimating the resources needs during the HL-LHC phase
  - using the ongoing Run-3 MC production campaigns as a model to assess how much CPU will be needed for the HL-LHC
  - plan to publish soon

- Previous estimations from the HL-LHC Computing CDR (CERN-LHCC-2020-015)

Resource usage in 2028 (LHCC common scenario)	CPU [MHS06· y]	Disk [PB]	Tape Tier-1 [PB]	Tape Tier-0 [PB]
Baseline	83	3510	2370	925
Conservative R&D	47	2180	2000	924
Aggressive R&D	20	1030	1760	924
Sust. budget model +20%	16	930	1240	280
Sust. budget model +10%	9	510	674	150

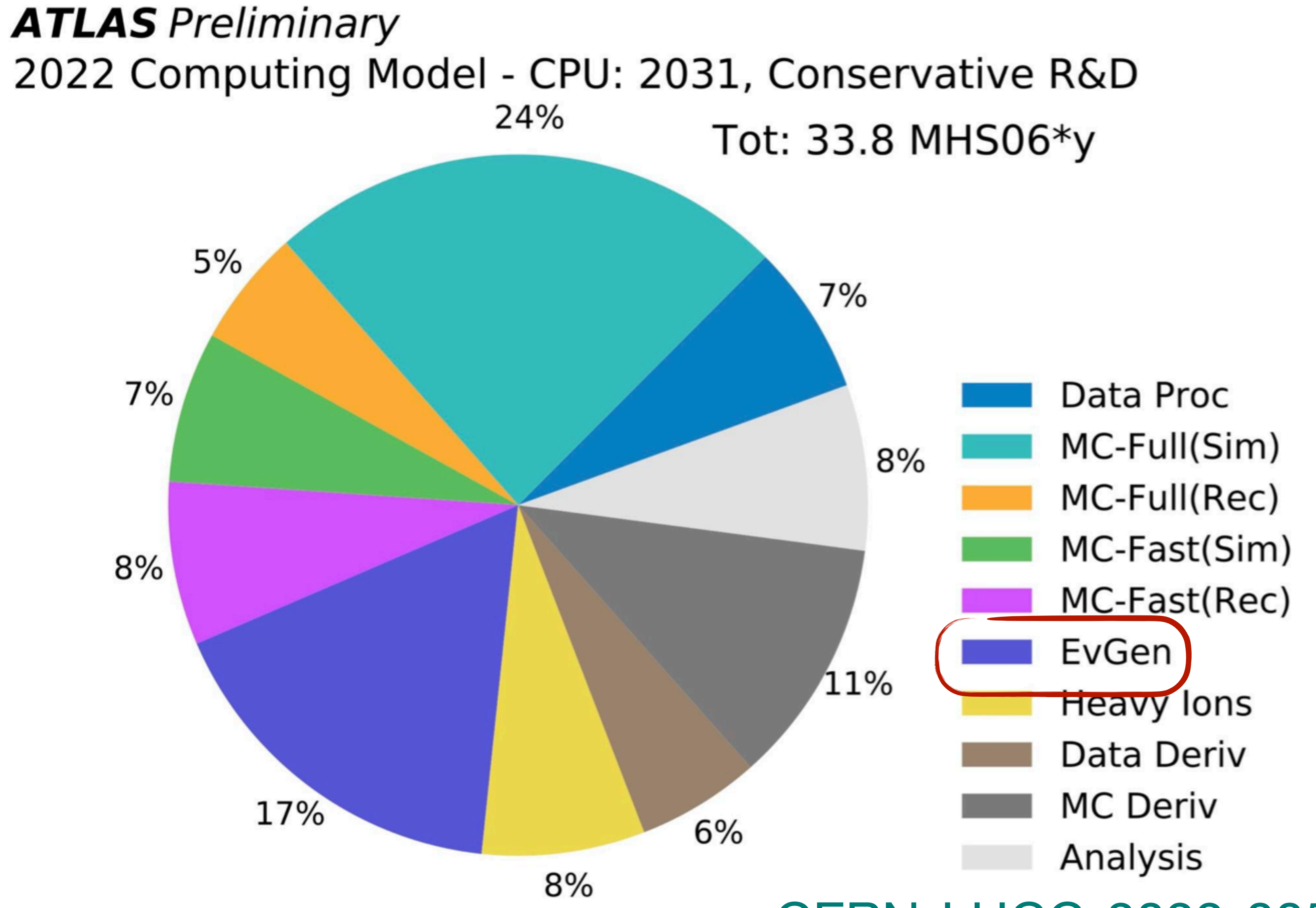
Table 11: Resource estimates under the jointly ATLAS and CMS assumptions (as from table 10) during 2028 for the three ATLAS computing scenarios.

Current CPU bottlenecks

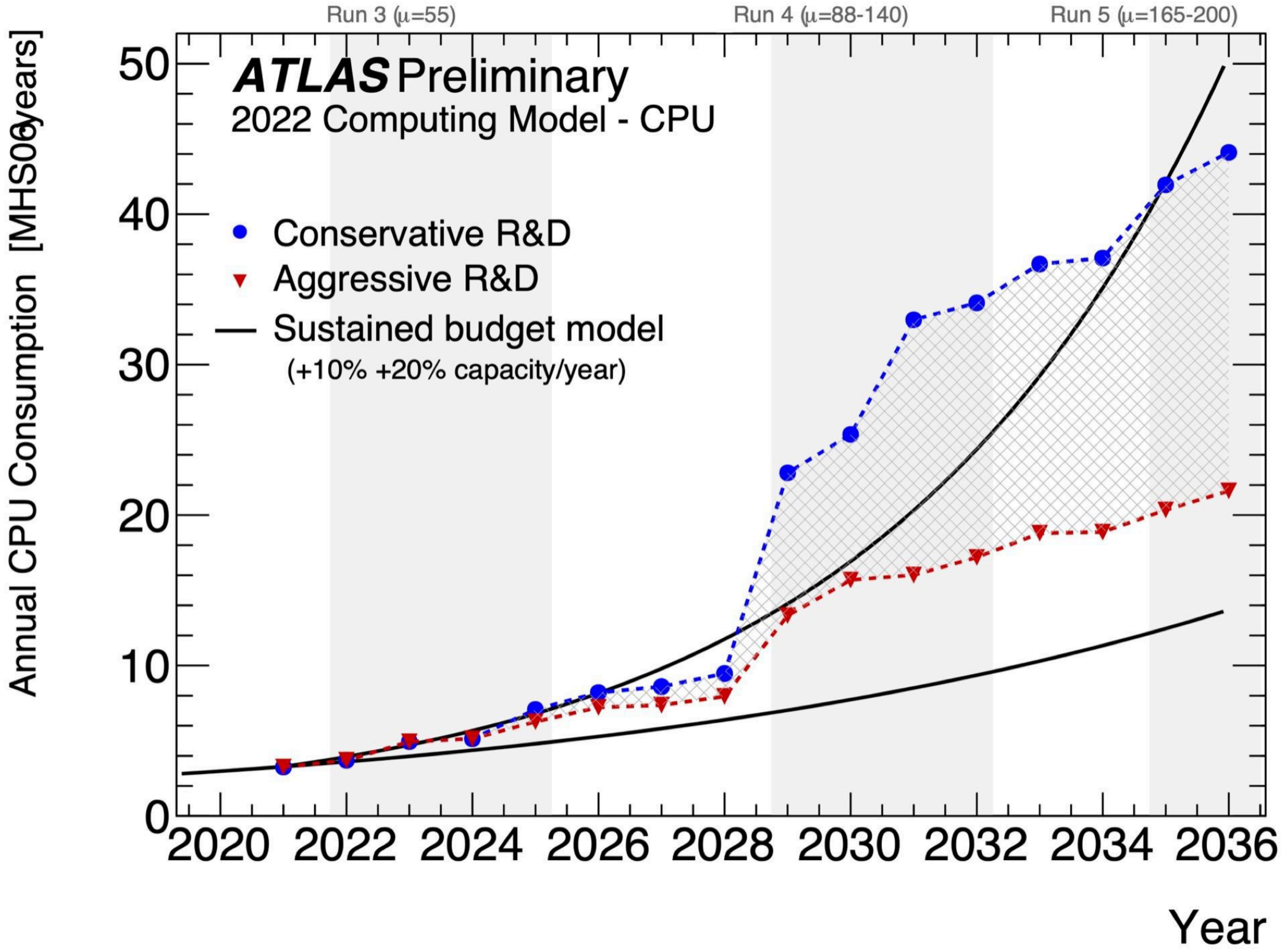
- **Event generation production takes a significant part of the CPU**
  - we used 14% CPU on event generation last year
  - expect ~20% during the HL-LHC phase

- Projected evolution of computing usage from 2020 until 2036, under the conservative (blue) and aggressive (red) R&D scenarios
  - estimations from 2022 (CERN-LHCC-2022-005)

- **Current and planned approaches to improve the CPU efficiency**
  - More efficient event generation (reducing negative weights fraction)
  - Accelerating the calculations (GPUs/parallelisation)
  - Statistical enhancement
  - Moving from alternative setups to internal weights
  - …and various generator-specific improvements of the per-event CPU time

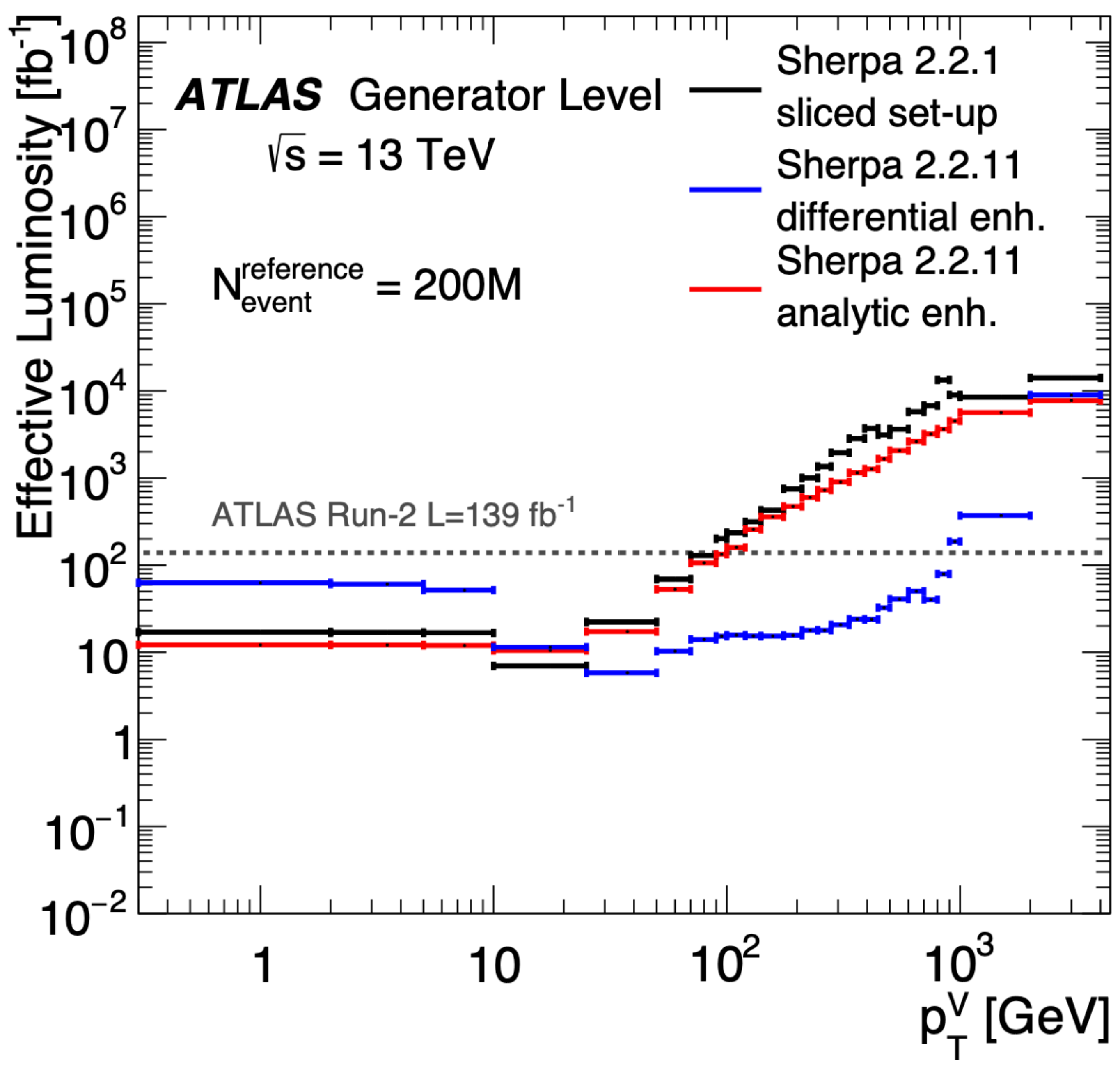


CERN-LHCC-2022-005



Resource bookkeeping

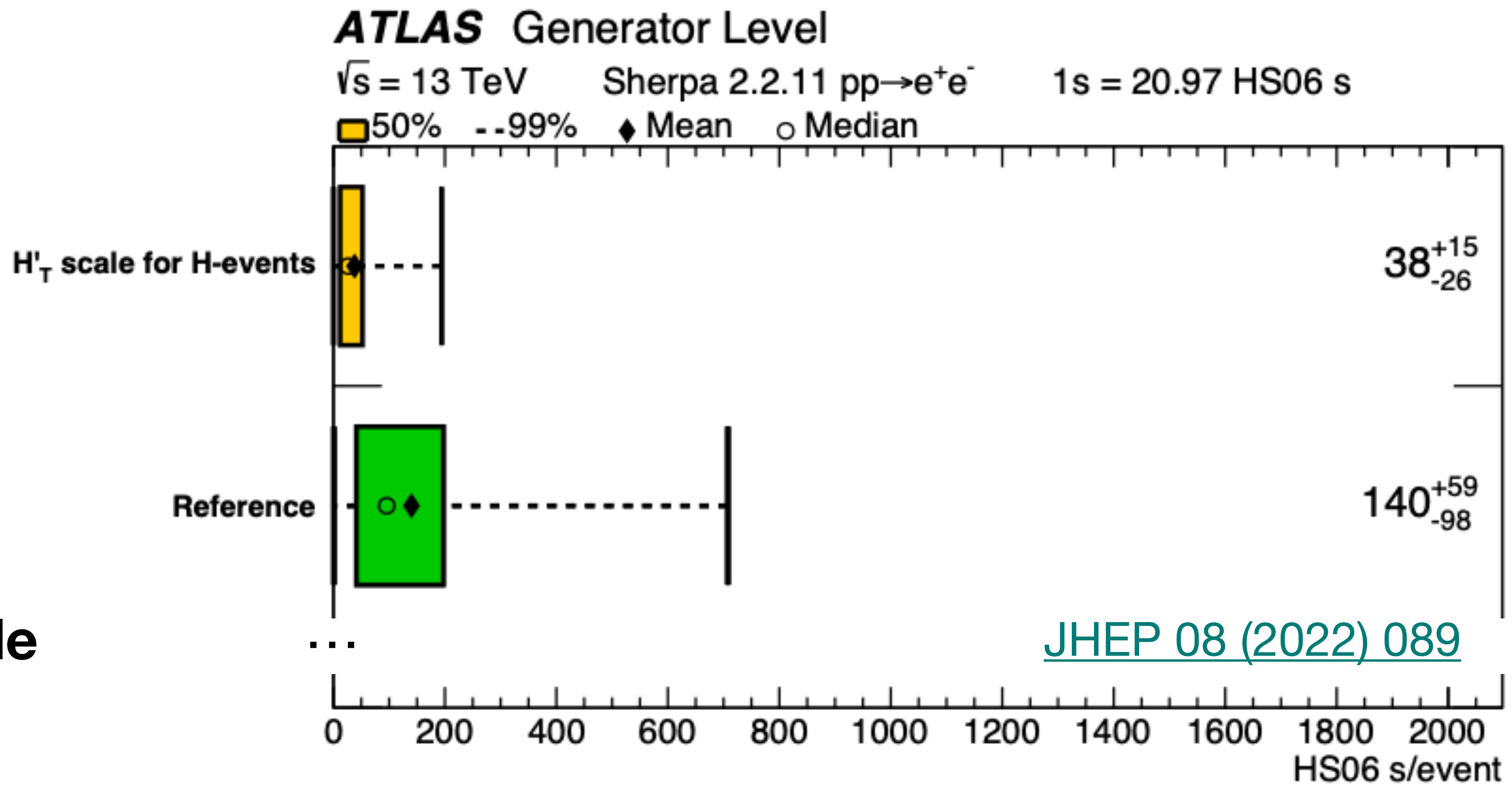
- Need to do accounting of the resources required to produce different kinds of processes
  - we have the numbers for the latest Run-3 MC production taken from the grid → can do the HL-LHC projection
  - analysed the most commonly used Standard Model processes & generators
- **Largest fraction of EvGen CPU time is taken by generation of multi-leg MC predictions**
  - namely, multijet merged Sherpa V+jets
  - current generator version allowed to reduce the CPU consumption by a factor of 3-4 w.r.t. the previous ones (see next slide)
- **Other time-consuming samples:**
  - dijet (Sherpa and Powheg)
  - Powheg NLO inclusive $t\bar{t}$
    - calculation itself is fast
    - but need huge samples for nominal + several systematic variations
- Still need to factor in **negative weights** to the overall picture
  - they cause a ~20-30% increase in the overall budget
- For the discussion: does the generated effective luminosity of a sample need to exceed the data set for the full inclusive phase-space?



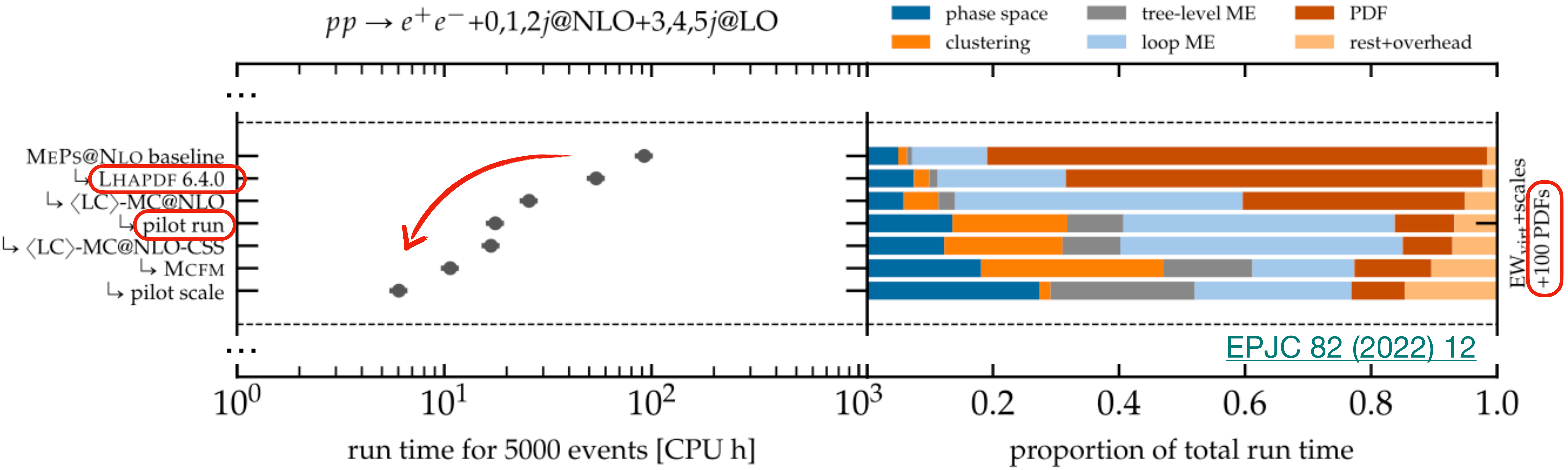
- **Plans to make a public note on these numbers including HL-LHC projections**
  - previous bookkeeping exercise was presented in Josh's slides

Recent improvements in CPU efficiency in Sherpa

- Sherpa 2.2.1→2.2.11: **~3x improvement** in per-event CPU time for the V+jets events due to switching to H'_T scale for $\mathbb{H}$-events
  - ➡ shown in an ATLAS paper JHEP 08 (2022) 089



- Simplified pilot runs and fast PDFs in Sherpa 2.2.12 (EPJC 82 (2022) 12): **3-4x speed-up** if no variations are calculated, up to **an order of magnitude more** if PDF variations are included
  - demonstrated for Z+jets and $t\bar{t}$+jets samples



Phase-space strategy	Mean [s/event]	Mean [HS06 s/event]
Sherpa 2.2.11 configuration		
$\left(\frac{\max(H_T, p_T^V)}{20}\right)^2$ analytic enhancement	17.9 ± 0.2	375 ± 4
Sherpa 2.2.1 configuration		
$0 < \max(H_T, p_T^V) < 70$ GeV	4.7 ± 0.5	99 ± 11
$70 < \max(H_T, p_T^V) < 140$ GeV	34.6 ± 2.3	725 ± 48
$140 < \max(H_T, p_T^V) < 280$ GeV	36.8 ± 1.2	772 ± 25
$280 < \max(H_T, p_T^V) < 500$ GeV	53.7 ± 2.2	1126 ± 46
$500 < \max(H_T, p_T^V) < 1000$ GeV	67.6 ± 3.0	1418 ± 63
$\max(H_T, p_T^V) > 1000$ GeV	108.4 ± 5.7	2273 ± 120

Event filtering

- For a lot of analyses we need to provide enough statistics for the processes in specific kinematic regions or with special final states → use filters:
  - E_T^{miss}/H_T-filtering in $t\bar{t}$ samples
  - heavy-flavour filtering in $t\bar{t}$ and V+jets samples
  - filtering for fake backgrounds, e.g. muon fakes

- Filter efficiencies are often small → need to produce huge samples

- Few examples of filters: ⟶

Sample	Filter	Filter efficiency, %
Powheg+Pythia8 ttbar	E_T^{miss} 200–300	0.8
	H_T 1k–1.5k	0.4
	bb	0.9
Sherpa 2.2.11 Z(ll)+jets	b	2.5
	c	13
Sherpa 2.2.11 W(lv)+jets	b	0.9
	c	15

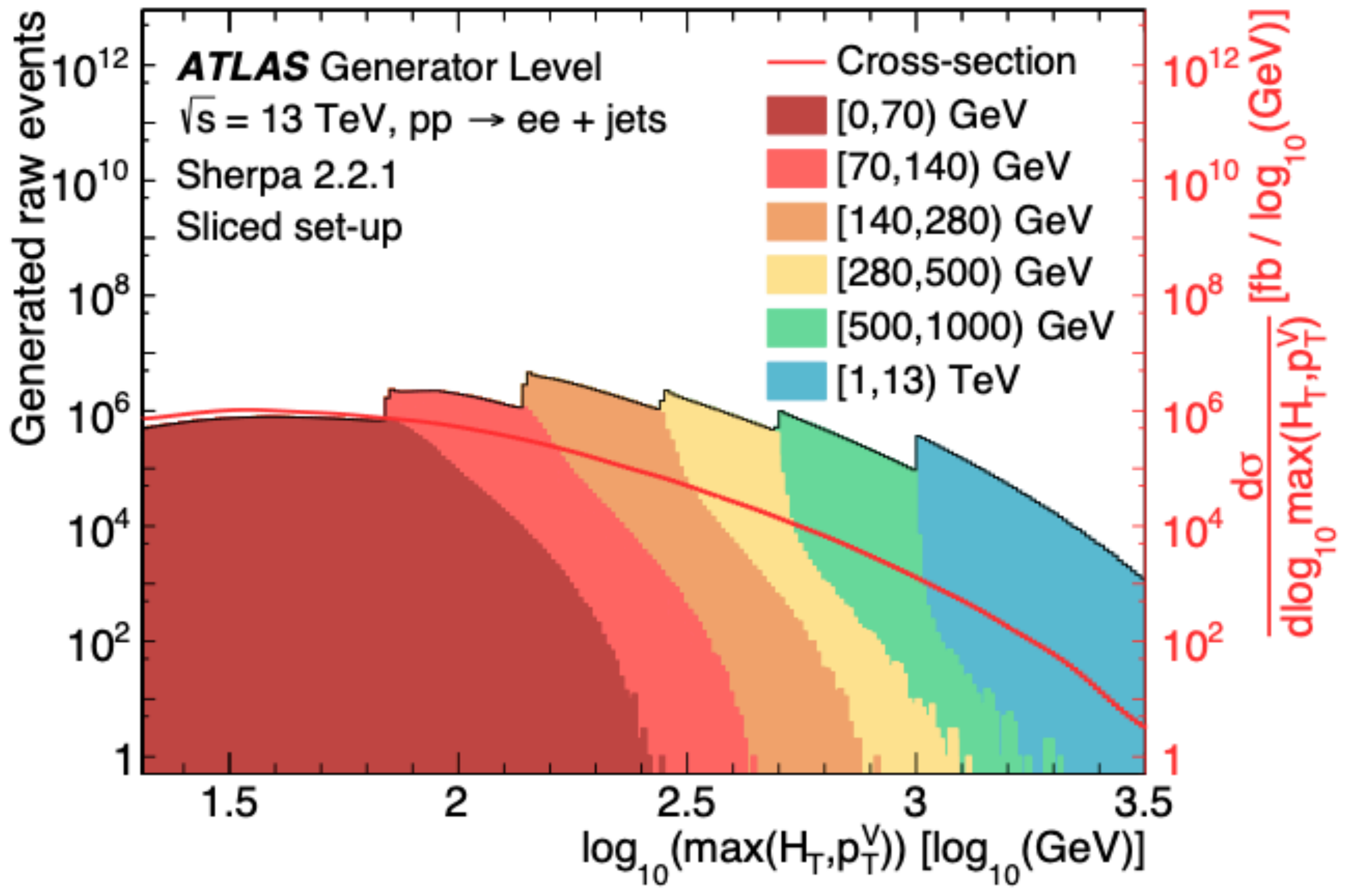
- Having flavour enhancement instead of flavour filtering would help a lot in saving the CPU resources

Phase-space biasing

- **Instead of filtering one can additionally populate remote phase-space regions to ensure enough statistical precision there**

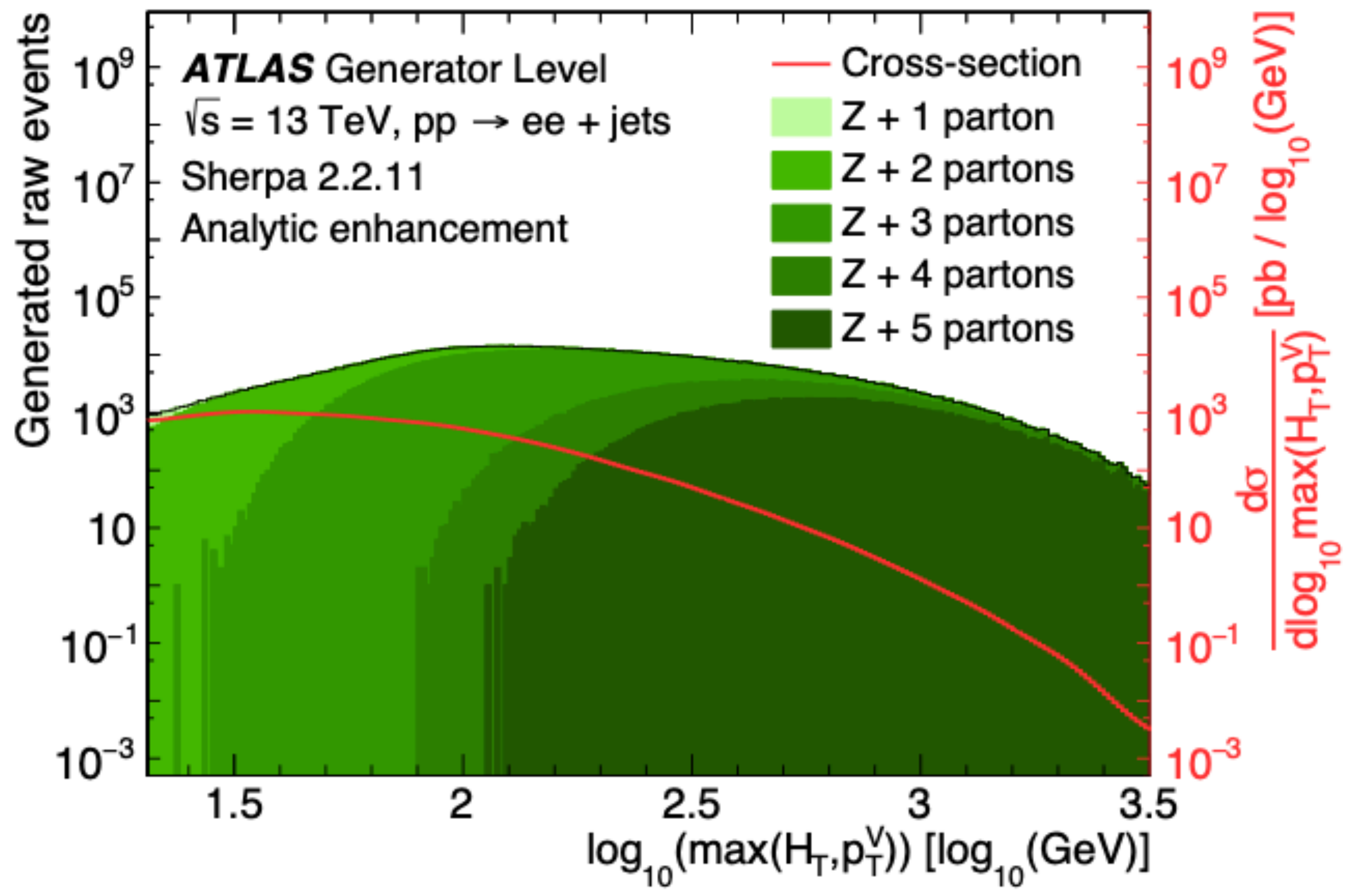
- Performance of the enhancement techniques available in Sherpa 2.2.1 and Sherpa 2.2.11 for the configurations used in ATLAS was compared in JHEP 08 (2022) 089

- **Sherpa 2.2.1**: discrete (sliced) enhancement depending on $\max(H_{\mathrm{T}}, p_{\mathrm{T}}^{V})$



- **Sherpa 2.2.11**: continuous enhancement
  - differential (inverse differential cross section)
  - analytic (function of a set of observables)

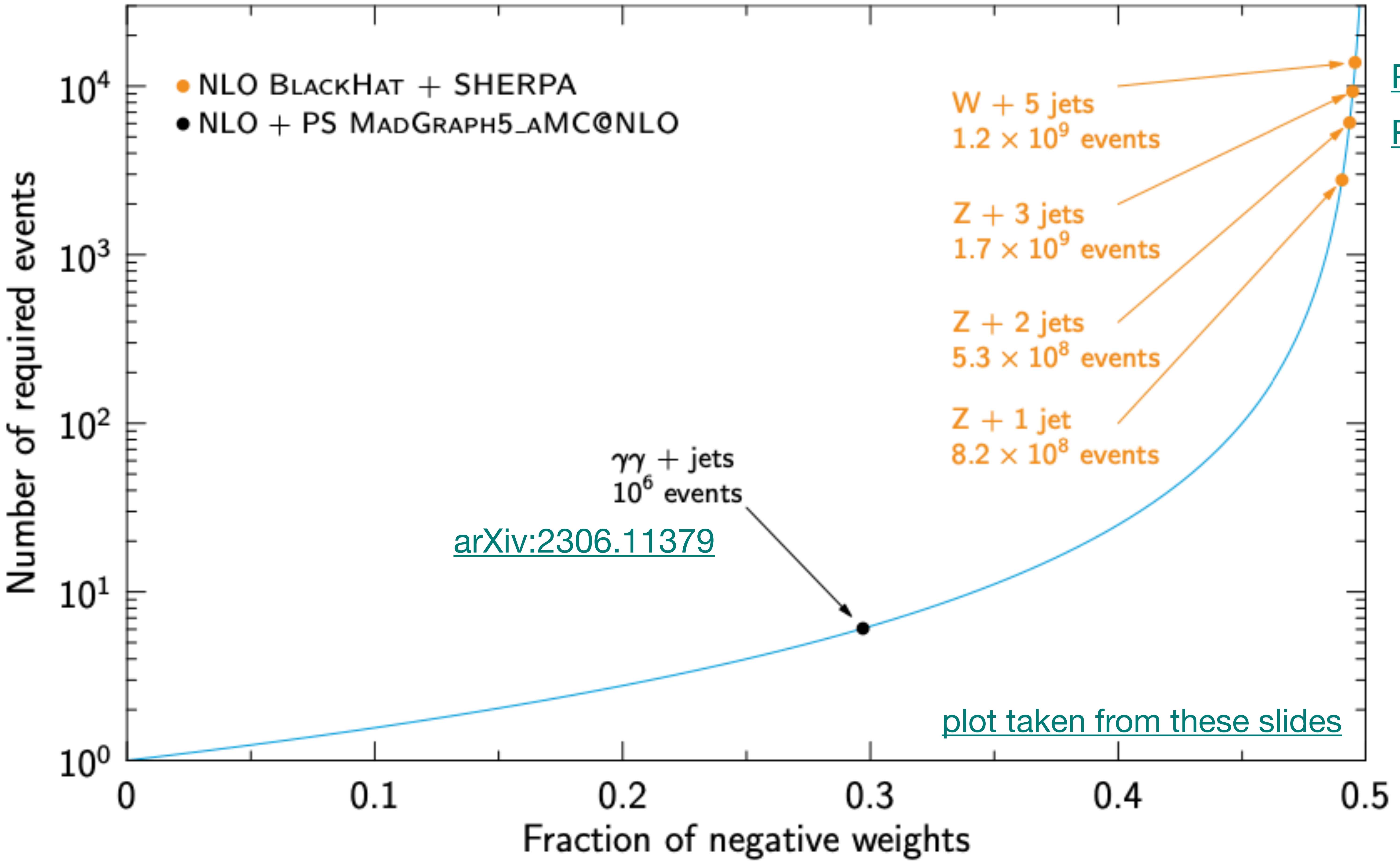
→ better statistical precision!



- For the photon processes in Sherpa enhancement of photon radiation and phase-space biasing are also being studied in ATLAS

Negative weights

- **Statistical power of a sample with negative weight fraction ϵ is reduced by** $1/(1-2\epsilon)^2$
  - ϵ = 25% → 4x larger sample is needed for the same statistical power
  - ϵ > 30% → sample is not really usable



PRD 88 (2013) 014025

PRD 97 (2018) 096010

Negative weights

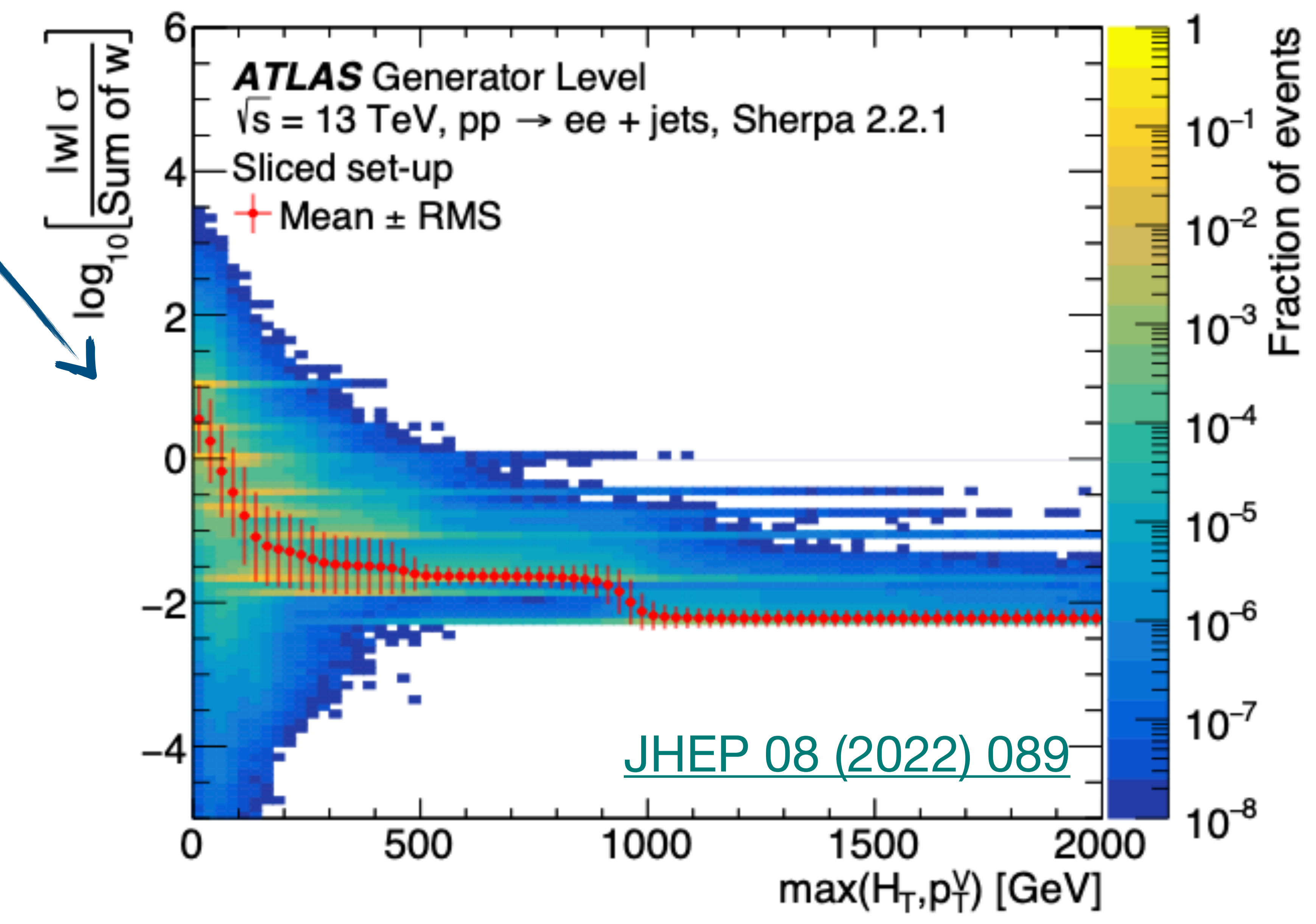
- **Long-standing problem for some generators and processes**
  - Herwig7 Matchbox
    - $t\bar{t}$: 20-40% negative weights, increases with the number of jets
    - dijets @ NLO in 5FS: 30-40% negative weights because of the 5FS

    ← practically excludes the possibility of using Matchbox
  - aMC@NLO → up to 40% negative weights
    - $t\bar{t}$, W+jets: 20%
    - $b\bar{b}H$: 40%

    ← in ATLAS, Powheg+Pythia is a preferable nominal setup for many top processes because of this
  - Sherpa (~always used as a nominal for V+jets/diboson in ATLAS)
    - in Sherpa 2.2.1 Z+jets have 20-30% negative weights, depending on p_{T}^{V}
  - Powheg is also sometimes problematic
    - e.g. 30-50% negative weights in $t\bar{t}b\bar{b}$, depending on the folding settings

Name	foldcsi	foldy	foldphi	neg. fraction (nominal)	neg. frac. (scale down variation)
551	5	5	1	9.7%	47.1%
552	5	5	2	9.1%	46.2%
555	5	5	5	5.2%	33.1%
1055	10	5	5	4.1%	32.7%

ATL-PHYS-PUB-2022-006



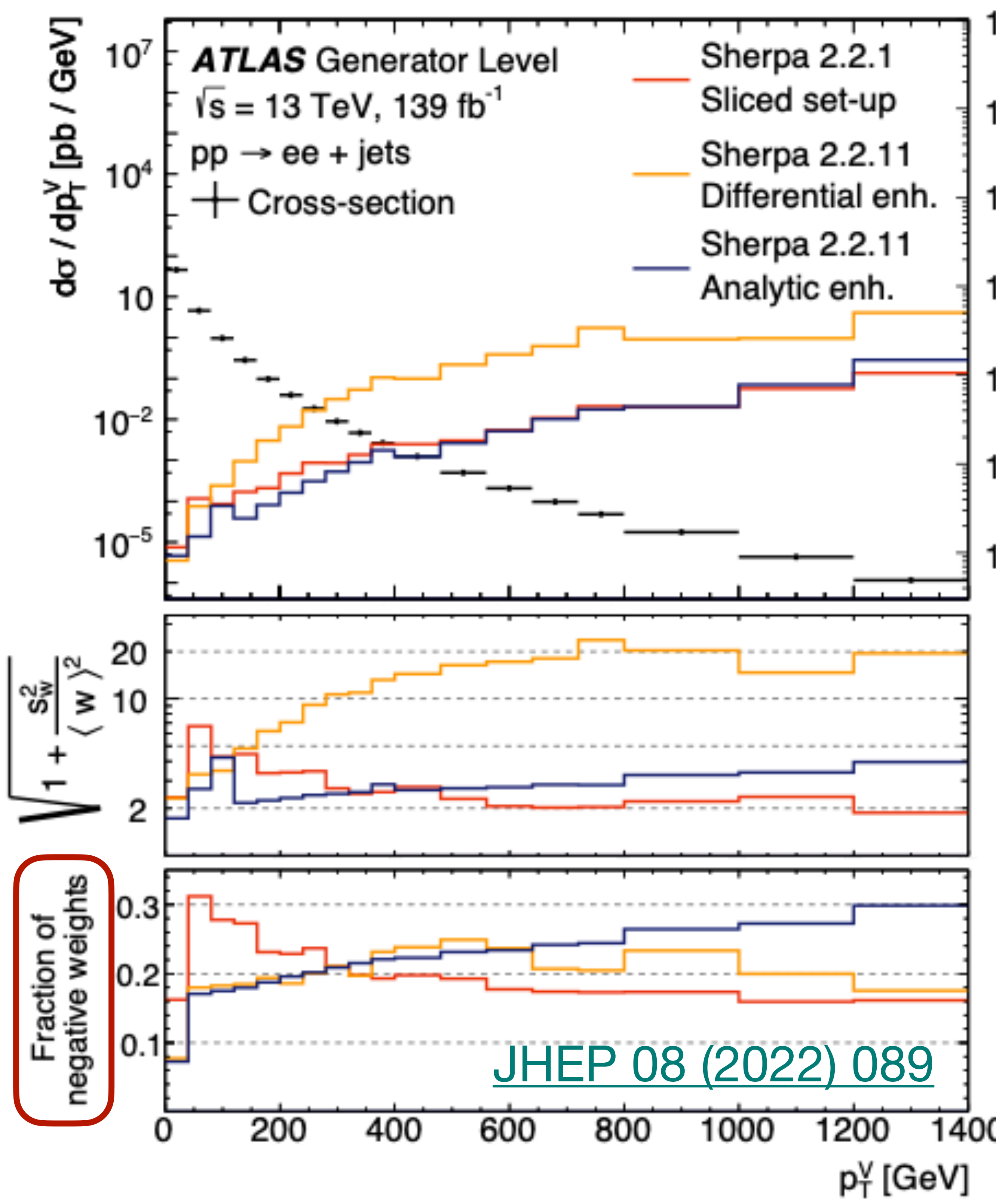
JHEP 08 (2022) 089

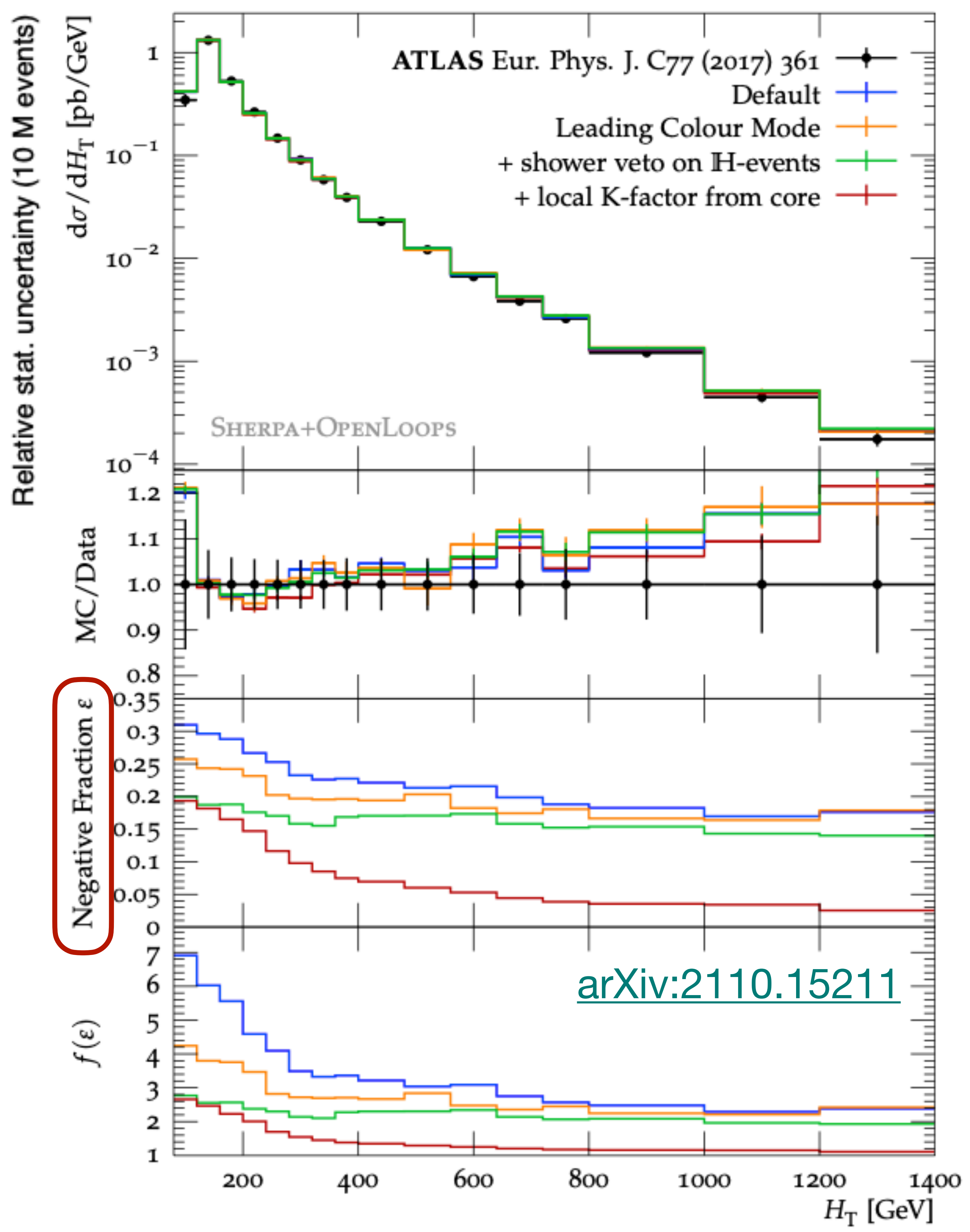
Negative weights: how to reduce?

Expected improvements from:

- **MC@NLO-Delta** arXiv:2002.12716
  - testing within ATLAS started
- **Herwig7**: alternative matching scheme KrkNLO APPB 48 (2017) 1121
  - looking forward to get it in Herwig7.3 (see this talk)
- **Generator-unspecific:**
  - Cell resampling arXiv:2303.15246 (testing within ATLAS planned)

- **Some progress in Sherpa:**
  - Reduced in 2.2.11 compared to 2.2.1 thanks to adjustments in MC@NLO matching and NLO/LO K-factor calculation (JHEP 08 (2022) 089)
  - More advancements in reducing the negative weights fraction: arXiv:2110.15211





Parallelisation: more optimisation needed…

- **Problem: integration of processes typically does not scale well for complex processes**
  - lack or reliability in sub-jobs
  - often does not scale well with number of CPUs, because one/few jobs need much much longer than all others

- **Example**: typical profile of a MadGraph FxFx Z(ee)+0-3j@NLO integration job (using a 64-Core/128 thread CPU machine)
  - 1 day: amplitude generation → 1 core
  - ~0.5 h: code compilation → 64 cores (needs lots available open file handles)
  - 6 days: Setting up grids → 64 cores working through 5336 subjobs
  - 8 days: Setting up grids draining and final few jobs completing, 1-2 cores
    - $Z(\tau\tau)$ job was stuck ~3 months in this stage
  - 22 days: Computing upper envelope → 64 cores working through 5336 subjobs
  - 1 day: Computing upper envelope slowly draining → 1-2 cores
  - < 1 h: finish up tasks

Setup	Days to finish integration
Z → ee peak	31
Z → μμ peak	27
Z → $\tau\tau$ peak	**48**
Z → $\nu\nu$ (high pT)	*15*
Z → ee low mass	37
Z → μμ low mass	**76 + rescue**
Z → $\tau\tau$ low mass	**71 + rescue**
Z → ee high mass	22
Z → μμ high mass	27
W → eν high mass	*19*
W → μν high mass	*16*

- Memory is not an issue, but CPU usage is large and usage profile very uneven: some subjobs run seconds, others weeks

- Also, there is always a risk to get a random glitch in some check routine which could spoil the computation of one of the subprocess (after 2 months of computations)
- Or, one of the machines can decide to reboot itself

} ok, one can hack a bit, and rescue the failed jobs instead of starting from scratch...

Usage of GPUs: current status and expectations

- **Active collaboration already ongoing between some generator authors and HSF Generators Working Group from CERN IT!**
- **MadGraph**
  - looks like there is big progress already (see the talk at CHEP23)
  - we are looking forward to have a user-friendly LO version soon (and we've been promised also the NLO version a bit later)
- **Sherpa**
  - has similarly "heavy" matrix element calculations to MadGraph, particularly at NLO
  - already has a GPU effort internally
- **Pythia8**
  - is already quite fast, Pythia8.3 did have a huge speedup vs. 8.2 (5-10x)
  - built-in MEs are not used much anymore, and are 2→1 or 2→2 LO, so have analytic ME samplings: no gain from GPU
  - ➡ GPUs for Pythia is not really in a priority right now
- **Some risk here**: one should ensure the GPU code can be understood and maintained by the generator teams themselves

---

- On the practical side, for now it looks a bit uncertain, there are still decisions to be made:
  - how soon we plan to put the MG4GPU workflow into production (if at all)
  - vector CPU also gives a lot more CPU efficiency on certain (existing) CPU hardware → would we prefer GPUs over this?
  - what if we have a large fraction of the HLT farm on GPUs?

Sharing of samples between ATLAS and CMS

- **Could save resources:** practically 50% CPU, if ATLAS and CMS use the same samples

- But:
  - not clear if it is reasonable to use *exactly the same setups for all the samples*
  - not that easily achievable due to different approaches to the modelling uncertainties estimation

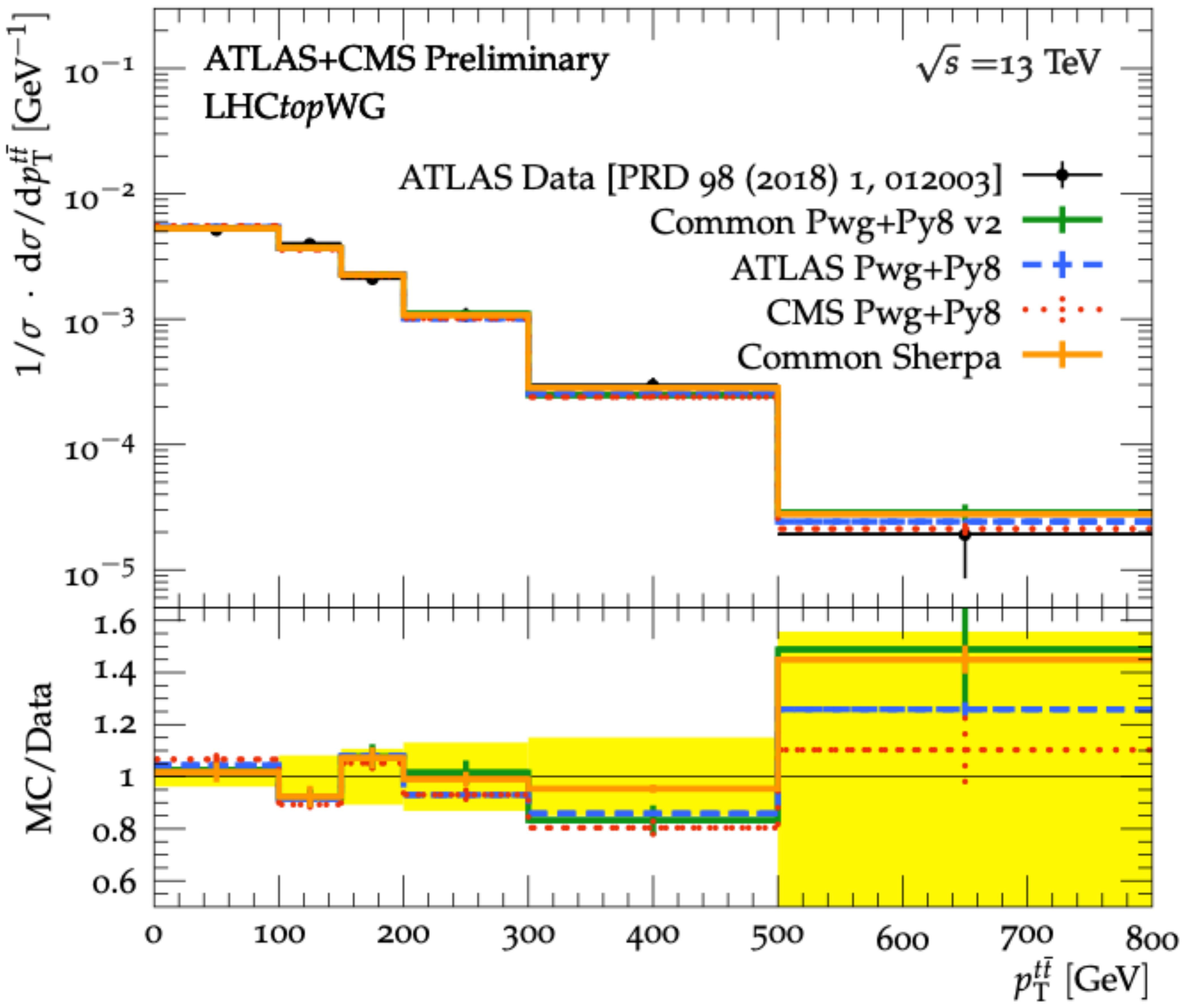
- Could at least have ~one common sample for each process, which one would use as nominal and other as a systematic sample / cross-check

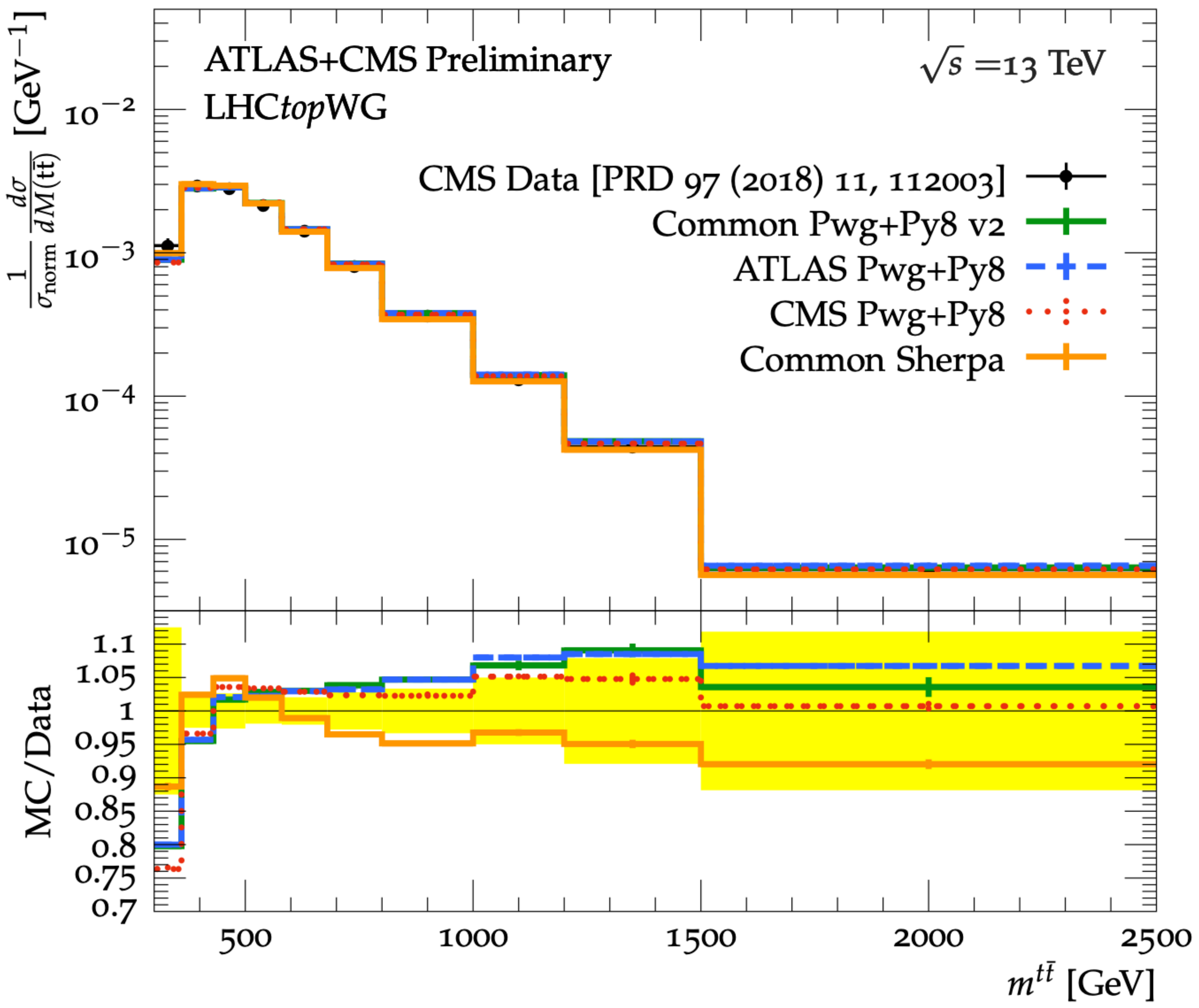
- Complication: output format varies between the experiments
  - common setups developed up to now are based on shared LHE events + shared Pythia parameters

- **Really beneficial would be to use a common particle level output format**
  - e.g. HD5?
  - common access to Rucio datasets would also be useful

Sharing of samples between ATLAS and CMS

- ✓ **First step: Improved Common $t\bar{t}$ Monte-Carlo Settings for ATLAS and CMS** ATL-PHYS-PUB-2023-016
  - for Powheg+Pythia8 and Sherpa
- ATLAS and CMS used to prefer different Pythia/Herwig tunes → would be good to reach some agreement
  - we would also like to hear opinions from the generators community side





New generator versions

- **All the new features and the new fancy generator versions take far too long to make them production-ready (at least, in ATLAS)…**
- We cannot just use the standalone new versions, need first to:
  - install the new version
  - (sometimes) update the Athena framework interface
  - validate the new version
  - adjust the setup/settings for our needs
  - ➡ can get stuck on either of these steps :(
    - for instance, due to lack of knowledge about all the details of the new features
- Experienced problems with:
  - in the past: bb4l, DIRE
  - now: VINCIA
- **What can be done on our/the authors side to improve this?**
  - we could collaborate more with the authors and start some testing before the official release (?)
  - would be useful to have one person from a generator group within the collaboration who knows a bit more in detail our software infrastructure

Conclusions

- HL-LHC projections show that ~1/5 of the CPU will be taken by the event generation
  - Multi-leg V+jets samples are the largest consumers of the overall CPU budget
- CPU efficiency is improving, e.g. in Sherpa
- Negative weights issue is still very relevant
  - Various techniques have been proposed for addressing it $\rightarrow$ test them all or choose one?
- Plans for speeding-up the matrix element generation using GPUs look promising
- Significant progress in establishing a common ATLAS+CMS setup for the $t\bar{t}$ sample
  - Let us try more processes?
- Close interaction between the generator authors and experts within the collaboration is a key for timely propagation of the new features into the actual MC samples
  - Lots of room for improvement here

➡ This was our biased ATLAS' view — we would be happy to hear the opinions from the MC community!

link to the public ATLAS MC-related results