Status of Pythia8 at HPC Facilities

Stephen Mrenna
Fermilab

(with extensive help from Jim Kowalkowski)

Outline

- Long and short term goals
- Accomplishments
- Difficulties
- Lessons
- Outlook
- Recap/Conclusions

Long Term Goal

- Event generators, such as Pythia, are a critical component of interpreting data at the energy and intensity frontiers.
- Many parameters are needed as an input to describe the complex structure of particle collisions.
- The goal of this project is to exploit the computing power of Mira to perform an extensive parameter scan
- Wish to maximize the agreement between benchmark data and the generator predictions.

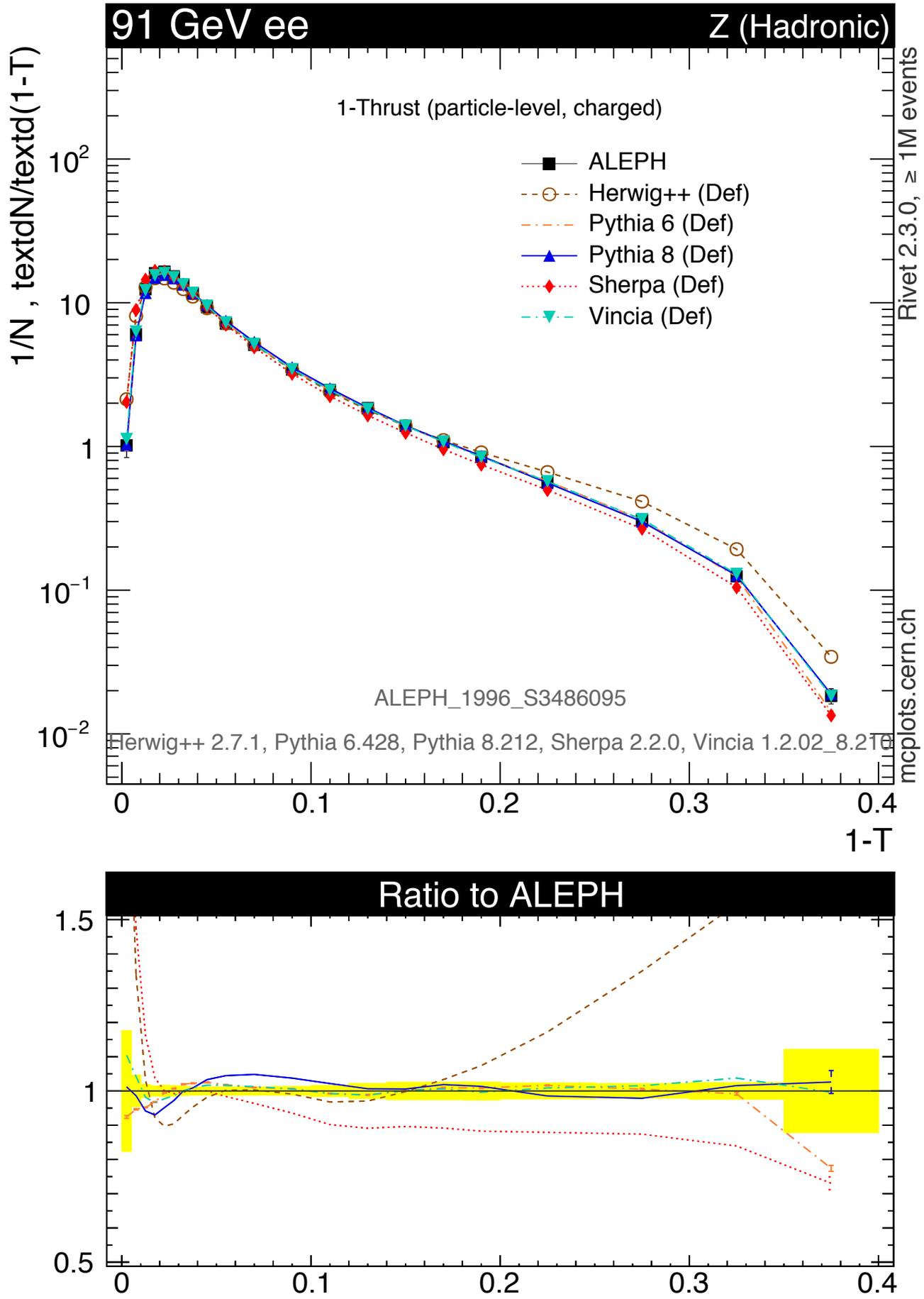
What is Generator Tuning?

Although this may seem a paradox, all exact science is dominated by the idea of approximation

- Bertrand Russell

- Tuning == using data to fix the dependence on models/cutoffs/order of approximation …

MCplots Example



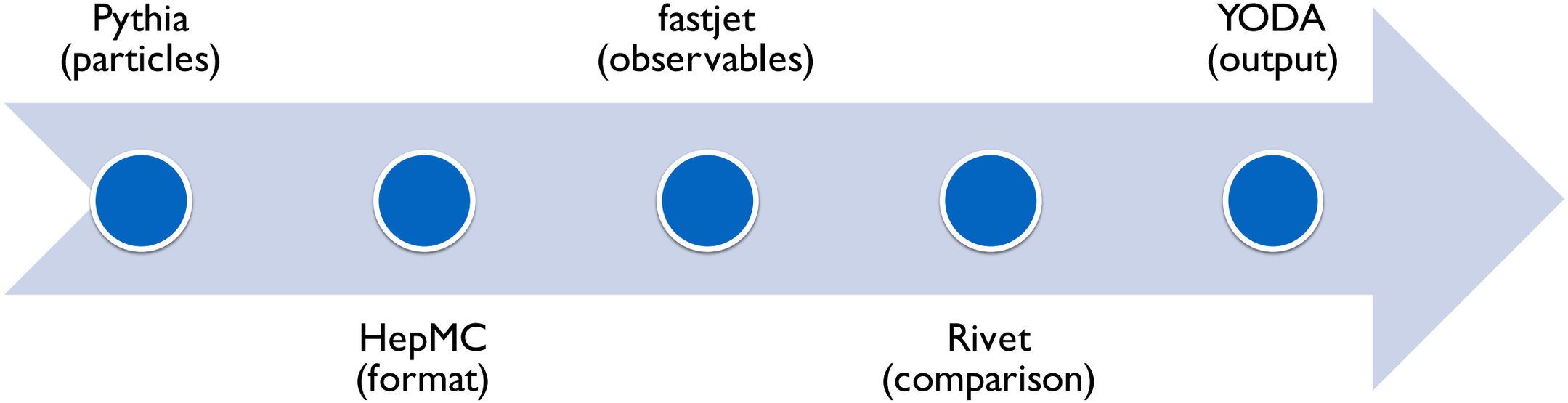
Short Term Goal(s)

- Understand the problem(s) of adapting community tools to HPC clusters
- Experiment with ways of submitting jobs, passing input, and gathering output
- Think about code design

Statement of the Problem

- Generator tuning using community tools:
  - Rivet, HepMC, fastjet, YODA
    - How to build them? Thread-safety?
- Broadcasting runtime configurations:
  - Pythia parameters, PDF data, Rivet analyses
- Collecting output
  - YODA files
- (Perform output analysis (tuning) elsewhere)

Workflow


Pythia
(particles)
fastjet
(observables)
YODA
(output)
HepMC
(format)
Rivet
(comparison)

Lessons from Alpgen Project

- I/O can be a killer
  - Avoid many jobs reading from the same files
    - Broadcast data from a master job
  - Avoid all jobs writing to disk at once
    - use a nested tree to accumulate results
- Keep jobs within walltime

Accomplishments

- Changes to Pythia8 constructors/inits to accept strings of data (configurations, PDFs, …)
- Built HEP community software using BGClang with C++11: Rivet, fastjet, HepMC, YODA
- Constructed MPI-based main_xx.exe to manage broadcast, joining, gathering of data
- Run toy jobs to exercise comparing to Rivet analyses
- Ready to perform physics run to reproduce previous tunes
- Will soon perform new, granular tune

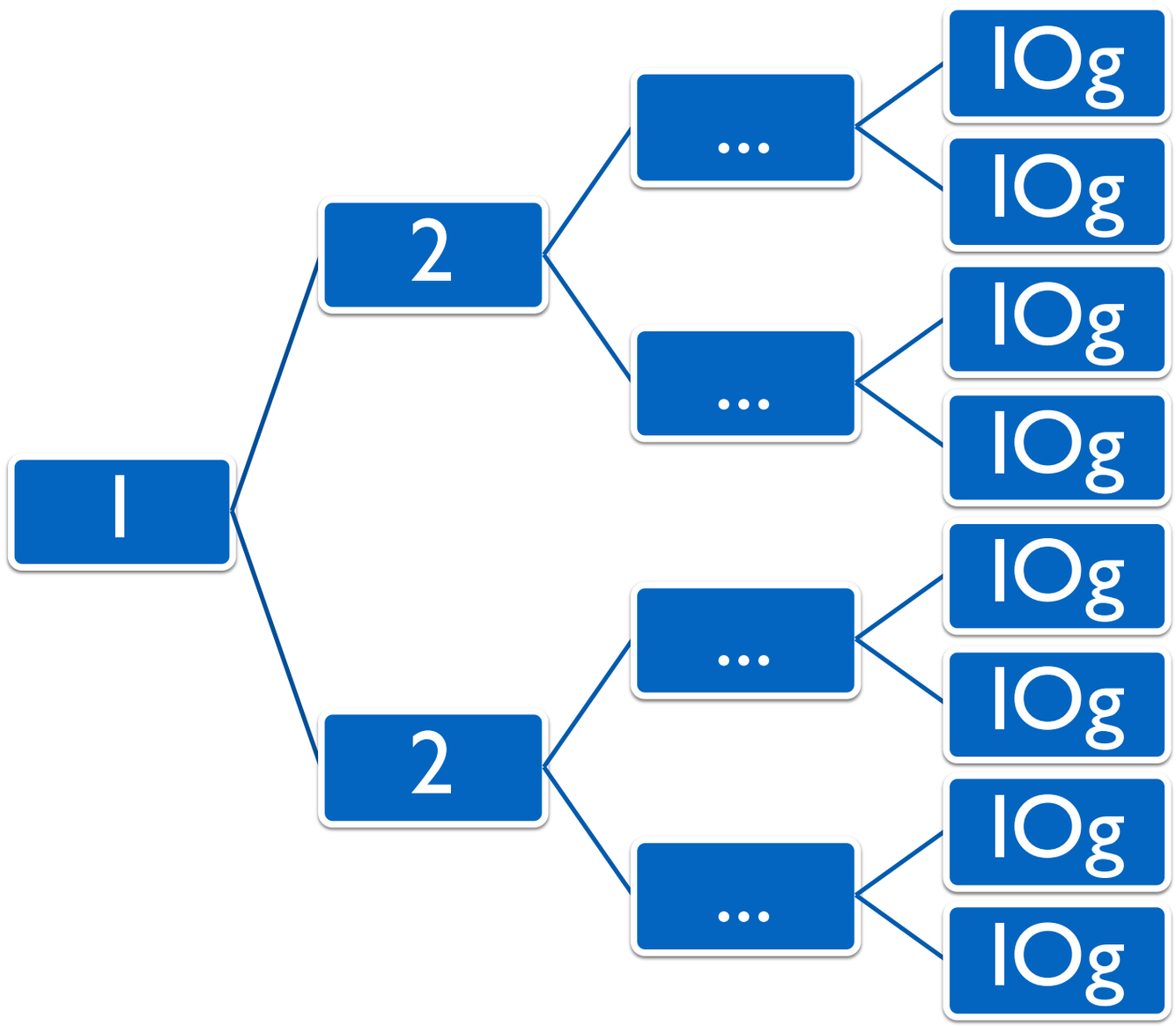
Challenges

- Making changes to Pythia8 was straightforward
- Building external tools quite challenging
  - Most based on autotools – not well adapted to ANL machi1es: extensive, tiresome hacking!
    - Found a compiler bug!
- needed CMS thread-safe version of fastjet
- Even then: spurious results
  - Debugging on CETUS, e.g., not trivial
  - Autopsy of core dumps and output information

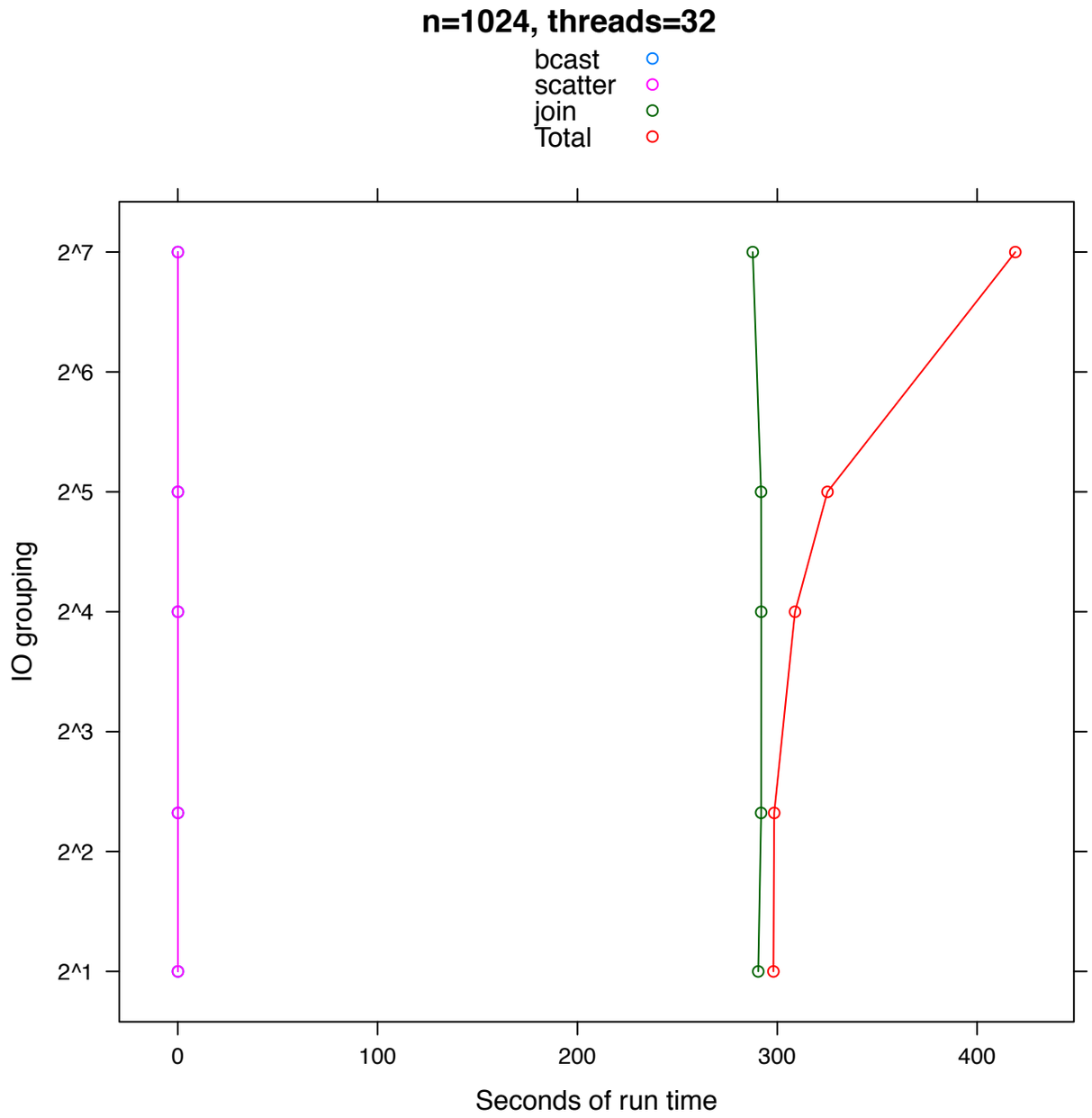
Lessons

- Clashes all related to initialization of Rivet routines
  - Handled with <mutex> in main_xx.cc
  - Some small hacks to Rivet core code
- note: we are handling the threading ourselves using standard std::thread features
  - Performance loss if n_threads not matched to hardware
- YODA output (strings) accumulated in groups of nodes (iogroups)
  - Doesn't quite work how we expect
- Pythia is rather verbose – these needs to be shutoff

Structure of IO groupings



Timing Results for CETUS runs



Difference btw Red and Green comes from collecting YODA (tuning) data

Small impact for 5 < iogroups < 16

Outlook

- Ready to perform realistic parameter scans
- The ability to reweight parton showers (see recent paper with P. Skands) reduces the number of parameters to be scanned
- A quick turn-around on tuning is important NOT just for new data, but for improved physics tools
  - Better perturbative physics → less dependence on non-perturbative models/parameters

Recap/Conclusions

- Full machinery of Pythia + Rivet is working on ANL HPC facility using MPI tools
- Current build tools not ideal
- Current code design might not be ideal
- Ready for useful physics results