Analysis tools and measurement reinterpretation

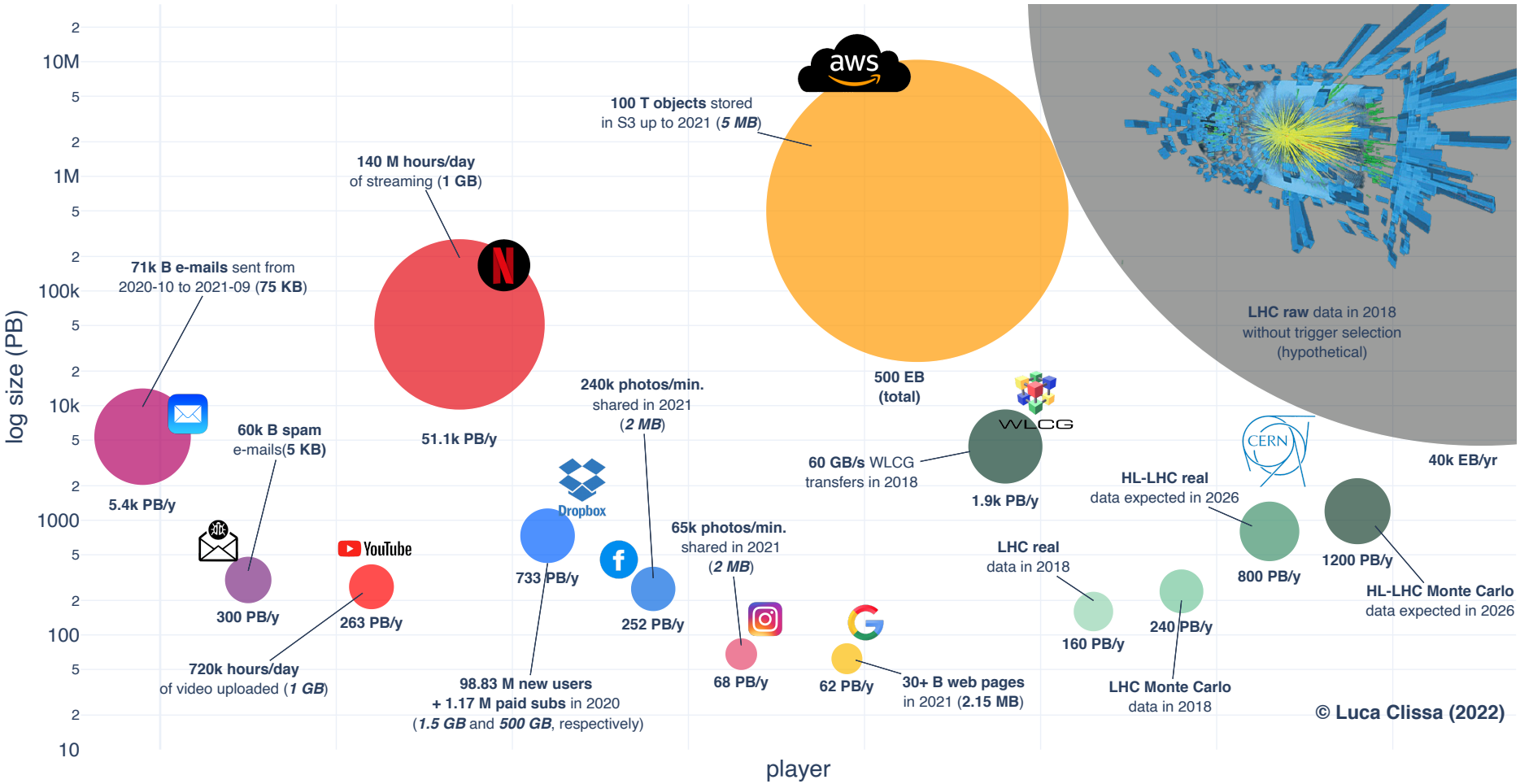
Christian Gütschow

YETI, Durham

13 July 2026

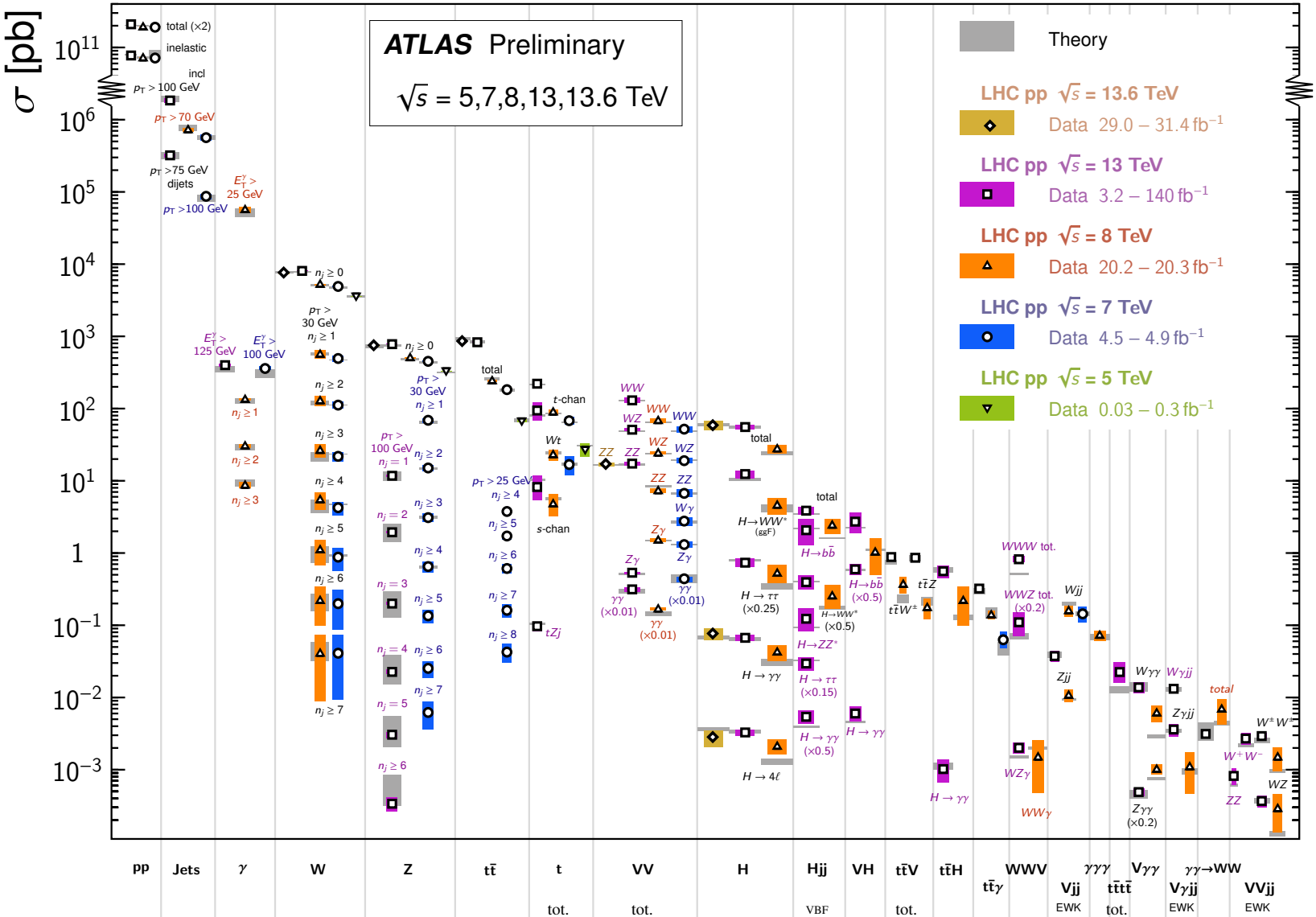
LHC = Big Data

[arXiv:2202.07659]

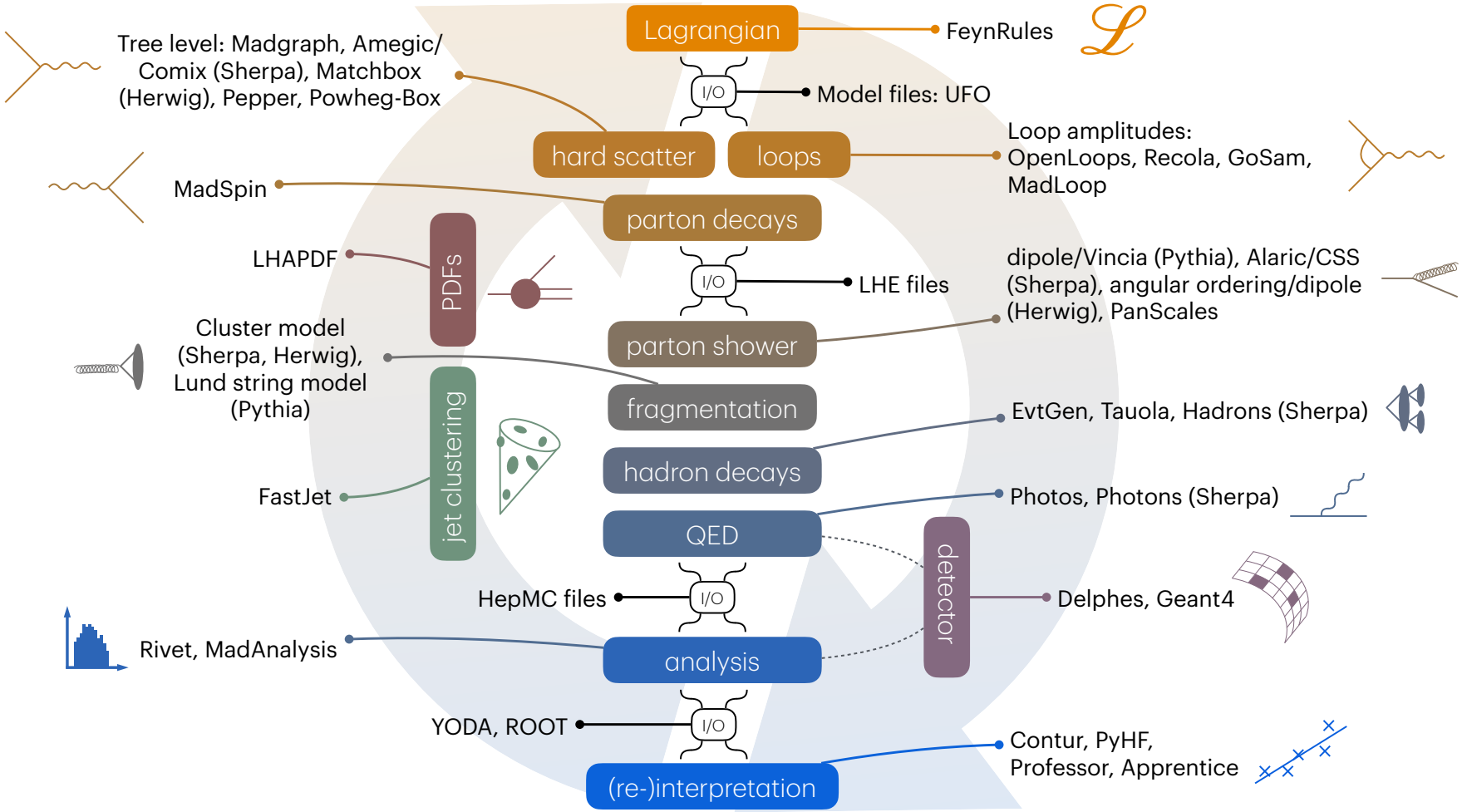


Standard Model Production Cross Section Measurements

Status: June 2024



The Monte Carlo ecosystem in High-Energy Physics



[arXiv:2605.16036]

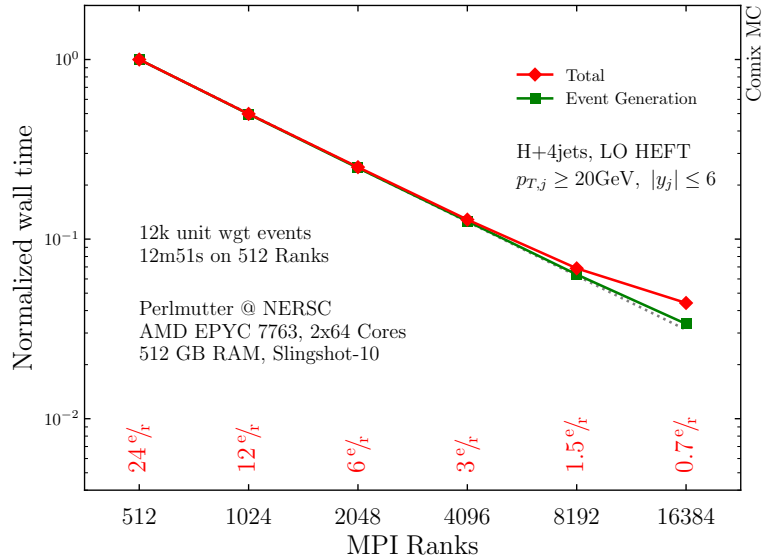
Open-source tools & community infrastructure

- Monte Carlo (support) tools are community-driven and open-source
  - Transparent, inspectable, and modifiable code
- Widely used and maintained by both theorists and experimentalists:
  - FastJet: Standard library for jet clustering in collider physics
  - LHAPDF: Standardised interface for accessing and interpolating PDF sets
  - HepMC: Common event record format for storing full generator output
  - Rivet: Portable, experiment-independent analysis toolkit for generator validation
  - YODA: Histogramming and statistical accumulator library used by Rivet
  - Contur: Constraint setting from LHC measurements and beyond using Rivet
  - Professor/Apprentice: Generator tuning framework using fast parameterisation
  - HEPData: Archival and retrieval of experimental data for reuse in MC validation
- Shared infrastructure = less duplication, better validation, reproducibility
- Open tools = easier training and broader contributions
- Cross-talk between experiments, theorists, and developers is essential.

Generator output formats

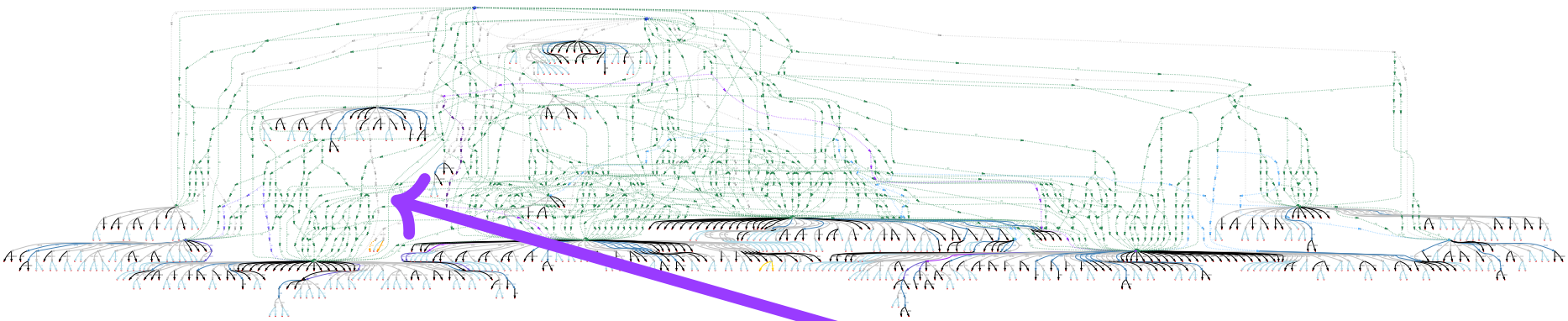
- Les Houches Event (LHE) format:
  - Parton-level only (pre-shower)
  - includes matrix-element weights and metadata (PDFs, scales, etc.)
  - Mainly used for input to shower generators
  - available as ASCII or HDF5
- HepMC:
  - Full event record including parton shower, hadronisation, decays
  - Common standard for full event simulation
  - Supports multiweights, named particles, vertex info, etc.
- ROOT ntuples:
  - Mainly a wrapper around HepMC and/or detector-simulated samples

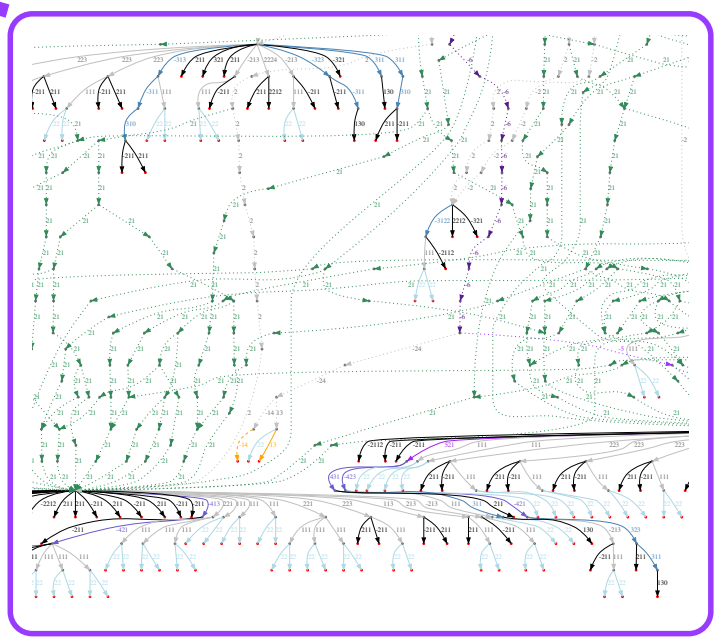
LHEH5 strong scaling at parton level



[PRD 109 (2024) 1]

The HepMC event graph





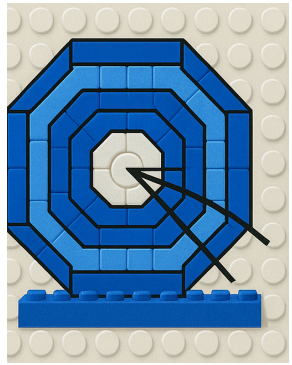
- A MC event graph showing all the particles (lines) and interactions (vertices) in the Pythia event record.
- Guess the event type?
  (Hint: 2212 = proton, 1–6 = quarks,
  11–16 = leptons, 21–25 = bosons)
- Ouff . . . how to safely analyse *that*?
  - There is no unambiguous truth in quantum mechanics!
  - Event records are half physics, half debug info – and zero indication of amplitude interference!
  - Kinematic frames aren't defined until the final state, momentum not necessarily conserved at all vertices.
  - **Beware!**

Fiducial cross-sections: simple idea, big impact

➜ A surprisingly powerful idea

- Define cross-sections using observable final-state particles, within a clearly specified kinematic region.
- Makes it easy to reproduce key plots, enabling real understanding, catching issues early, and improving MCs.
- Establishes a shared language between theory and experiment – essential for tuning, fits, and reinterpretation.

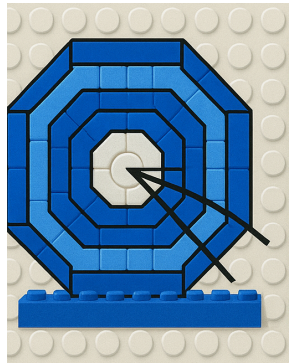
[fiducial]



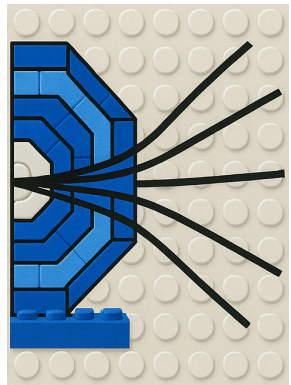
Fiducial cross-sections: simple idea, big impact

- A surprisingly powerful idea
  - Define cross-sections using observable final-state particles, within a clearly specified kinematic region.
  - Makes it easy to reproduce key plots, enabling real understanding, catching issues early, and improving MCs.
  - Establishes a shared language between theory and experiment – essential for tuning, fits, and reinterpretation.
- But it's tempting to cheat . . .
  - Partons, bosons, and other "truth" objects in the event record look easy to use.
  - In practice, they're often ambiguous, model-dependent, or even non-existent (e.g. in higher-order simulations).
  - Subtracting processes with the same fiducial final state (e.g. different production modes) introduces unnecessary model dependence.
  - Measure the fiducial final state directly; interpretations in terms of production mechanisms can always be added later.

[fiducial]



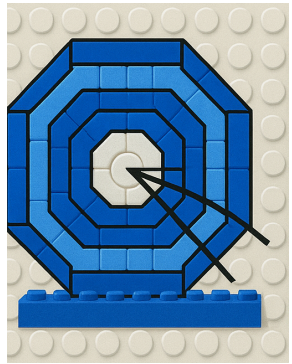
[extrapolated]



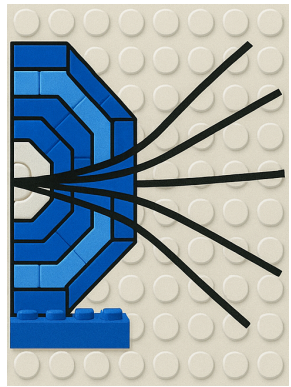
Fiducial cross-sections: simple idea, big impact

- A surprisingly powerful idea
  - Define cross-sections using observable final-state particles, within a clearly specified kinematic region.
  - Makes it easy to reproduce key plots, enabling real understanding, catching issues early, and improving MCs.
  - Establishes a shared language between theory and experiment – essential for tuning, fits, and reinterpretation.
- But it's tempting to cheat . . .
  - Partons, bosons, and other "truth" objects in the event record look easy to use.
  - In practice, they're often ambiguous, model-dependent, or even non-existent (e.g. in higher-order simulations).
  - Subtracting processes with the same fiducial final state (e.g. different production modes) introduces unnecessary model dependence.
  - Measure the fiducial final state directly; interpretations in terms of production mechanisms can always be added later.
- Who knows – **nature might still surprise us**!

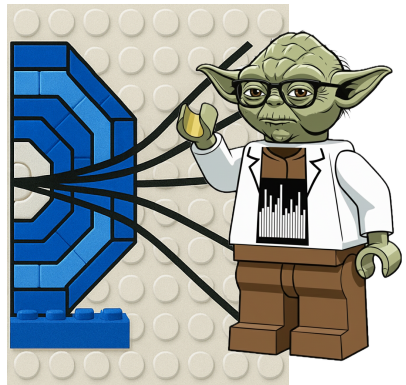
[fiducial]



[extrapolated]



[nature]



Why it matters

- Standardisation sounds boring – but agreeing on things like status codes PDG IDs, and weight schemes saves huge headaches later.
- Avoiding unphysical shortcuts has led to genuine physics insight, including:
  - Recognition of hadronisation as a “decoherence barrier”, guiding analysis definitions
  - Better truth tagging strategies: e.g. using fragmenting heavy-flavour hadrons rather than ancestry of soft partons
  - The shift to dressed leptons by default, i.e. truth leptons with their photon halos attached
- Common pitfalls to avoid:
  - Coloured objects aren’t final-state particles: “final-state tops” aren’t physical!
  - Electroweak bosons aren’t final states either: prefer leptonic or hadronic decay products.
  - Missing energy $\neq$ sum of neutrino momenta: use particle-level missing transverse momentum instead.
  - Hidden cuts hide physics: all vetoes and selections should be encoded in the fiducial definition, not just the code.

[arXiv:1901.05407]

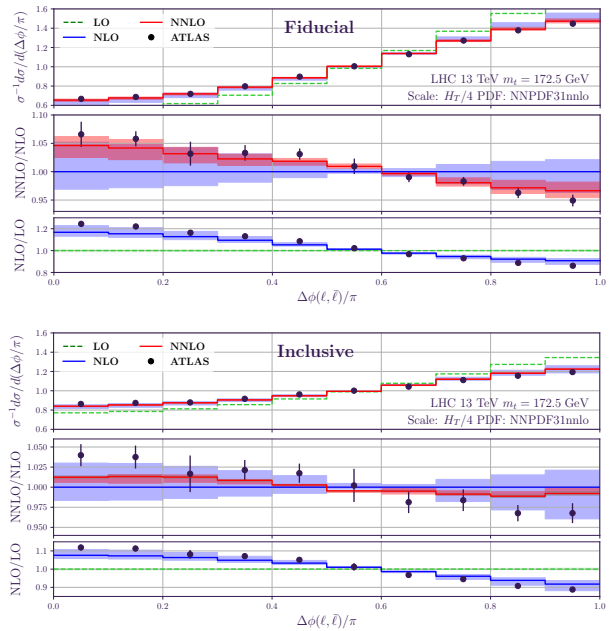


FIG. 1: NNLO QCD predictions for the fiducial (top) and inclusive selections (bottom) of the normalized $\Delta\phi_{\ell\ell}$ distribution versus ATLAS data [20]. Uncertainty bands are from 7-point scale variation.

Introducing Rivet

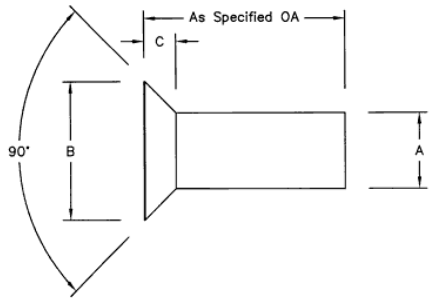
- Robust and Independent Validation of Experiment and Theory! [rivet.hepforge.org]
- Widely adopted by both experimental and theoretical particle physics communities as the common “language” for MC analysis.
- First released in 2007, fourth major version available as of 2024. [gitlab.com/hepcedar/rivet]
- Written in C++, with Python-based command-line tools for flexible workflows.
- Ensures consistent and robust comparison of theoretical predictions and experimental measurements.

Robust Independent Validation of Experiment and Theory: Rivet version 4 release note

Christian Bierlich, Andy Buckley, Jonathan Butterworth, Christian Gutschow, Leif Lonnblad, Tomasz Procter, Peter Richardson, Yoran Yeh

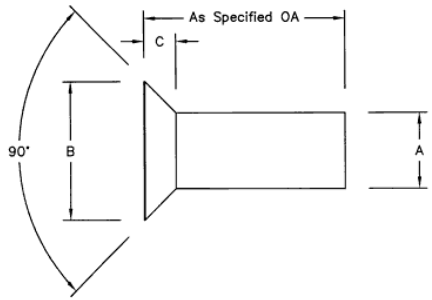
[arXiv:2404.15984]

Designing the Rivet



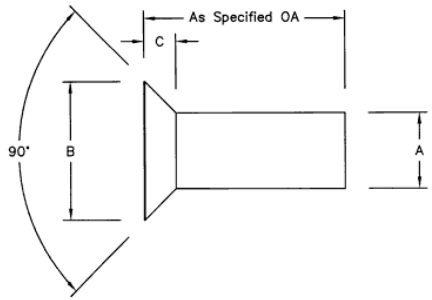
- Ease of use
  - Focus on enabling physicists to concentrate on physics insights rather than technical details.
  - Minimal boilerplate code for cleaner, simpler analysis writing.
  - Familiar event loop structure and intuitive histogramming tools.
  - Streamlined integration for syncing results with external data sources like [HepData].

Designing the Rivet



- Ease of use
  - Focus on enabling physicists to concentrate on physics insights rather than technical details.
  - Minimal boilerplate code for cleaner, simpler analysis writing.
  - Familiar event loop structure and intuitive histogramming tools.
  - Streamlined integration for syncing results with external data sources like [HepData].
- Flexible and embeddable
  - Core functionality in modern C++ with Python bindings for enhanced scripting flexibility.
  - Works with any event generator using the standard [HepMC] format for seamless integration.
  - Analyses are modular and dynamically loaded as “plugins”, promoting code reuse and clarity.

Designing the Rivet



- Ease of use
  - Focus on enabling physicists to concentrate on physics insights rather than technical details.
  - Minimal boilerplate code for cleaner, simpler analysis writing.
  - Familiar event loop structure and intuitive histogramming tools.
  - Streamlined integration for syncing results with external data sources like [HepData].
- Flexible and embeddable
  - Core functionality in modern C++ with Python bindings for enhanced scripting flexibility.
  - Works with any event generator using the standard [HepMC] format for seamless integration.
  - Analyses are modular and dynamically loaded as “plugins”, promoting code reuse and clarity.
- Efficient and scalable
  - Built-in caching system to avoid redundant computations during event processing.

What is a Rivet routine?

- A modular analysis plugin to process simulated collision events.
- Encodes physics logic for event selection, kinematic calculations, and histogramming.
- Pre-built analysis functions for standard observables.
- Designed to work with any MC event generator.

Each Rivet analysis is a plugin inheriting from `Rivet::Analysis`.

Setup phase: Book histograms, declare support algorithms etc.

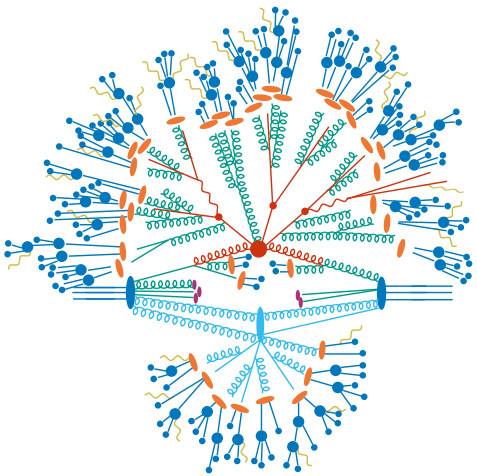
Histograms are automatically managed and linked to reference data where available.

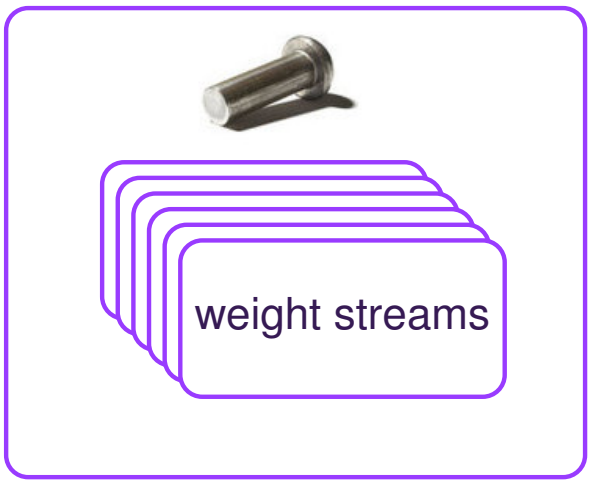
Main event loop: Retrieve event data, apply selections, fill histograms.

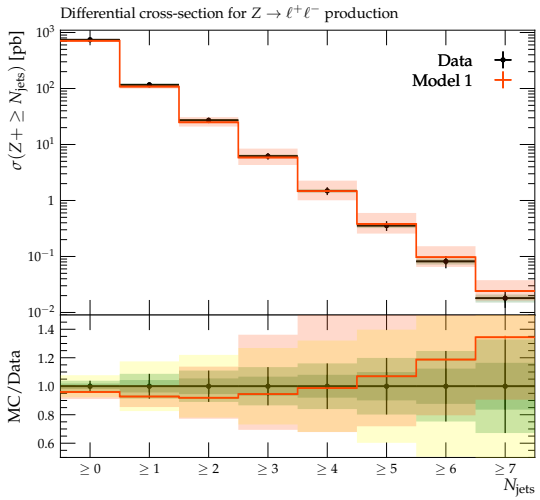
Final normalisation step: Scale histograms based on event weight sum.

```
#include "Rivet/Analysis.hh"
#include "Rivet/Projections/FastJets.hh"

namespace Rivet {
  class MySimpleAnalysis : public Analysis {
  public:
    RIVET_DEFAULT_ANALYSIS_CTOR(MySimpleAnalysis);

    void init() {
      FastJets fj(FinalState(), JetAlg::ANTIKT, 0.4);
      declare(fj, "Jets");
      book(_h_jetPt, "Jet_Pt", 50, 0, 500);
    }

    void analyze(const Event& event) {
      const Jets& jets = apply<FastJets>(event, "Jets").jetsByPt();
      for (const Jet& jet : jets) _h_jetPt->fill(jet.pT()/GeV);
    }

    void finalize() {
      normalize(_h_jetPt);
    }

  private:
    Histo1DPtr _h_jetPt;
  };

  RIVET_DECLARE_PLUGIN(MySimpleAnalysis);
}
```

Rivet workflow overview







- Monte Carlo event generators produce simulated collision events
- Rivet reads these events, applies analysis routines, and fills histograms.
  - Analysis routines automatically loaded and executed by Rivet's event loop framework.
- Histograms are stored in YODA format, facilitating further analysis, visualisation and reinterpretation studies.

Enter YODA

- Yet more Objects for Data Analysis!
  [yoda.hepforge.org]
- Designed for memory efficiency and speed in high-performance environments.
- First released in 2013, second major version available as of 2023. [gitlab.com/hepcedar/yoda]
- Written in C++ and programmatically usable from C++ and Python, complemented by a set of command-line tools for dataset inspection, manipulation and combination.

Consistent, multidimensional differential histogramming and summary statistics with YODA 2

Andy Buckley, Louie Corpe, Matthew Filipovich, Christian Gutschow, Nick Rozinsky, Simon Thor, Yoran Yeh, Jamie Yellen

[arXiv:2312.15070]

- Emerged from the sub-field of MC event generator analysis in particle physics, but library is deliberately agnostic of any particular application

Flexible bin partitioning for modern analysis needs

- New flexible `Axis` class supports both continuous and discrete data types.
  - Template-based design adapts to different edge types automatically.

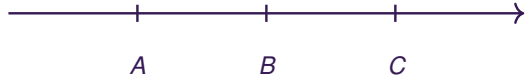
Flexible bin partitioning for modern analysis needs

- New flexible `Axis` class supports both continuous and discrete data types.
  - Template-based design adapts to different edge types automatically.
- Continuous axis (classic mode)
  - Triggered by floating-point types using standard type traits.
  - Binning defined by $N + 1$ edges for N bins, plus under-/overflow bins.
  - Bin widths handle infinite ranges where needed.

Flexible bin partitioning for modern analysis needs

- New flexible `Axis` class supports both continuous and discrete data types.
  - Template-based design adapts to different edge types automatically.
- Continuous axis (classic mode)
  - Triggered by floating-point types using standard type traits.
  - Binning defined by $N + 1$ edges for N bins, plus under-/overflow bins.
  - Bin widths handle infinite ranges where needed.

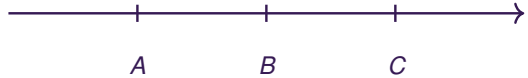
- Discrete axis (new mode)
  - Designed for non-continuous types (e.g. integers, categories).
  - Bins defined by N edges and a special “otherflow” bin for outliers.
  - Ideal for multiplicities, cutflows, and categorical data handling.



Flexible bin partitioning for modern analysis needs

- New flexible `Axis` class supports both continuous and discrete data types.
  - Template-based design adapts to different edge types automatically.
- Continuous axis (classic mode)
  - Triggered by floating-point types using standard type traits.
  - Binning defined by $N + 1$ edges for N bins, plus under-/overflow bins.
  - Bin widths handle infinite ranges where needed.

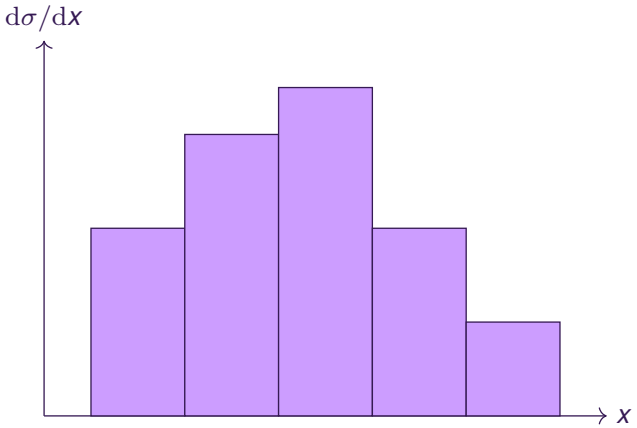
- Discrete axis (new mode)
  - Designed for non-continuous types (e.g. integers, categories).
  - Bins defined by N edges and a special “otherflow” bin for outliers.
  - Ideal for multiplicities, cutflows, and categorical data handling.



- Advanced `Binning` features
  - Seamlessly translates between local bin indices and global index positions.
  - Supports slicing and marginalisation across multi-dimensional spaces.

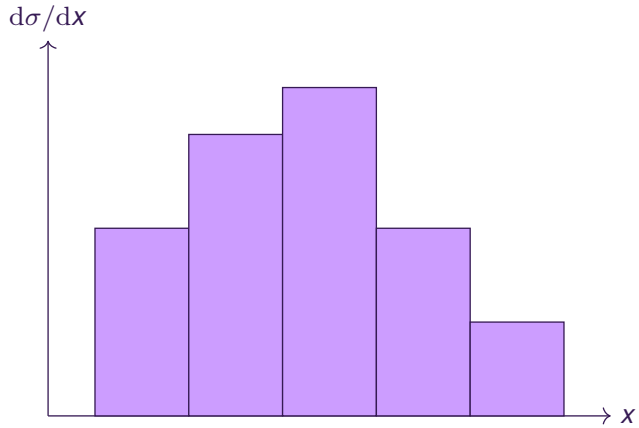
Flexible bin content types for advanced data handling

- Live content (`Dbn` class)
  - Generalised multi-dimensional version of YODA1's distribution class.
  - Tracks exact first- and second-order statistical moments, including mixed moments.
  - Flexible `fill()` method: accepts coordinates, weights, and fill fractions for dynamic updates.

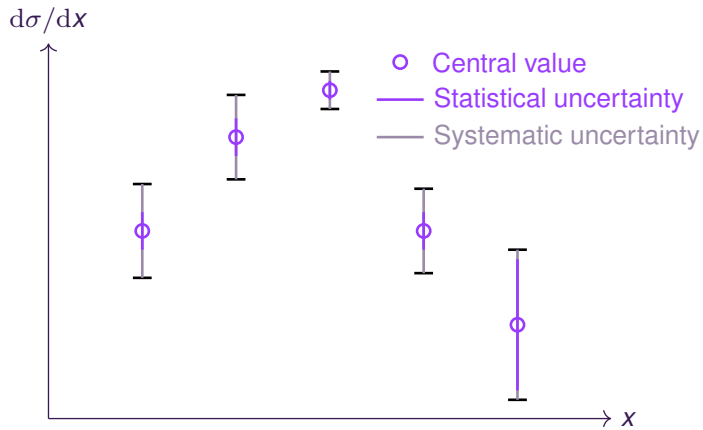


Flexible bin content types for advanced data handling

- Live content (`Dbn` class)
  - Generalised multi-dimensional version of YODA1's distribution class.
  - Tracks exact first- and second-order statistical moments, including mixed moments.
  - Flexible `fill()` method: accepts coordinates, weights, and fill fractions for dynamic updates.



- Inert content (`Estimate` class)
  - Central value representation, optionally with detailed error breakdowns.
  - Encodes uncertainties as labeled {down, up} variations to capture dependence on theoretical or experimental parameters.
  - Supports both correlated and uncorrelated treatments of errors.
  - Arithmetic operations respect these uncertainty relationships for robust statistical handling.

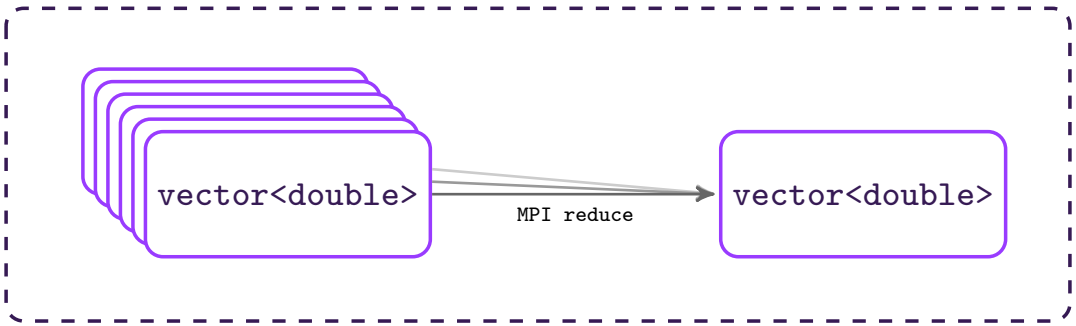


HPC support for distributed and parallel workflows

- Efficient serialisation for MPI communication:
  - `AnalysisObject` base class can be (de-)serialised into/from a `std::vector<double>`.
  - Facilitates easy communication of data across nodes in distributed environments like MPI.

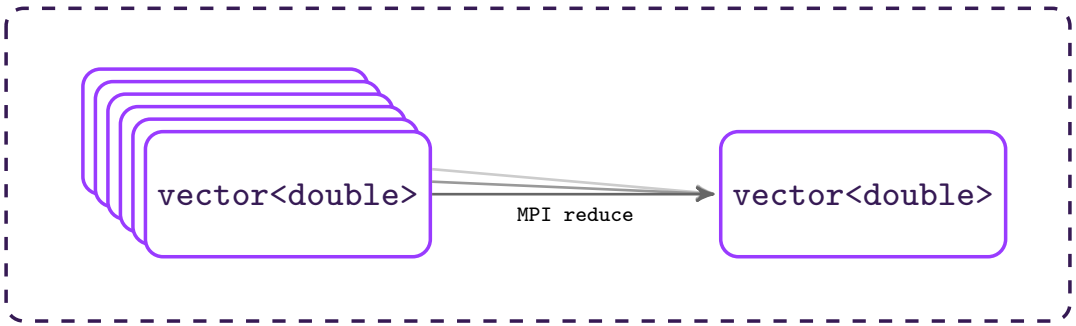
HPC support for distributed and parallel workflows

- Efficient serialisation for MPI communication:
  - `AnalysisObject` base class can be (de-)serialised into/from a `std::vector<double>`.
  - Facilitates easy communication of data across nodes in distributed environments like MPI.
- On-the-fly stacking & merging
  - Serialised data enables efficient stacking and merging of histograms during computation.
  - Supports parallel workflows where intermediate results can be combined dynamically across multiple processes.



HPC support for distributed and parallel workflows

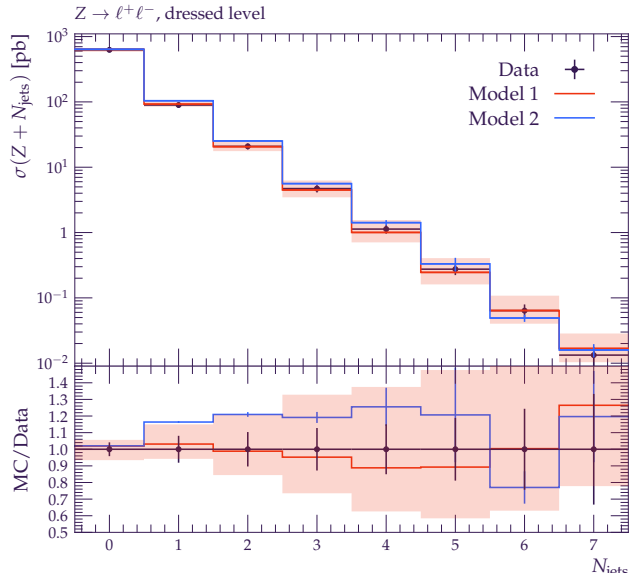
- Efficient serialisation for MPI communication:
  - `AnalysisObject` base class can be (de-)serialised into/from a `std::vector<double>`.
  - Facilitates easy communication of data across nodes in distributed environments like MPI.
- On-the-fly stacking & merging
  - Serialised data enables efficient stacking and merging of histograms during computation.
  - Supports parallel workflows where intermediate results can be combined dynamically across multiple processes.

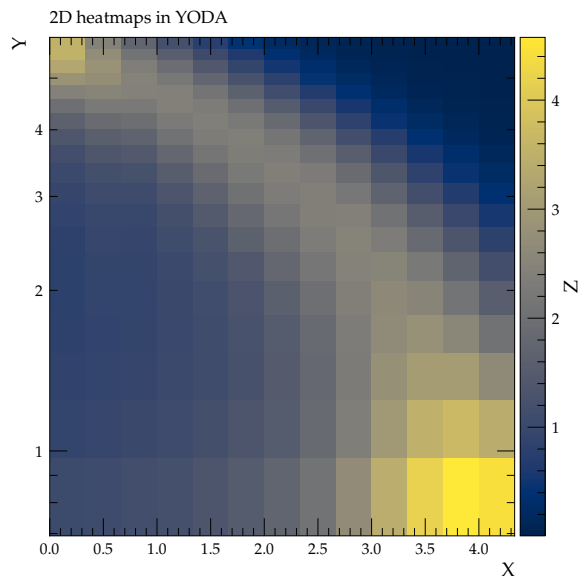


- Optimised for scalability
  - Built to handle large datasets with **minimal memory overhead**, making it well-suited for HPC applications.
  - Seamless integration with parallel computing frameworks ensures **scalability** for big data analysis in particle physics.
  - Applications in Machine Learning: Serialised data can be **easily integrated** into machine learning pipelines for model training, feature extraction, and data preprocessing.

Python API & plotting for seamless integration

- Python bindings via `Cython`
  - YODA provides Python bindings for scripting and integration into Python-based workflows.
  - Enables efficient use of YODA objects and operations from within Python scripts.
- **Customisable** `matplotlib`-based plotting
  - Automatically generates Python scripts that produce plots with `matplotlib`.
  - Self-contained plots: Once the script is generated, no YODA installation is required to produce the plot. Ideal for sharing results with collaborators.
  - Full control over plot aesthetics, allowing for customisation without altering the underlying data structures.
- Share Python-generated plotting scripts with collaborators, ensuring **consistency** in results and **reproducibility**.





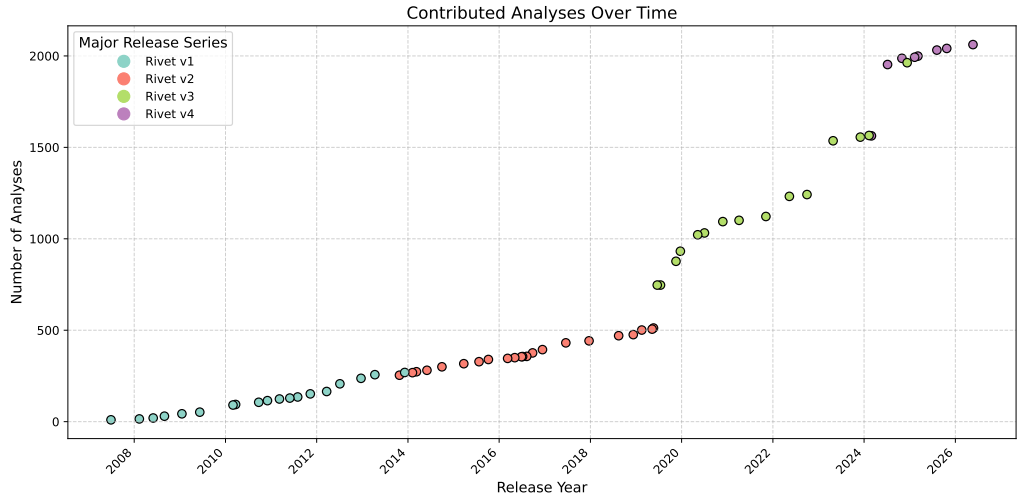
Ensuring long-term impact of HEP results

- Colliders explore **complex final states** and observables in a **new kinematic regime**.
- Robust validation of MC tools will be critical to understand detector effects and theoretical uncertainties in this precision frontier.
- **Future theory developments must be testable against today's measurements.**
- LHC experience shows that without structured analysis preservation, valuable insights are quickly lost:

Key	ALICE	ATLAS	CMS	LHCb	Forward	HERA	e^+e^- (≥ 12 GeV)	e^+e^- (≤ 12 GeV)	Tevatron	RHIC	SPS
Rivet wanted (total):	153	195	197	143	16	413	630	79	672	185	70
Rivet REALLY wanted:	40	67	72	13	3	17	1	0	9	1	7
Rivet provided:	30/183 = 16%	188/383 = 49%	130/327 = 40%	74/217 = 34%	10/26 = 38%	48/461 = 10%	233/863 = 27%	1005/1084 = 93%	59/731 = 8%	7/192 = 4%	15/85 = 18%

LHC analysis preservation in practice

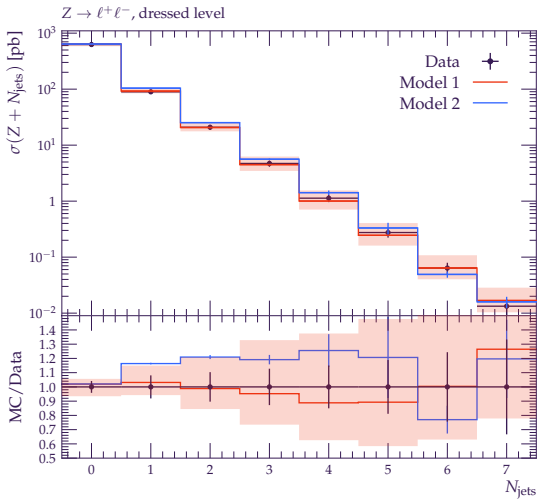
- Validated repository
  - Central library of hundreds of published LHC analyses (and more).
- Transparency in scientific workflow
  - Open-source routines allow full inspection and replication of results.
  - Clear documentation of selection criteria and observable definitions.



- Cross-checking experimental results
  - Independent validation of collider results as well as benchmarking of MC event generators.
  - Direct reproduction of key measurements across experiments, boosting citations.
- Theory reinterpretation studies
  - Quickly test new theory models against archived analysis data.
  - Enables discovery potential for new physics.

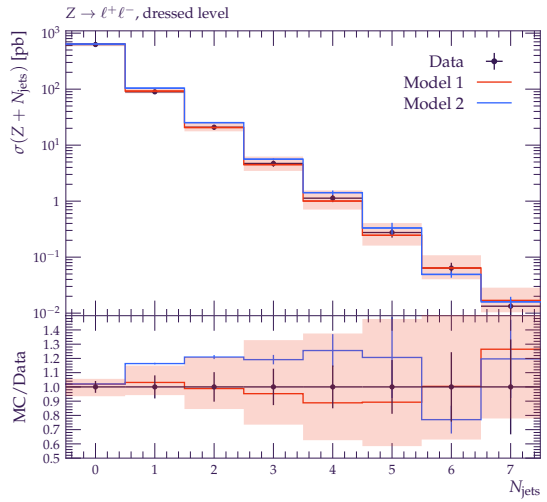
Community and collaboration

- Bridging theory & experiment
  - Rivet is the **de facto standard** for comparing MC event generators with LHC data.
  - Provides a **common framework** that ensures **consistent validation** of theoretical models.
  - Enables a **shared language** between theorists and experimentalists.



Community and collaboration

- Bridging theory & experiment
  - Rivet is the **de facto standard** for comparing MC event generators with LHC data.
  - Provides a **common framework** that ensures **consistent validation** of theoretical models.
  - Enables a **shared language** between theorists and experimentalists.
- Fostering common standards
  - Having a unified toolset facilitates discussions on **consistent methodologies**.
  - Adoption by multiple communities (LHC experiments, MC developers, theorists) helps **align best practices**.



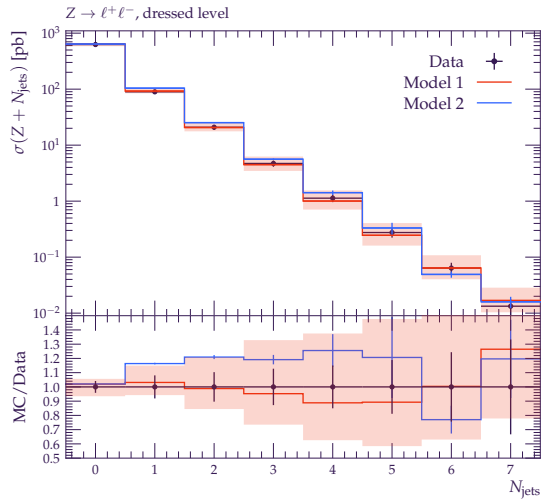
A standard convention for particle-level Monte Carlo event-variation weights

Enrico Bothmann, Andy Buckley, Christian Gütschow, Stefan Prestel, Marek Schönherr, Peter Skands, Jeppe Andersen, Saptaparna Bhattacharya, Jonathan Butterworth, Gurpreet Singh Chahal, Louie Corpe, Leif Gellersen, Matthew Gignac, Deepak Kar, Frank Krauss, Jan Kretzschmar, Leif Lönnblad, Josh McFayden, Andreas Papaefstathiou, Simon Plätzer, Steffen Schumann, Michael Seymour, Frank Siegert, Andrzej Siódmok

[arXiv:2203.08230]

Community and collaboration

- Bridging theory & experiment
  - Rivet is the **de facto standard** for comparing MC event generators with LHC data.
  - Provides a **common framework** that ensures **consistent validation** of theoretical models.
  - Enables a **shared language** between theorists and experimentalists.
- Fostering common standards
  - Having a unified toolset facilitates discussions on **consistent methodologies**.
  - Adoption by multiple communities (LHC experiments, MC developers, theorists) helps **align best practices**.
- Open & evolving
  - Actively maintained by the **HEP software community** with open-source contributions.
  - Regular **workshops, training sessions**, and discussions to drive future improvements.
  - Integrated into **analysis preservation efforts** for long-term impact.



A standard convention for particle-level Monte Carlo event-variation weights

Enrico Bothmann, Andy Buckley, Christian Gütschow, Stefan Prestel, Marek Schönherr, Peter Skands, Jeppe Andersen, Saptaparna Bhattacharya, Jonathan Butterworth, Gurpreet Singh Chahal, Louie Corpe, Leif Gellersen, Matthew Gignac, Deepak Kar, Frank Krauss, Jan Kretzschmar, Leif Lönnblad, Josh McFayden, Andreas Papaefstathiou, Simon Plätzer, Steffen Schumann, Michael Seymour, Frank Siegert, Andrzej Siódmok

[arXiv:2203.08230]

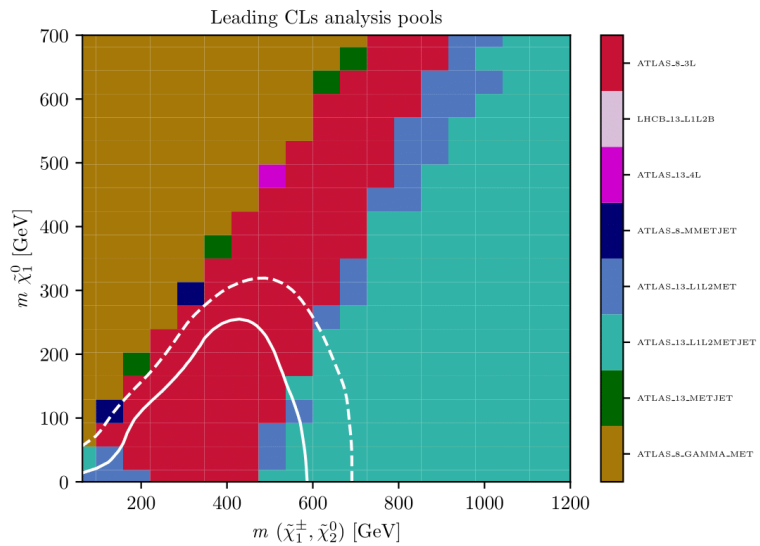
Reinterpreting LHC data for new physics

- Testing new physics hypotheses
  - LHC experiments produce vast datasets of precision measurements.
  - Rivet allows injecting new-physics signals on top of Standard Model predictions.
  - If a Standard Model extension leads to statistically significant deviations, it can be ruled out.

Reinterpreting LHC data for new physics

- Testing new physics hypotheses
  - LHC experiments produce vast datasets of precision measurements.
  - Rivet allows injecting new-physics signals on top of Standard Model predictions.
  - If a Standard Model extension leads to statistically significant deviations, it can be ruled out.
- Fast and broad coverage
  - Rivet provides a synoptic view – many independent measurements across different energies and processes.
  - New Rivet-based analyses can be used in a likelihood fit within hours.

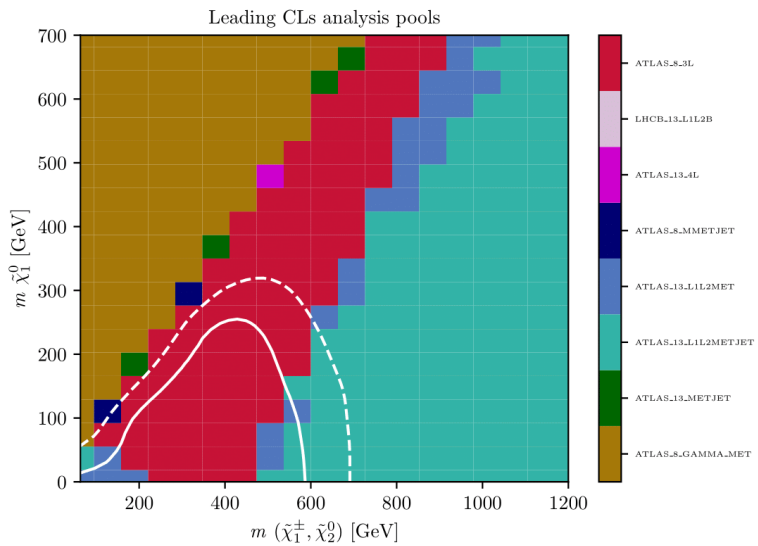
[Contur]



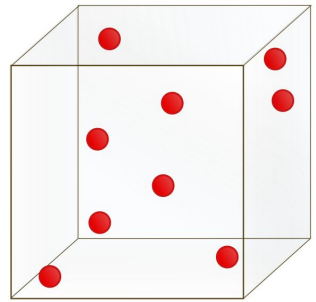
Reinterpreting LHC data for new physics

- Testing new physics hypotheses
  - LHC experiments produce vast datasets of precision measurements.
  - Rivet allows injecting new-physics signals on top of Standard Model predictions.
  - If a Standard Model extension leads to statistically significant deviations, it can be ruled out.
- Fast and broad coverage
  - Rivet provides a synoptic view – many independent measurements across different energies and processes.
  - New Rivet-based analyses can be used in a likelihood fit within hours.
- Extending the new physics search frontier
  - Increasingly precise calculations and new measurements expand the space of possible new physics models.
  - Reusable, high-quality data ensures long-term discovery potential.

[Contur]

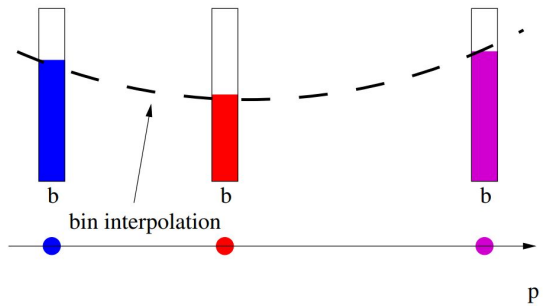


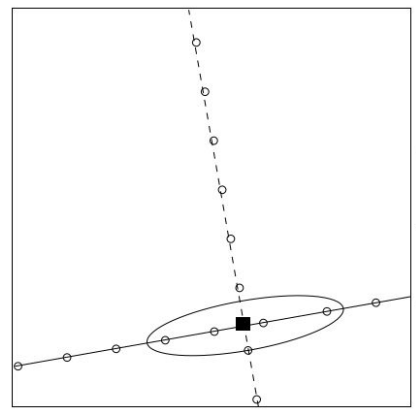
Model tuning



[Andy Buckley]

- Tuning was historically a mix of brute force and expert intuition.
- Professor/Apprentice accelerates convergence by building surrogate models:
  - Sample parameter points p_n from a hypercube/sphere.
  - Generate MC run-sets (for beams, processes, etc.) at each p_n.
  - Run jobs in parallel on batch/grid systems to produce histograms.
  - Fit surrogate models $\hat{b}(p)$ for each histogram bin using the MC outputs – traditionally 3rd/4th-order polynomials, though other form are possible (e.g. rational approximants via Apprentice).
  - Construct a surrogate goodness-of-fit from these models and optimise the parameters efficiently.
- Expert knowledge is still essential – but surrogates dramatically reduce the cost of scanning high-dimensional parameter spaces.
- Machine learning? Sure – but if polynomials (possibly after a variable transformation) capture the behaviour well, they remain simple, transparent, and robust.





Challenges and lessons learned

- ⚖ Balancing generality & usability
  - ➜ Modular design with plugin-based analysis routines.

Challenges and lessons learned

- Balancing generality & usability
  - ➜ Modular design with plugin-based analysis routines.
- Interfacing with experiment & theory
  - ➜ Rivet uses HepMC as a standard interface and maintains version compatibility.

Challenges and lessons learned

- Balancing generality & usability
  - ➜ Modular design with plugin-based analysis routines.
- Interfacing with experiment & theory
  - ➜ Rivet uses HepMC as a standard interface and maintains version compatibility.
- Analysis preservation & long-term reusability
  - ➜ Rigorous archival of validated analyses and automated tests to ensure compatibility.

Challenges and lessons learned

- Balancing generality & usability
  - ➜ Modular design with plugin-based analysis routines.
- Interfacing with experiment & theory
  - ➜ Rivet uses HepMC as a standard interface and maintains version compatibility.
- Analysis preservation & long-term reusability
  - ➜ Rigorous archival of validated analyses and automated tests to ensure compatibility.
- Standardisation matters
  - ➜ Ensuring consistent validation reduces discrepancies in MC predictions.

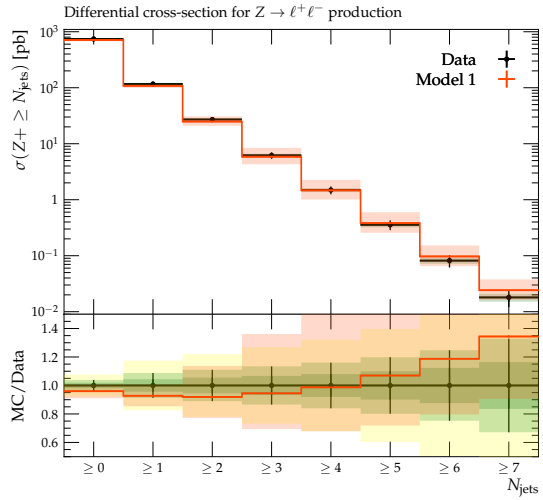
Challenges and lessons learned

- Balancing generality & usability
  - ➜ Modular design with plugin-based analysis routines.
- Interfacing with experiment & theory
  - ➜ Rivet uses HepMC as a standard interface and maintains version compatibility.
- Analysis preservation & long-term reusability
  - ➜ Rigorous archival of validated analyses and automated tests to ensure compatibility.
- Standardisation matters
  - ➜ Ensuring consistent validation reduces discrepancies in MC predictions.
- Community-driven design is key
  - ➜ User feedback has shaped features like multi-pass execution for heavy-ion analyses.

Challenges and lessons learned

- Balancing generality & usability
  - ➜ Modular design with plugin-based analysis routines.
- Interfacing with experiment & theory
  - ➜ Rivet uses HepMC as a standard interface and maintains version compatibility.
- Analysis preservation & long-term reusability
  - ➜ Rigorous archival of validated analyses and automated tests to ensure compatibility.
- Standardisation matters
  - ➜ Ensuring consistent validation reduces discrepancies in MC predictions.
- Community-driven design is key
  - ➜ User feedback has shaped features like multi-pass execution for heavy-ion analyses.
- Sustainability needs effort
  - ➜ Workshops, onboarding & continuous development keep Rivet relevant.

Summary: Rivet for collider analysis & beyond



- Standard tool for MC validation & analysis preservation
- Bridges experiment & theory with a common framework
- Designed for usability, flexibility, and long-term reusability
- Supports evolving physics needs
- Sustained by an active community – contributions welcome!
- Further reading:
  - **Practical Collider Physics** (2021)
    [Buckley, White (×2)]
  - **MC primer** (2026)
    [arxiv:2605.16036]

[rivet.hepforge.org]

[gitlab.com/hepcedar/rivet]

[docker:hepstore/rivet]

[arXiv:2404.15984]

- These tools are long-lived, shared infrastructure – you'll use them for years, and you can contribute back!

MCnet short-term studentships

- 3-month PhD placements with MC developer teams, in collaboration with the LHC Physics Centre at CERN

- What is it?
  - A chance to work closely with MC generator developers on a focused project.
  - Similar to past MCnet-funded short-term placements — but now open more broadly.
  - Not limited to MCnet member nodes — external projects and students welcome.

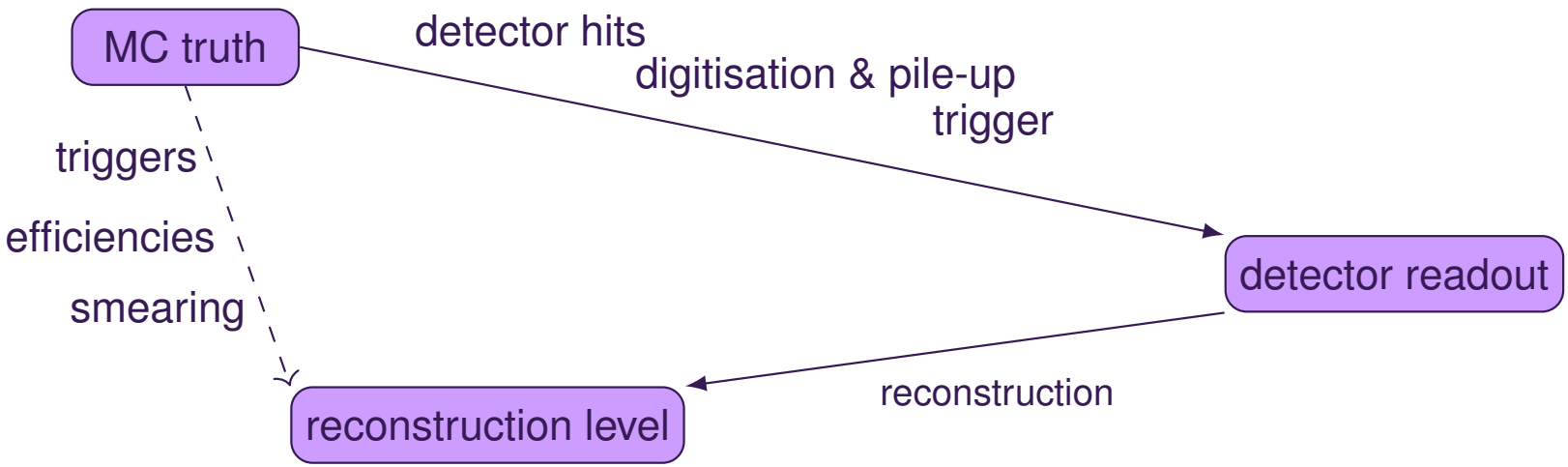
- Who can apply?
  - PhD students interested in event generators, theory/phenomenology, or experimental MC use.
  - Especially suited for students looking to deepen their expertise in LHC simulation tools.

- More info & available projects [click me]
  - New rounds of placements will be announced 2–3 times per year.
  - Feel free to reach out with ideas for new projects – external proposals encouraged!

Backup

Event generators in experiment data processing

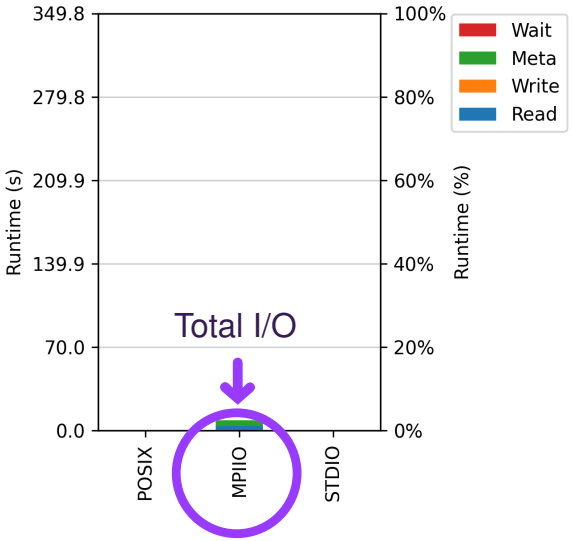
- Typical experimental use of generators is to feed their output into a detector simulation (e.g. based on Geant4)
- Then apply the same reconstruction & analysis as for data:

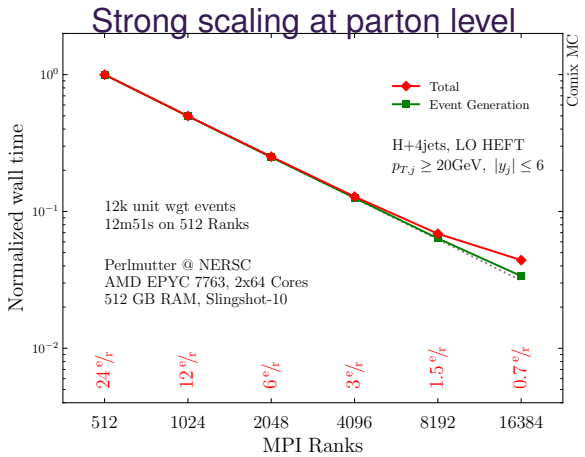


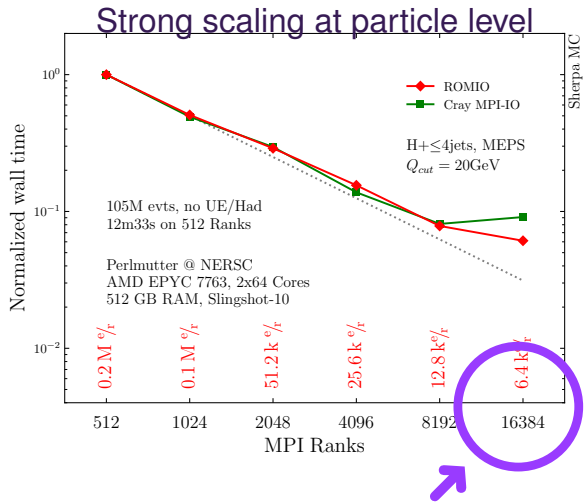
- The generator bit of this chain was historically cheap.
  - CPU/memory requirements much less than detector geometry, *B*-field stepping, material interaction and secondaries.

⚠ **Not true these days!**

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

Tactics for tuning

- Factorise the parameter space
  - Traditionally split hadron flavours/spectra, jet structure, event topologies, underlying event – typically $\mathcal{O}(10)$ parameters
  - Approximate but practical; reduces dimensionality and stabilises fits.
  - Possible to automate grouping via mutual sensitivities / parameter-bin influence matrices.
- Weighting, observable balance, and uncertainties
  - Some data types dominate the fit unless reweighted — balance is essential.
  - Models cannot fully describe all bins: examine envelopes, sensitivity maps, parameter ranges, and consider per-bin weighting.
  - Custom goodness-of-fit is common, but regularisation weakens strict statistical interpretation.
  - Even "χ^2" is non-classical in practice: eigentunes, empirical tolerances. . . still room for more principled approaches.
- Future directions
  - Improved treatment of heavy flavour; incorporating matching/merging systematics.
  - Systematic-uncertainty handling via event weights.
  - More flexible surrogate models and robust optimisation strategies.