Swift-HEP Generators: Status Update

Enrico Bothmann (Göttingen), Andy Buckley (Glasgow), Ilektra Christidi (UCL), Christian Gütschow (UCL), Stefan Höche (FNAL), Max Knobbe (Göttingen), Marek Schönherr (Durham)

Swift-HEP workshop

30 March 2023
Expected computing requirements

- Latest update to the 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.

---

**ATLAS Preliminary**
2022 Computing Model - CPU: 2031, Conservative R&D

- Total: 33.8 MHS06*y

---

**ATLAS Preliminary**
2022 Computing Model - CPU

- Conservative R&D
- Aggressive R&D
- Sustained budget model (+10% +20% capacity/year)

---

**Data Proc**: 24%
**MC-Full(Sim)**: 7%
**MC-Full(Rec)**: 5%
**MC-Fast(Sim)**: 7%
**MC-Fast(Rec)**: 8%
**EvGen**: 17%
**Heavy Ions**: 8%
**Data Deriv**: 6%
**MC Deriv**: 11%
**Analysis**: 8%
Systematic profiling

Most event generation CPU spent on multi-leg NLO calculations [JHEP 08 (2022) 089]

- used for main Standard Model processes
- relevant to measurements and searches alike
- extremely large event sample sizes
Systematic profiling

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

- Study CPU performance of MEPS@NLO calculations for
  $e^+ e^- + 0, 1, 2j@NLO+3, 4, 5j@LO$ and $t\bar{t} + 0, 1j@NLO+2, 3, 4j@LO$
  with Sherpa 2.2.11, OpenLoops 2.1.2 and LHAPDF 6.2.3 using VTune 2021.7.1
Systematic profiling

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

- Study CPU performance of MEPS@NLO calculations for \( e^+e^- + 0, 1, 2j@NLO+3, 4, 5j@LO \) and \( t\bar{t} + 0, 1j@NLO+2, 3, 4j@LO \)
  with Sherpa 2.2.11, OpenLoops 2.1.2 and LHAPDF 6.2.3 using VTune 2021.7.1

- Performance dependence on the number of multiweights studied using different setups:
  - baseline MEPS@NLO (no variations)
  - + EW\(_{\text{virt}}\) corrections
  - + 7-point variations of factorisation and renormalisation scales in matrix element and parton shower
  - + 100 (1000) NNPDF3.0nnlo replicas
Systematic profiling

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

→ Study CPU performance of MEPS@NLO calculations for $e^+e^- + 0, 1, 2j@NLO+3, 4, 5j@LO$ and $t\bar{t} + 0, 1j@NLO+3, 4j@LO$
  with Sherpa 2.2.11, OpenLoops 2.1.2 and LHAPDF 6.2.3 using VTune 2021.7.1

→ performance dependence on the number of multiweights studied using different setups:
  → baseline MEPS@NLO (no variations)
  → + EW$_{virt}$ corrections
  → + 7-point variations of factorisation and renormalisation scales
  in matrix element and parton shower
  → + 100 (1000) NNPDF3.0nnlo replicas

→ detailed write-up presented in [EPJC 82 (2022) 12]
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]

→ explore two approaches in parallel: make LHAPDF faster and rework LHAPDF call strategy
**Impact of new LHAPDF**

- ATLAS $V+\text{jets}$ 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 in Sherpa 2.2.12: the 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 $V+\text{jets}$ 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 $\sim 35\%$ speed improvement for the $V+\text{jets}$ setup
Full suite of improvements

- study the impact of different improvements sequentially:
Full suite of improvements

➔ study the impact of different improvements sequentially:
  ➔ improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)
Full suite of improvements

➜ study the impact of different improvements sequentially:

➜ improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)

➜ replace full-colour spin-correlated S-MC@NLO algorithm with leading-colour spin-averaged $\langle LC \rangle$-MC@NLO ($\text{NLO\_CSS\_PSMODE} 0 \rightarrow 1$)

➜ this disables subleading colour corrections in the parton shower
Full suite of improvements

➔ study the impact of different improvements sequentially:

➔ improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)

➔ replace full-colour spin-correlated S-MC@NLO algorithm with leading-colour spin-averaged ⟨LC⟩-MC@NLO (NLO_CSS_PSMODE 0 → 1)
  ➔ this disables subleading colour corrections in the parton shower

➔ introduce pilot run in Sherpa (2.2.11 → 2.2.12)
Full suite of improvements

→ study the impact of different improvements sequentially:
  
  ➜ improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)
  
  ➜ replace full-colour spin-correlated S-MC@NLO algortihm with leading-colour spin-averaged $\langle LC \rangle$-MC@NLO ($\text{NLO}_\text{CSS}_\text{PSMODE} \ 0 \ → \ 1$)
    ➜ this disables subleading colour corrections in the parton shower
  
  ➜ introduce pilot run in Sherpa (2.2.11 → 2.2.12)
  
  ➜ defer leading-colour MC@NLO until after the unweighting ($\text{NLO}_\text{CSS}_\text{PSMODE} \ 1 \ → \ 2$)
Full suite of improvements

- study the impact of different improvements sequentially:
  - improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)
  - replace full-colour spin-correlated S-MC@NLO algorithm with leading-colour spin-averaged $\langle LC \rangle$-MC@NLO ($\text{NLO\_CSS\_PSMODE} \ 0 \rightarrow 1$)
    - this disables subleading colour corrections in the parton shower
  - introduce pilot run in Sherpa (2.2.11 → 2.2.12)
  - defer leading-colour MC@NLO until after the unweighting ($\text{NLO\_CSS\_PSMODE} \ 1 \rightarrow 2$)
  - use analytic one-loop amplitudes from MCFM in pilot run
Full suite of improvements

- study the impact of different improvements sequentially:
  - improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)
  - replace full-colour spin-correlated S-MC@NLO algorithm with leading-colour spin-averaged $\langle LC\rangle$-MC@NLO ($\text{NLO\_CSS\_PSMODE}\ 0 \rightarrow 1$)
    - this disables subleading colour corrections in the parton shower
  - introduce pilot run in Sherpa (2.2.11 → 2.2.12)
  - defer leading-colour MC@NLO until after the unweighting ($\text{NLO\_CSS\_PSMODE}\ 1 \rightarrow 2$)
  - use analytic one-loop amplitudes from MCFM in pilot run
  - use a simplified pilot scale for the unweighting
**Full suite of improvements**

- study the impact of different improvements sequentially:
  - improved interpolation strategies in LHAPDF (6.2.3 → 6.4.0)
  - replace full-colour spin-correlated S-MC@NLO algorithm with leading-colour spin-averaged \(⟨LC⟩\)-MC@NLO (\(\text{NLO\_CSS\_PSMODE} \ 0 \ → \ 1\))
    - this disables subleading colour corrections in the parton shower
  - introduce pilot run in Sherpa (2.2.11 → 2.2.12)
  - defer leading-colour MC@NLO until after the unweighting (\(\text{NLO\_CSS\_PSMODE} \ 1 \ → \ 2\))
  - use analytic one-loop amplitudes from MCFM in pilot run
  - use a simplified pilot scale for the unweighting

<table>
<thead>
<tr>
<th>setup variant</th>
<th>(pp \rightarrow e^+ e^- +) jets runtime [CPU h/5k events]</th>
<th>(pp \rightarrow t\bar{t} +) jets runtime [CPU h/1k events]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>old new speed-up</td>
<td>old new speed-up</td>
</tr>
<tr>
<td>no variations</td>
<td>20 h 5 h 4×</td>
<td>15 h 8 h 2×</td>
</tr>
<tr>
<td>(\text{EW}_{\text{virt}})</td>
<td>35 h 5 h 6×</td>
<td>20 h 8 h 2×</td>
</tr>
<tr>
<td>(\text{EW}_{\text{virt}} +)</td>
<td>45 h 5 h 7×</td>
<td>25 h 8 h 4×</td>
</tr>
<tr>
<td>scales</td>
<td>90 h 5 h 15×</td>
<td>55 h 8 h 7×</td>
</tr>
<tr>
<td>(\text{EW}_{\text{virt}} +)</td>
<td>725 h 8 h 78×</td>
<td>440 h 9 h 51×</td>
</tr>
<tr>
<td>scales+100 PDFs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{EW}_{\text{virt}} +)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scales+1000 PDFs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Breakdown of CPU budget in $V + \text{jets}$
Case study: latest ATLAS baseline configuration

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

Swift-HEP workshop, 30 Mar 2023

chris.g@cern.ch
Ongoing COMIX studies

→ no more ‘low hanging fruits’ at this point

→ focus now shifting to COMIX
   (used for high-multiplicity LO legs)

→ code memory bound, but no localised bottlenecks
   → implementation of a weight cache to avoid having to resolve virtual classes showed promising results at LO but tricky to generalise to NLO

→ not amenable to auto-vectorisation: could vectorise manually, but poor gain/pain ratio (processing time fragmented across too many different parts of the code)

→ memory cache misses checked but not too significant (∼5% L1, ∼11% L2, ∼4% L3)

→ targeted profiling of costliest functions
   → attempt to optimise memory access in Vegas interface showed little success
   → substitution of custom shared pointer with STL implementation looking more promising
Looking ahead

- Phase-space sampling
- Matrix-element calculation
- Monte Carlo integration
- Monte Carlo unweighting
- Phase-space optimisation
- Parton showering
- Hadronisation, decays, QED
- Particle-level filtering

Sampling inefficiency
Memory limitations
Unweighting inefficiency
Matching inefficiency + interface deficiencies
Interface deficiencies
Filtering inefficiency

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

➔ overall factor 40 speed-up following dedicated profiling of ATLAS multi-leg NLO setups
  ➔ latest LHAPDF release series brings major performance improvements
    with noticeable impact on overall event-generation run time
  ➔ introduction of pilot run in Sherpa brings a factor 5 improvement
  ➔ using analytic QCD loop amplitudes in the unweighting brings another factor 1.5

➔ achieves major factor-10 milestone set by HEP Software Foundation

➔ remaining processing time fragmented with no obvious bottleneck
  ➔ auto-vectorisation doesn’t seem to help much
  ➔ switch to modern C++ utilities (e.g. smart pointers) appears more promising
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
Breakdown of CPU budget in $t\bar{t}+\text{jets}$

- $pp \rightarrow t\bar{t}+0,1@\text{NLO}+2,3,4@\text{LO}$

- Diagram showing breakdown of CPU budget with categories:
  - Phase space
  - Tree-level ME
  - PDF
  - Clustering
  - Loop ME
  - Rest + overhead

- Run time for 1000 events [CPU h]: $10^0$ to $10^3$
- Proportion of total run time: 0.2 to 1.0

- Comparison of different runs:
  - Mc@NLO baseline
  - LHAPDF 6.4.0
  - $\langle LC \rangle$-MC@NLO
    - Pilot run
    - Pilot scale
  - $\langle LC \rangle$-MC@NLO-CSS
  - McFM
  - $\langle LC \rangle$-

- No variations
- EW virt + scales + 100 PDFs
- EW virt + scales + 1000 PDFs

Swift-HEP workshop, 30 Mar 2023
chris.g@cern.ch
Cluster-independent scale definition

- employ clustering-independent scale definition ($H'_T/2$) for $H$-events in $\bar{t}t +$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:

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

$pp \rightarrow t\bar{t} + 0, 1@NLO + 2, 3, 4@LO$

→ second unweighting would reduce the efficiency by less than factor 2 for large $N_{events}$