Experiments view on the generators

25th MCNet meeting
27 April 2023

Dominic Hirschbühl (Wuppertal)

Introduction

- Significant discussion in the HSF Whitepaper on improving Event Generators
- ATLAS spent ~14% of our CPU (1100 HS06 years) in 2022 on event generation
  - Not including a very expensive recent run of NNLO calculations

- Share projected to rise slightly over the next 10 years to 17-20%
  - Projections are difficult without knowing what calculation orders will be available, what programs we will run, what techniques will be used, etc

- Our biggest samples in terms of CPU currently are:
  - Sherpa V+jets NLO multi-leg
  - Powheg NLO inclusive ttbar - nominal + 5 systematic variations!

- Extensive use of Powheg and Sherpa for SM, MadGraph5_aMC@NLO for BSM

Computing efficiency

Many ways we could improve our CPU efficiency

- Improvements in the calculation itself
  - Profiling of generator code
  - Usage of GPUs
  - Reducing negative events weight fraction
  - Optimised parallelisation
    - E.g. time to the last process's completion in MG5_aMC is a problem for multi-process running (in the pre-integration step)
- Improvements in the physics setup
  - Efficient slicing and Improving filter efficiencies
  - New phase-space sampling techniques that avoid biases
  - Integrating (all?) systematic uncertainties as weights

Most of this is only needed for multi-leg @NLO or NNLO+PS setups

Profiling of generator code

- The experiments have quite some experience with profiling software
  - → Can help identifying bottlenecks, sometimes in fixing them
- Need open-source code (esp. relevant for new / NNLO generators)
- Also need to be sure we profile the correct thing
  - Need common setups for ATLAS, CMS, and theory groups *for the future.*
  - Is Vincia/DIRE the future of showering? If so, that’s what we should profile.
  - Inclusion of PDFs (LHAPDF), systematic uncertainties, etc is vital
  - We’ve seen things like scale choices make an enormous difference
- Probably worth a bookkeeping exercise first to be sure we invest according to time spent in ATLAS+CMS
  (ongoing work for Run 3 in preparation for HL-LHC)

Usage of GPUs

- Experiments all investigating GPU usage, and GPUs are more available
  - → Landscape of LHC computing hardware may change

- Would like to avoid the risk of making event generators “legacy computing”

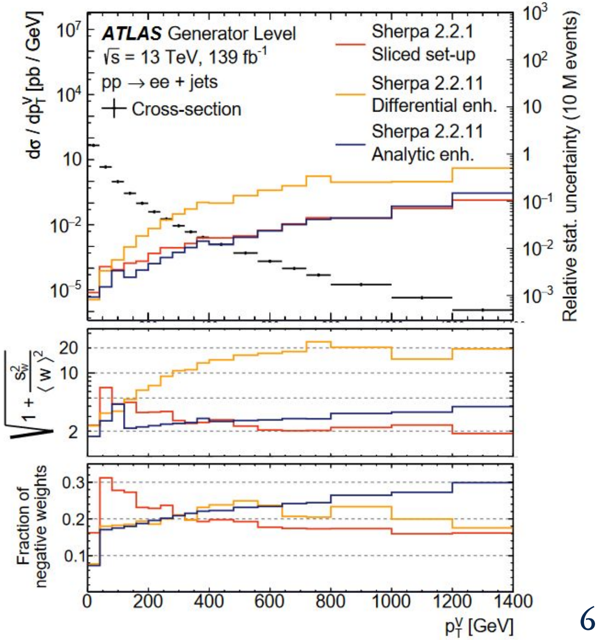
- Experiments are building up expertise on GPUs and could also offer help
- With *modular* event generators, GPU-based parts could be shared
  - A very old idea, of course, which has never really taken flight
- Some risk here — we should ensure the GPU code can be understood and maintained by the generator teams themselves

Treatment of negative weights

- Negative weights are a statistics killer
  - Statistical power of a sample with negative weight fraction ε is reduced by $1/(1-2\varepsilon)^2$
  - ε=25% → 4x larger sample is needed for the same statistical power

- If the negative weight fraction is >30%, samples are hardly usable

- Various techniques have been proposed (1, 2, 3) for improving this
  - How can we ensure a widely-deployed solution?
  - Should we focus the community on *one* solution to avoid divergence?

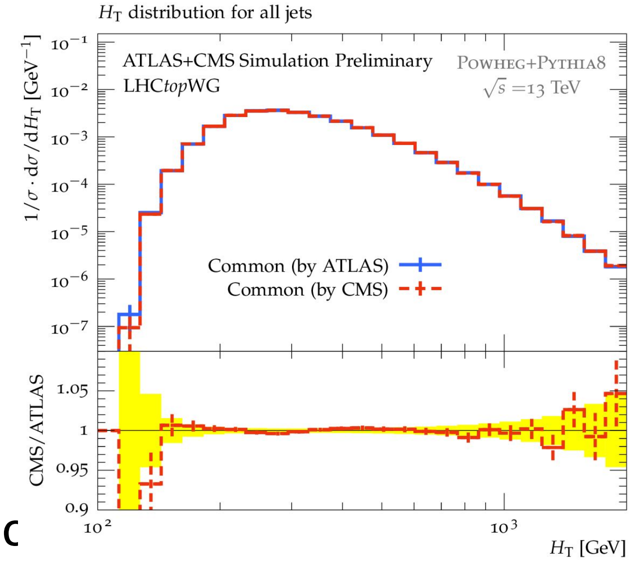


Physics setup

- Efficient slicing
  - Low statistics for very high cross-section kinematic regions
  - Higher statistics in the tails of distributions
- Improving filter efficiency (high p_T, heavy flavor, etc)
  - Ensuring we generate what we want
  - Major missing piece: ‘flavor enhancement’
- New phase-space sampling techniques that avoid biases
  - Particularly for populating unusual kinematic regions
- Integrating (all?) systematic uncertainties as weights
  - Current systematic model comes with up to 5 alternative samples

Sharing events

- Could save 50% of CPU and eventually disk space when sharing generated events
  - ○ Could at least overlay identical theory lines
  - ○ Or have a common alternative sample
  - ○ Or ATLAS alternative is CMS baseline etc.
- First step done within the LHCtopWG: Common Powheg+Pythia8 tt sample



The main problem here is to agree on commo

→ MC community might propose default settings, especially for shower tunes

Working Together

- Our production versions often lag behind development versions
- New advances are sometimes very hard to deploy or not well supported
  - Athena framework is quite different from standalone running
  - If a subset of generator authors worked on something new, all (or our contacts) might not know about the details
  - ATLAS has struggled with NLO bb4l for many years now; same for Dire, Vincia

- How can we work together better to ensure that we all profit from the newest advances as quickly as possible?
  - Is this something the HSF or an LPCC working group could help?
  - Closer interaction between generator experts and experts within the collaborations