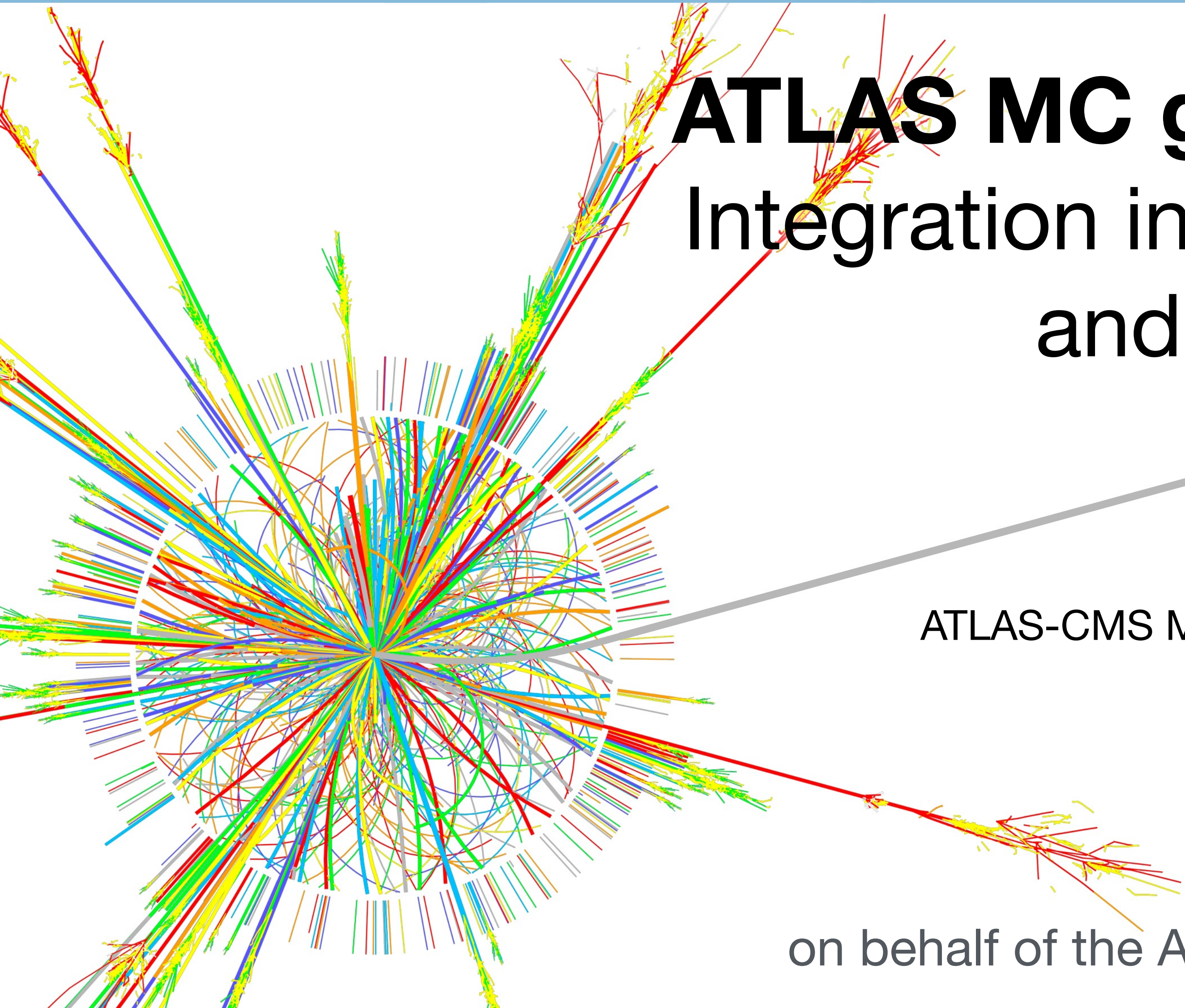


ATLAS MC generation
Integration into software and production

ATLAS-CMS MC Generator Workshop
11/1/2016

Josh McFayden

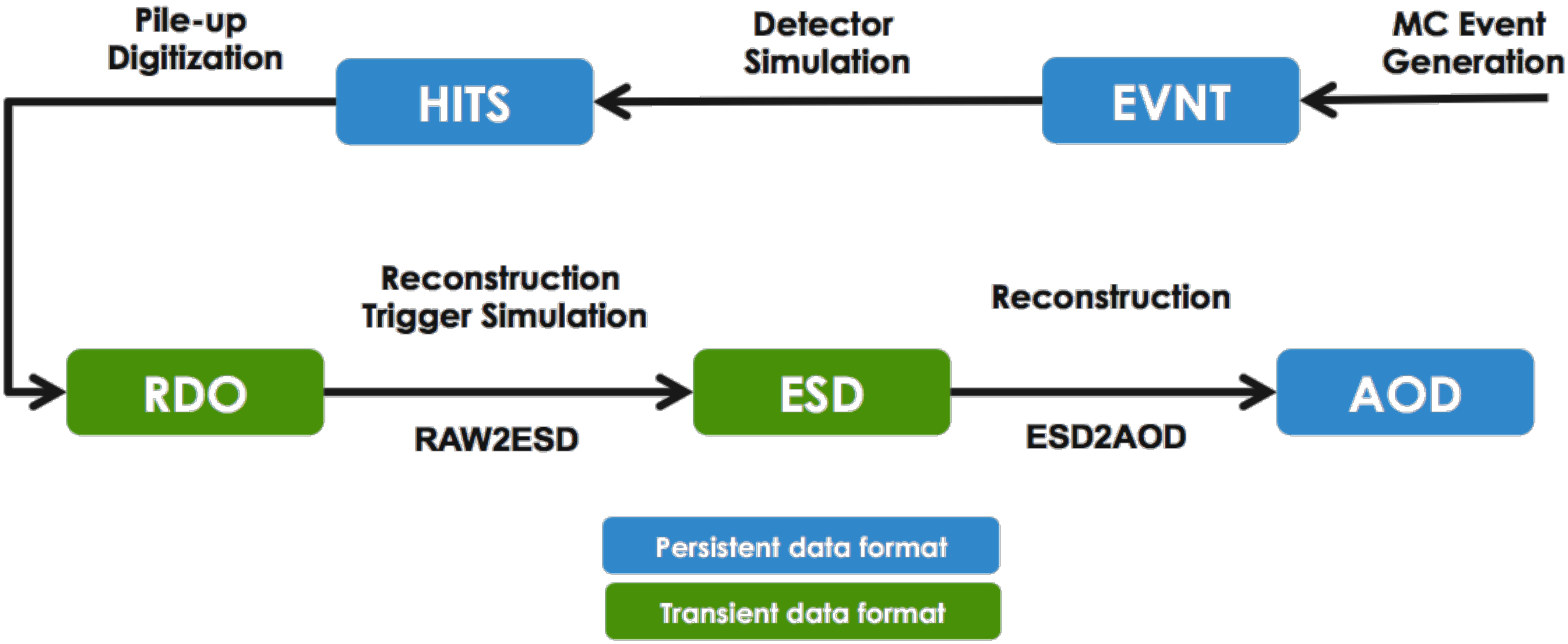
on behalf of the ATLAS Collaboration

Overview

- ATLAS Production System (ProdSys)
  - MC generation in the production system
- MC Generator Software Interfaces
  - Software integration and running modes
- MC Generator Validation
  - A closer look at validation procedures
- Analytics of MC generator usage
  - Production modes & generator types & their CPU consumption

ProdSys | Overview

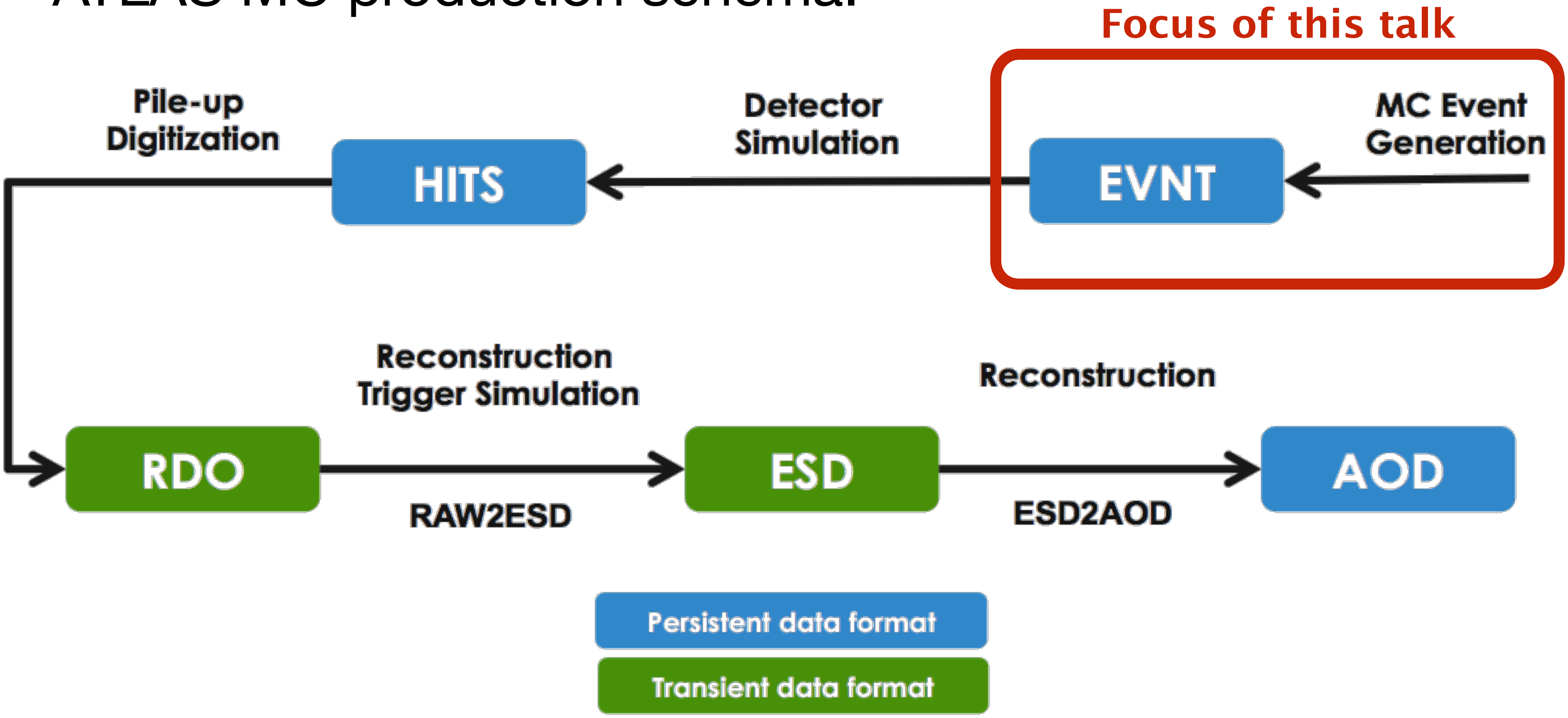
- ATLAS MC production schema:



- During the shutdown several upgrades were made to both the production system and the event data model.

ProdSys | Overview

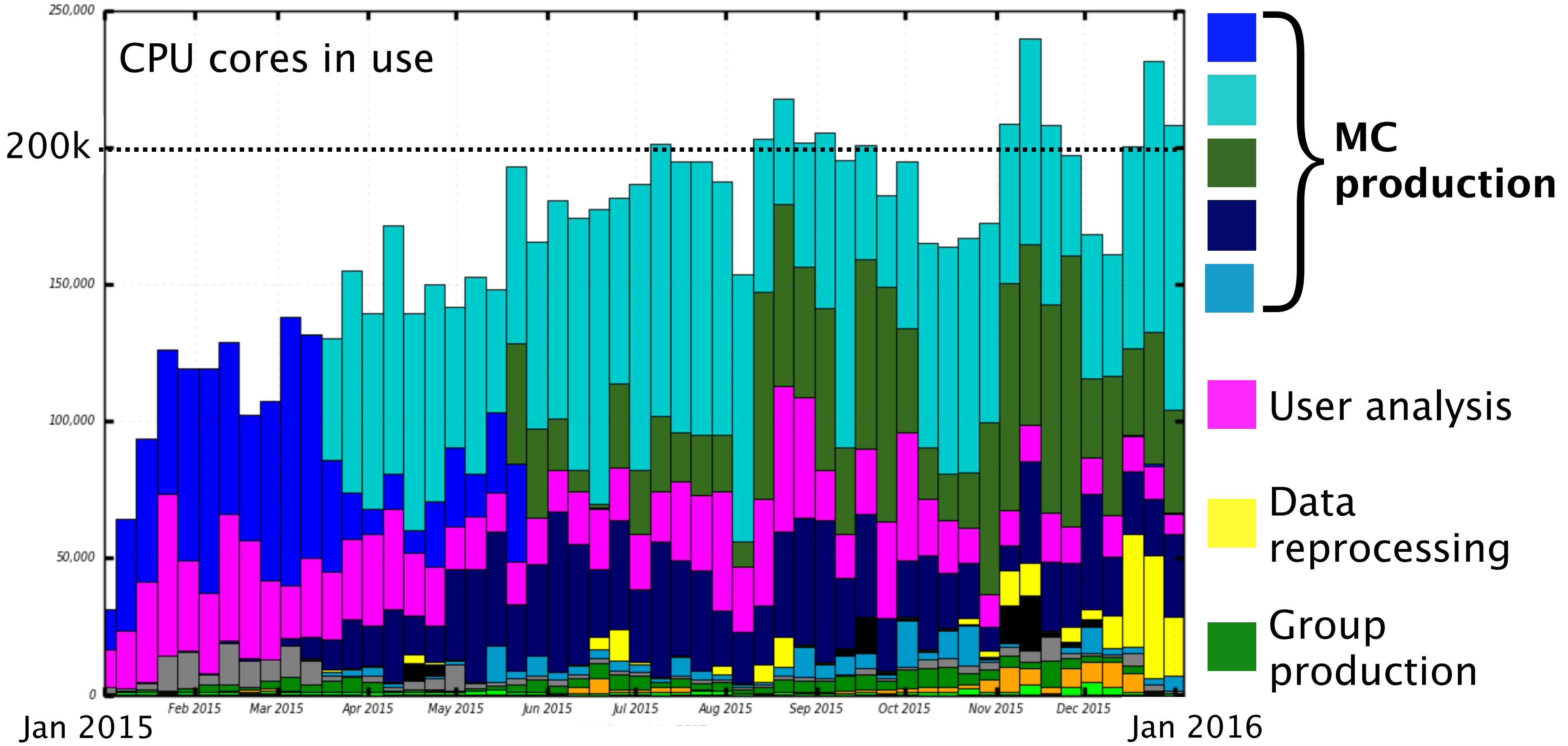
- ATLAS MC production schema:



- During the shutdown several upgrades were made to both the production system and the event data model.

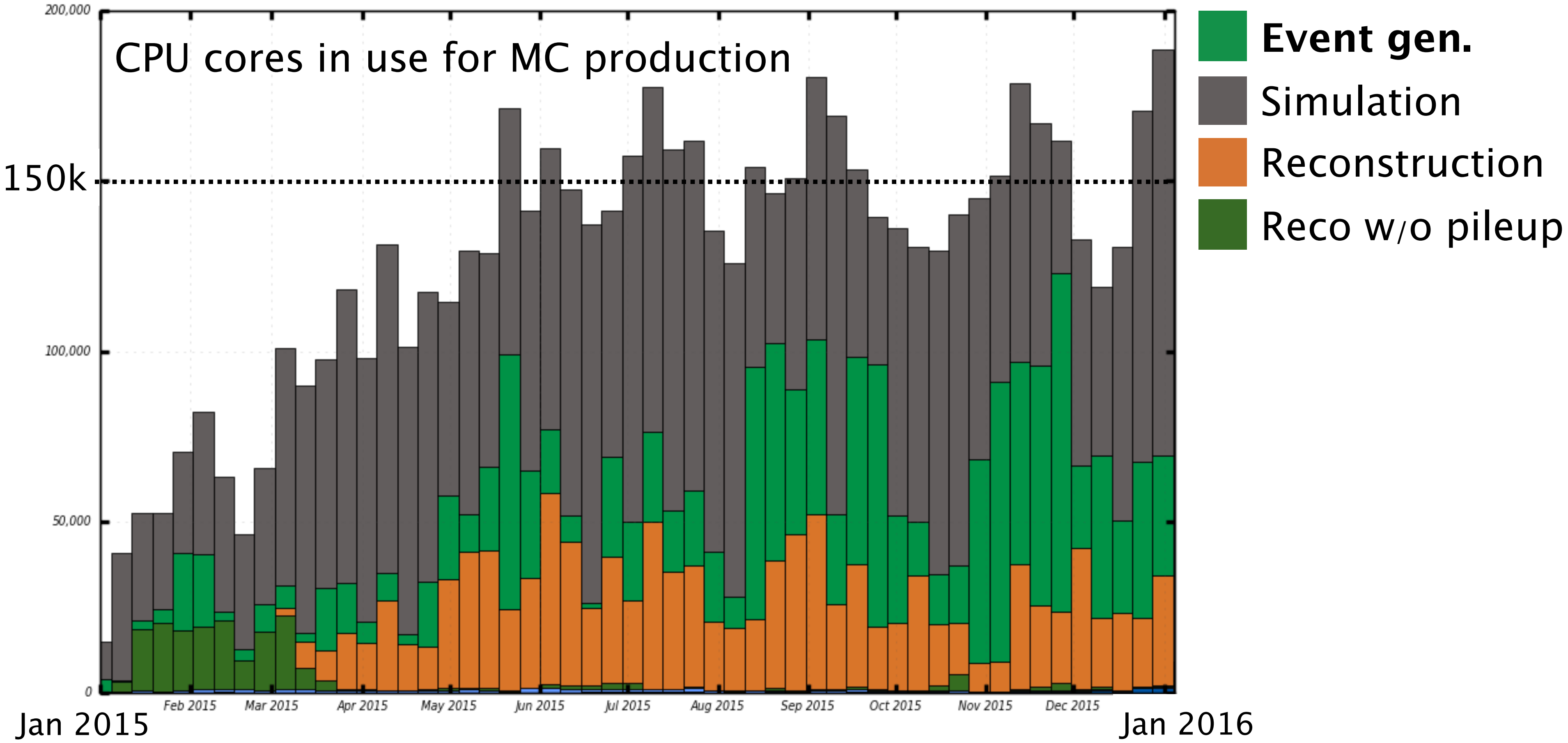
ProdSys | Resources - All

- ATLAS Production System overview.
  - Most of ATLAS's CPU consumption dedicated to MC production.



ProdSys | Resources - MC

- The majority of MC production resources is taken by simulation
    - But a significant fraction is event generation.

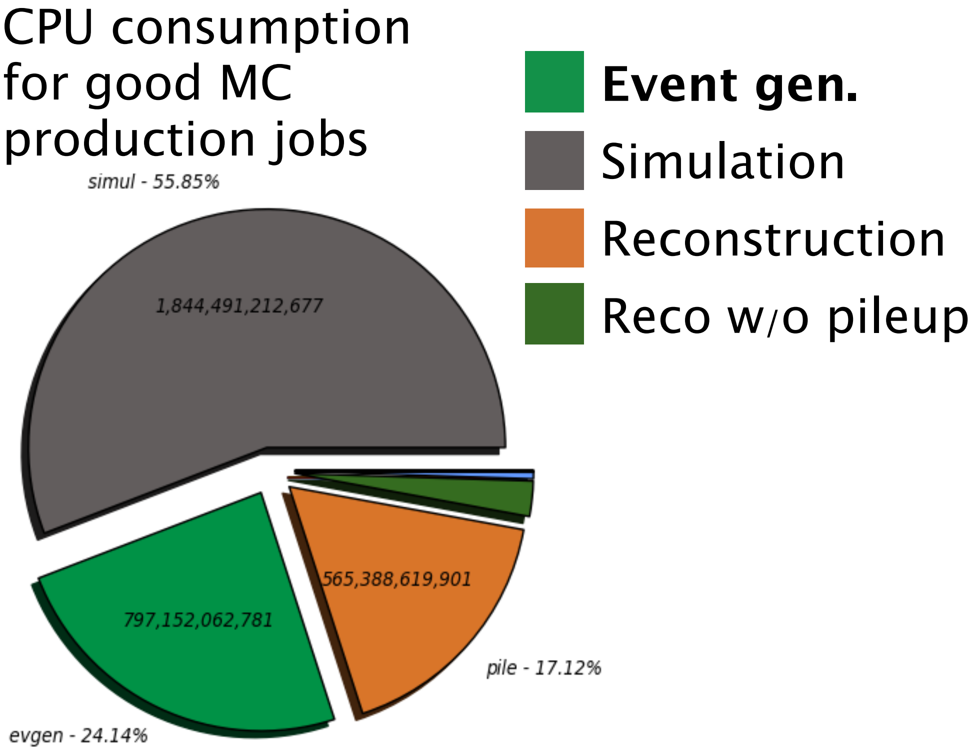


ProdSys | Resources - Evgen

- Over **7 billion** events were generated in the most recent MC production campaign.
  - ~25% of CPU consumption dedicated to MC event generation.

- Average CPU/event is ~90 s for event generation
  - Comparable to ~380 s for full simulation especially in the tails!

- Event generation is the first step in the production chain
  - If that is wrong then so is everything else.
    - **VALIDATION IS CRUCIAL**
    - Potential to waste huge amounts of CPU!



CPU consumption for good MC production jobs

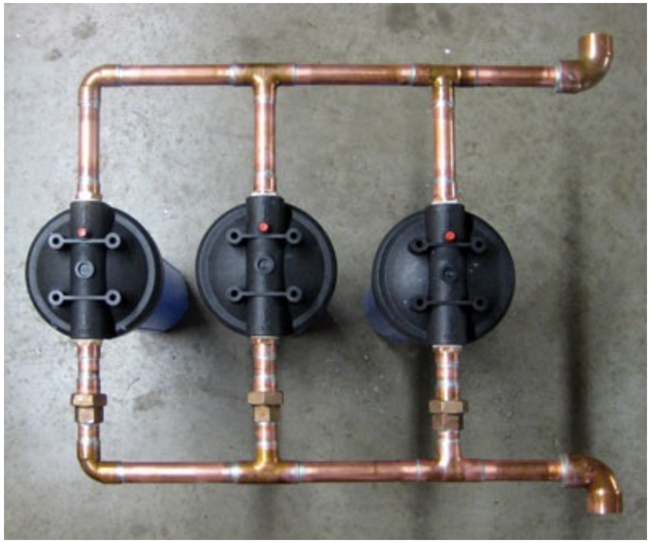
ProdSys | Configuration

- Typical configuration for event generation:
  - **Single core, 24hr job limit, 5000 events/job**
    - Multicore might soon be essential for more CPU intensive processes (many final state particles, low filter efficiency)
    - Can be necessary to reduce number of events/job to fit 24hr time limit.

- Use of high performance computing (HPC) clusters
  - **MIRA**: used to generate **Alpgen** 4-vectors (in progress) and **Sherpa** integration grids (W/Z systematic uncertainties).
  - Significant CPU consumption required for complicated processes e.g. V+2,3j@NLO
    - Not possible on most local clusters.
  - Better integration with HPC clusters would make such processes more accessible.
    - Not currently using Athena for HPC - have to be very careful with validation. Lightweight Athena release in preparation for easier integration.

Interface | General

- Many different types and combinations of generators
  - Matrix element only
    - MG5_aMC, Powheg, Alpgen, …
  - Parton shower
    - Pythia8, Herwig++
  - Combined
    - Sherpa
    - Herwig7



- Interface between ME and PS can be hardest part of configuration to get right
  - MG5_aMC+Py8, MG5_aMC+HWpp, Powheg+Py8, Powheg+HWpp, etc.

- Various possible configurations result in many different running modes
  - Also requires flexibility in the software integration and production system configuration.

Interface | Software integration

- All generators are external packages so some integration into ATLAS software is required.
  - But not always simple!

- GENSER generator installations
  - Use precompiled generators from GENSER ensure that the same code is used by all LHC experiments and to perform common testing.
  - Used for Sherpa, Herwig++, Pythia8
- Still a layer of C++ wrapping to integrate into Athena
  - E.g. Pythia8_i
    - Like a bare main file with UserHooks that can be loaded in.
- Different interfaces for ME-only, e.g MG5_aMC and Powheg:
  - MadGraphControl - MG5_aMC versions installed by hand.
  - AlpgenControl - Similar to MGControl, not well used in 2015
  - PowhegControl - Powheg modules installed by hand.

Interface | OTF interfaces

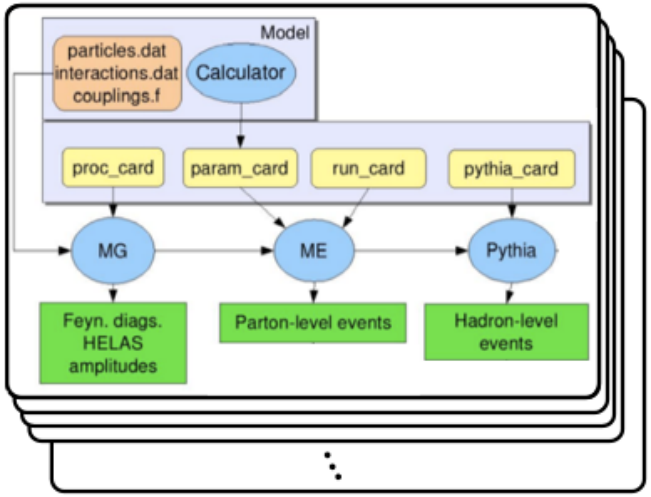
- On-the-fly (OTF) event generation interfaces are used for MG5_aMC and Powheg - *MadGraphControl* and *PowhegControl*
  - Athena initially designed for LHE inputs or a single parton shower run
    - **Adapted to add LHE event generation and showering all in one run = OTF**
  - Provides users with semi-automated interface, with default configurations provided. Python is used for all steering.

- MadGraphControl
  - Example configurations are flexible so that users have freedom to define new processes safely and with minimal effort.
  - Easy to define process and run_card parameters for new sample, and then use predefined shower configurations.
- PowhegControl
  - Default configurations are provided for number of modules
    - Including optimised integration parameters
  - Can have issues with certain processes: e.g trijet O(weeks) to generate

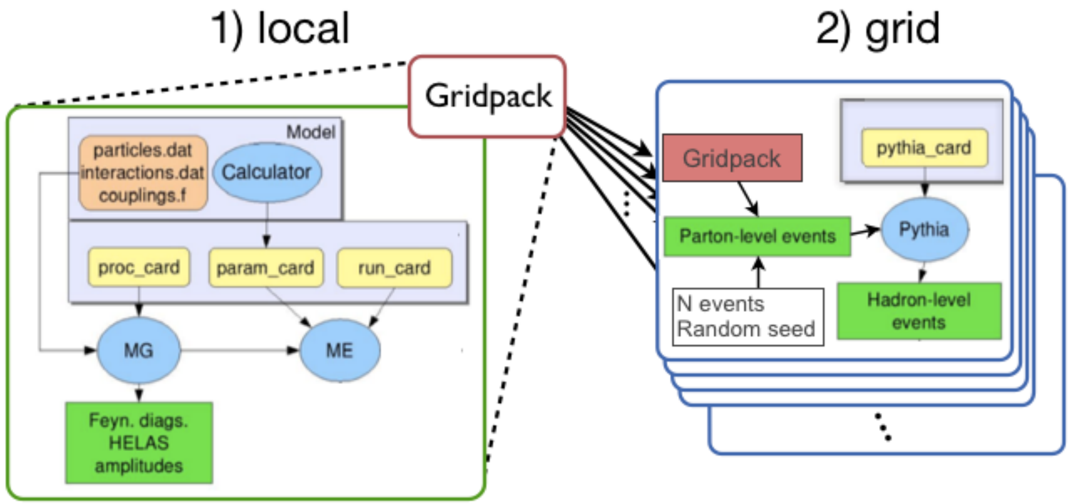
Interface | Production modes

- **LHE** - LHE file 4-vectors produced by external codes.
  - More danger of unvalidated configurations and lack of reproducibility.
  - Slightly more flexibility with generating complex processes on e.g. local cluster.
- **OTF** - LHE event generation and showering all in one grid job
  - ATLAS recommended (and validated) settings applied by default.
  - This is the preferred mode for ME+PS configurations.
- **Integration grids** - Use OTF interface for *local* ME calculation & integration.
  - Package into "**gridpack**" for jobs input - only event gen. and shower run in ProdSys.
  - Able to get around the 24hr grid job limit.

On-the-fly running:



Gridpack running:



Validation | Overview

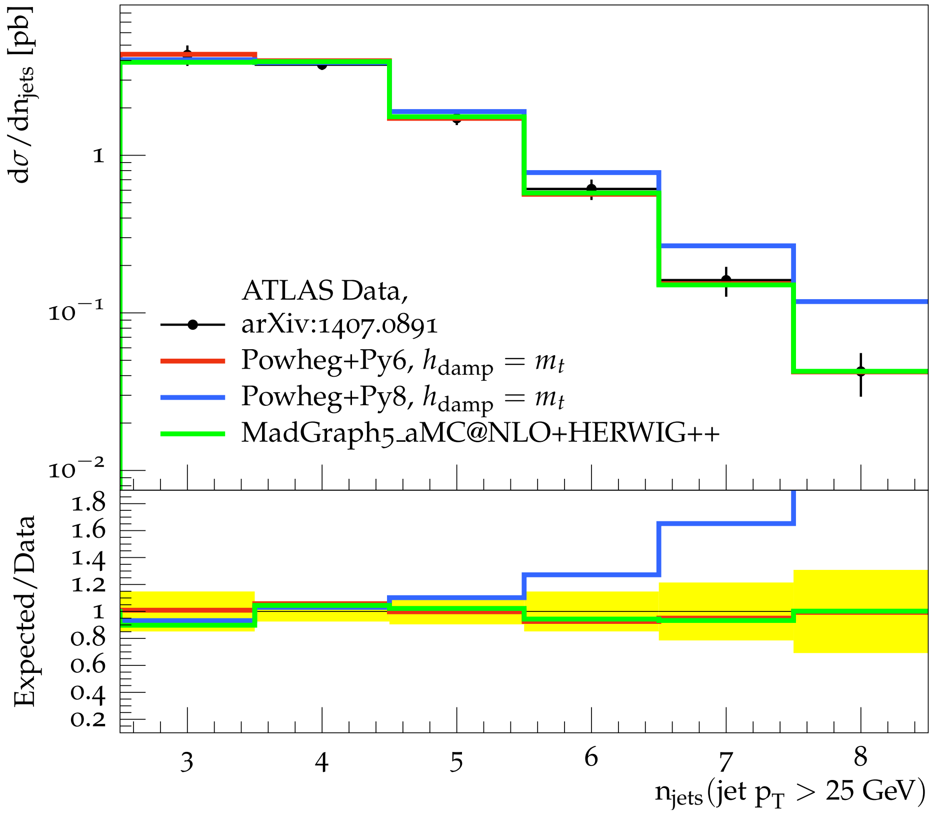
There are many layers of generator and process validation in ATLAS:

- **Physics validation**
  - New generators/generator setups are validated against data from SM measurements, e.g. V+jets & ttbar, by physics groups and the Physics Modelling Group (PMG).
- **Technical validation**
  - More "automated" technical validation is performed for smaller changes
    - changes to modelling of a specific sub-component in MC setup.
    - Minor MC generator revision for already validated generator.
  - Samples are passed through histogramming code that looks at both LHE (pre-shower) and HepMC (post-shower) variables.
- **Sample request validation**
  - Validation plots and log files are required in requests for new samples.

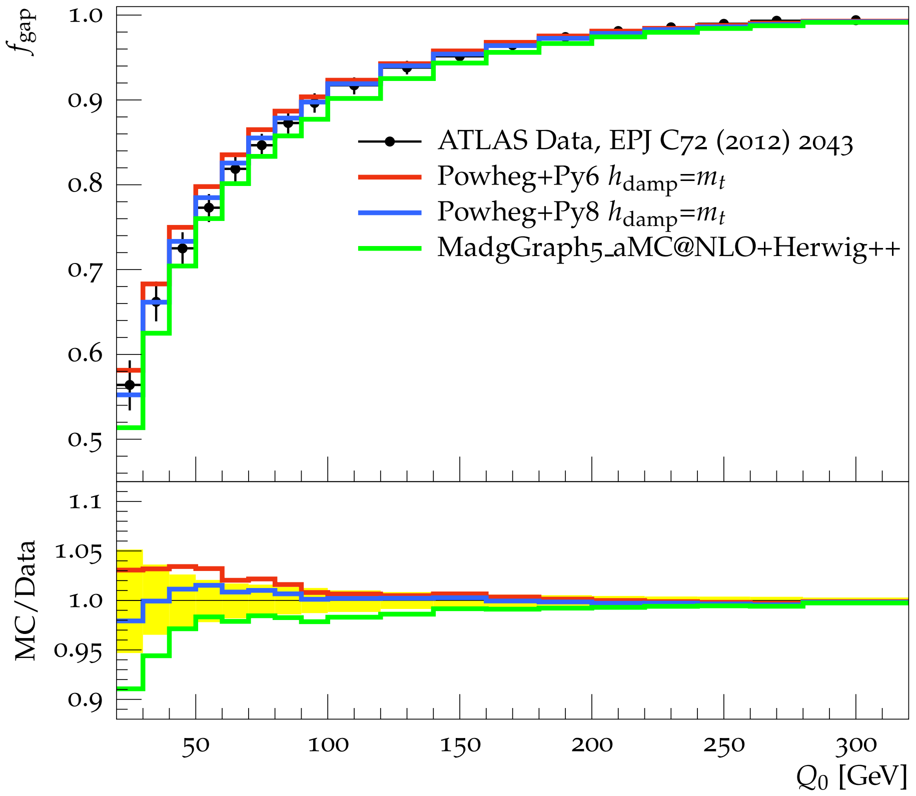
Validation | Physics

- Physics validation is usually the first step for major generator changes or brand new configurations.
  - Use existing measurements to validate generator output against data.
  - Performed by physics groups in conduction with PMG.

$t\bar{t}$ cross-section vs. jet multiplicity for jets above 25 GeV

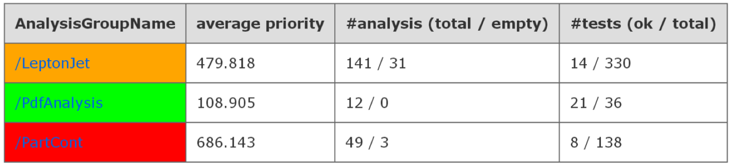


Gap fraction vs. Q_0 for veto region: $|y| < 2.1$

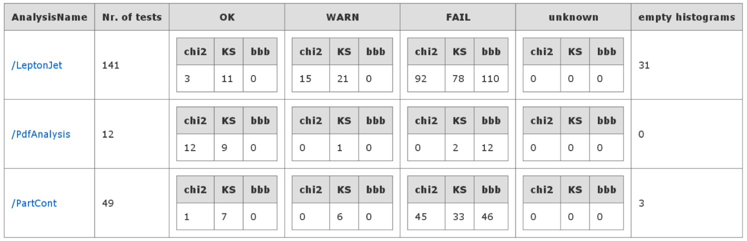


Validation | Technical

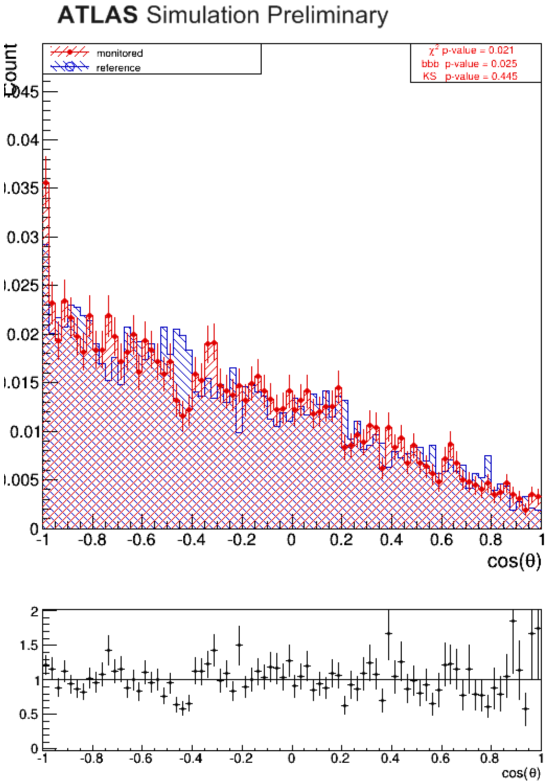
- A web interface, the *Job Execution Monitor* (JEM), is used to configure and display predefined sets of monitored and reference samples
  - *HepMCAnalysis* validation tools and provides a histogram-based output.
  - The agreement between histograms is quantified with statistical tests
    - Kolmogorov-Smirnov, Pearson's χ2 and a bin-by-bin method.
  - Information about the outcome is displayed in a colour-coded summary table.



AnalysisGroupName	average priority	#analysis (total / empty)	#tests (ok / total)
🟧 /LeptonJet	479.818	141 / 31	14 / 330
🟩 /PdfAnalysis	108.905	12 / 0	21 / 36
🟥 /PartCont	686.143	49 / 3	8 / 138

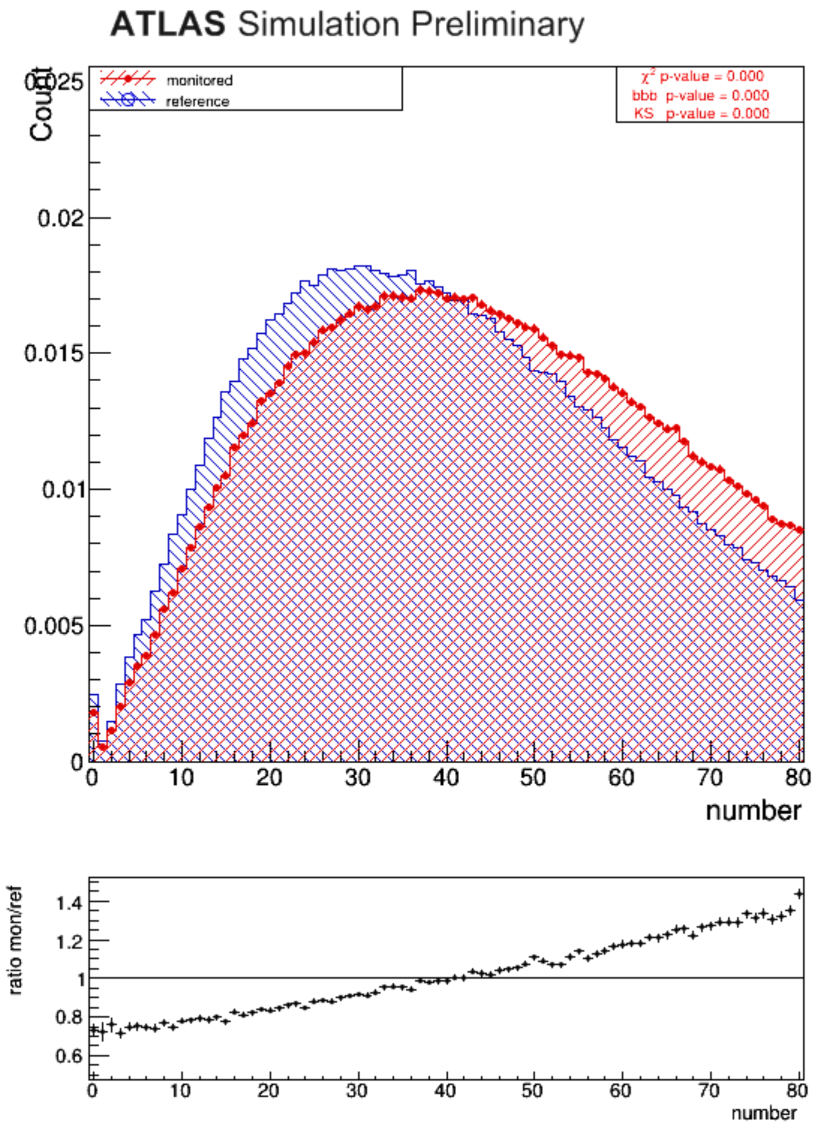


AnalysisName	Nr. of tests	OK			WARN			FAIL			unknown			empty histograms
		chi2	KS	bbb	chi2	KS	bbb	chi2	KS	bbb	chi2	KS	bbb	
/LeptonJet	141	3	11	0	15	21	0	92	78	110	0	0	0	31
/PdfAnalysis	12	12	9	0	0	1	0	0	2	12	0	0	0	0
/PartCont	49	1	7	0	0	6	0	45	33	46	0	0	0	3



Validation | Technical

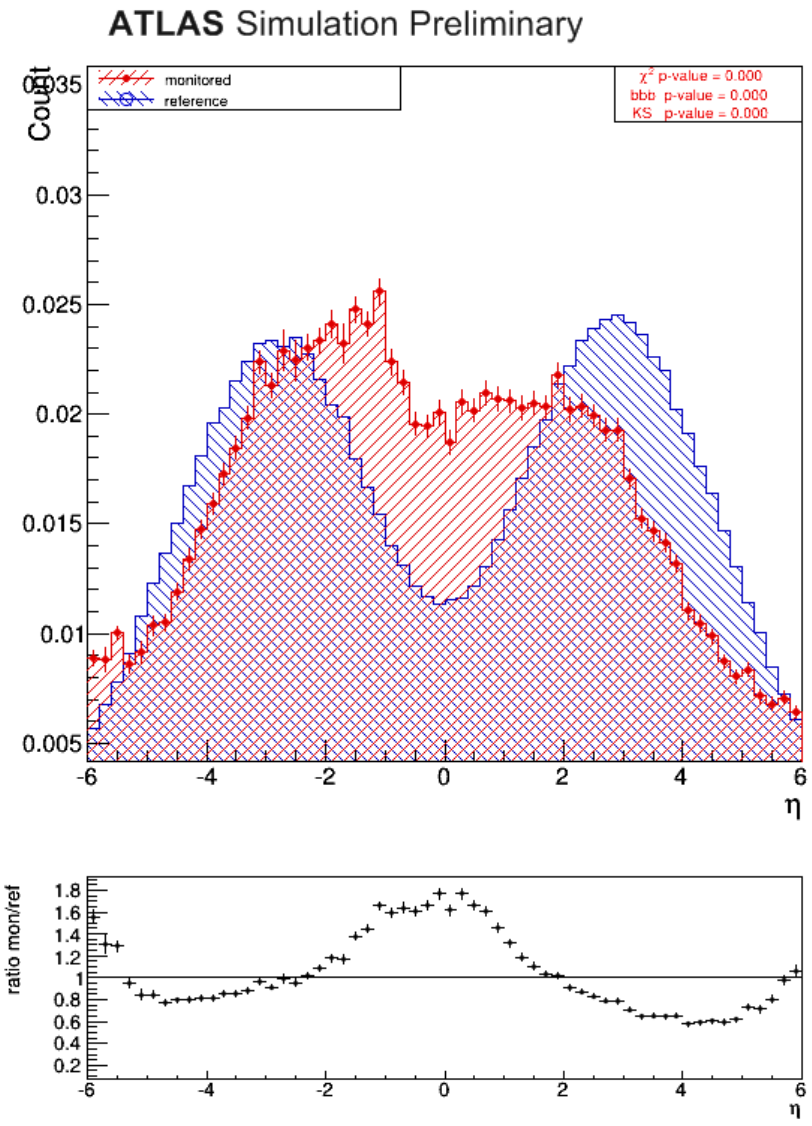
- Case studies:
  - Herwig++
    - Validation of **v2.7.1 with UE-EE5 tune** wrt the previous version **v2.6.3 with UE-EE4 tune**.
    - Differences seen in number of strange mesons in Z→ee events.
      - Mainly due to the new UE-EE5 tune.



Validation | Technical

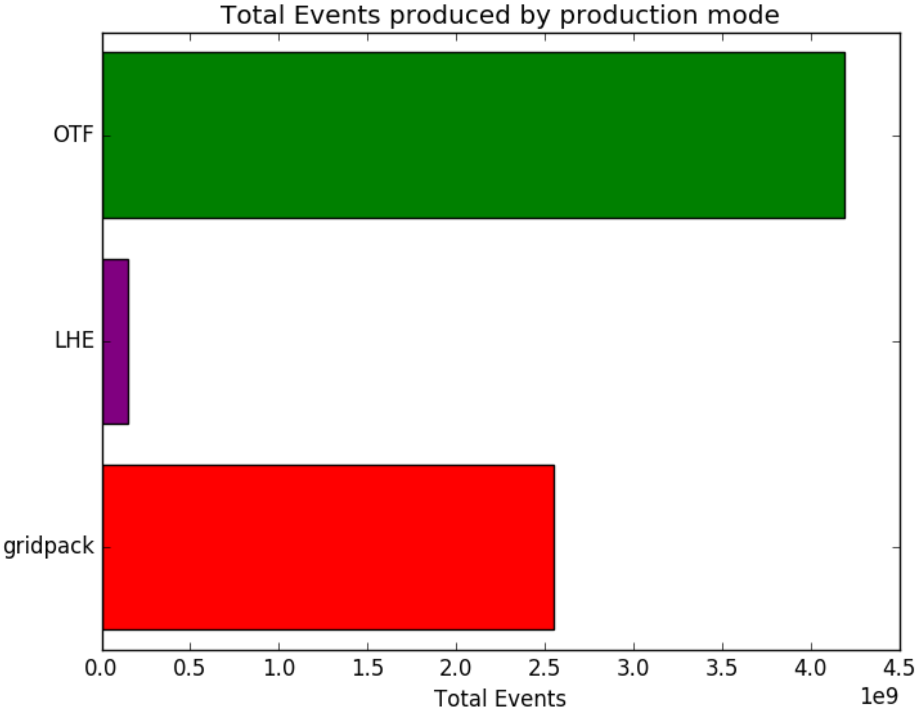
- Case studies:
  - Herwig++
    - Validation of **v2.7.1 with UE-EE5 tune** wrt the previous version **v2.6.3 with UE-EE4 tune**.
    - Differences seen in number of strange mesons in Z→ee events.
      - Mainly due to the new UE-EE5 tune.
  - Sherpa2.1
    - Validation of **v2.1** wrt **v2.0**
    - Differences seen in B-hadrons η
      - Issue in the MPI matrix elements.

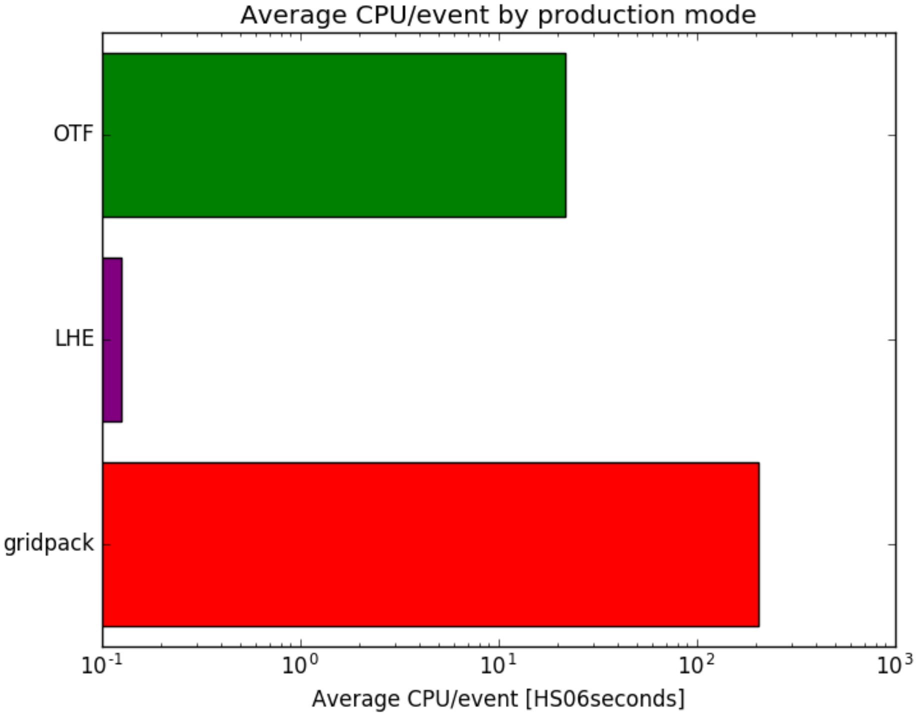
        Fix implemented by the authors in a subsequent 2.1.1 patch release.



Analytics | Production modes

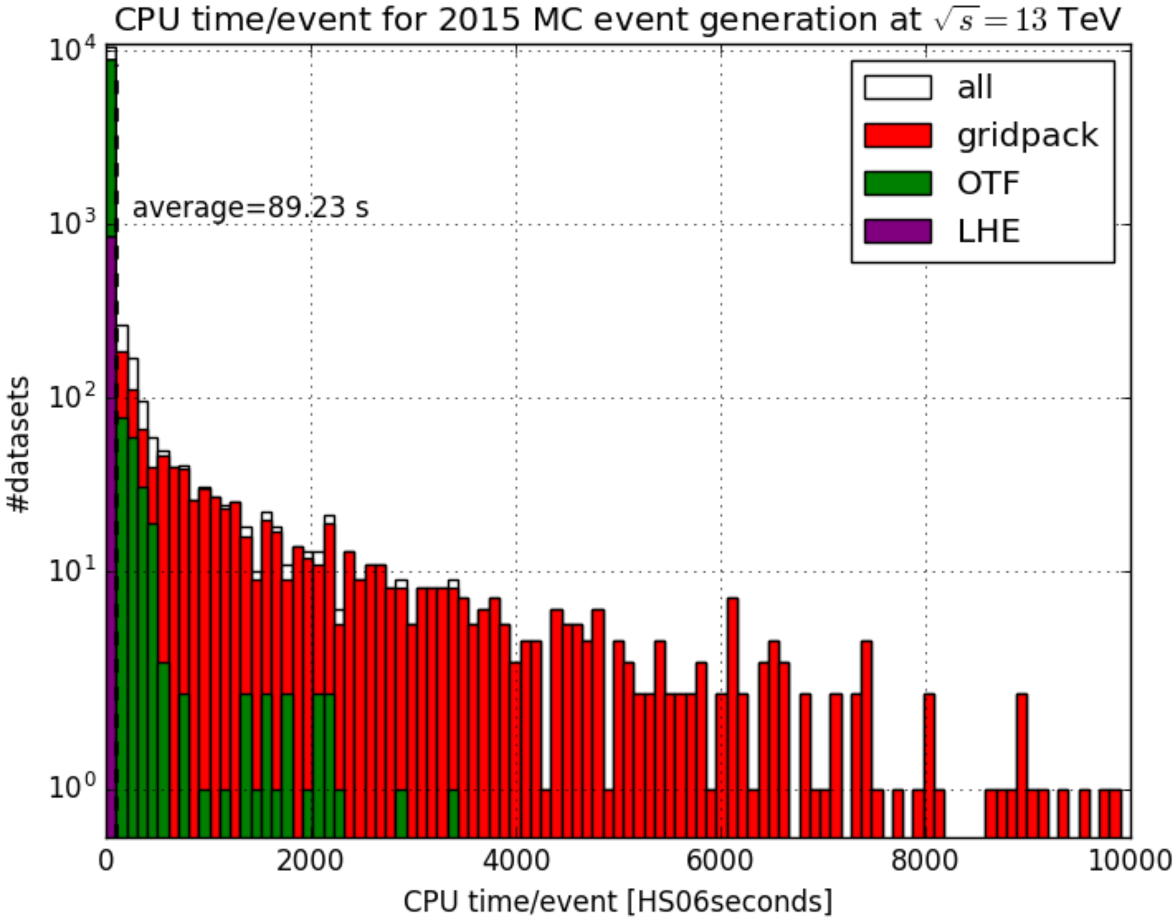
- **OTF** is now used for the majority of samples.
  - Close to 24hr limit with 5000 events/job
- More CPU time consuming processes use **integration grids**.
  - E.g. Sherpa V+jets with 2j@NLO & 4j@LO.
- **LHE** files still used but much less common.
  - Very different picture compared to Run 1.





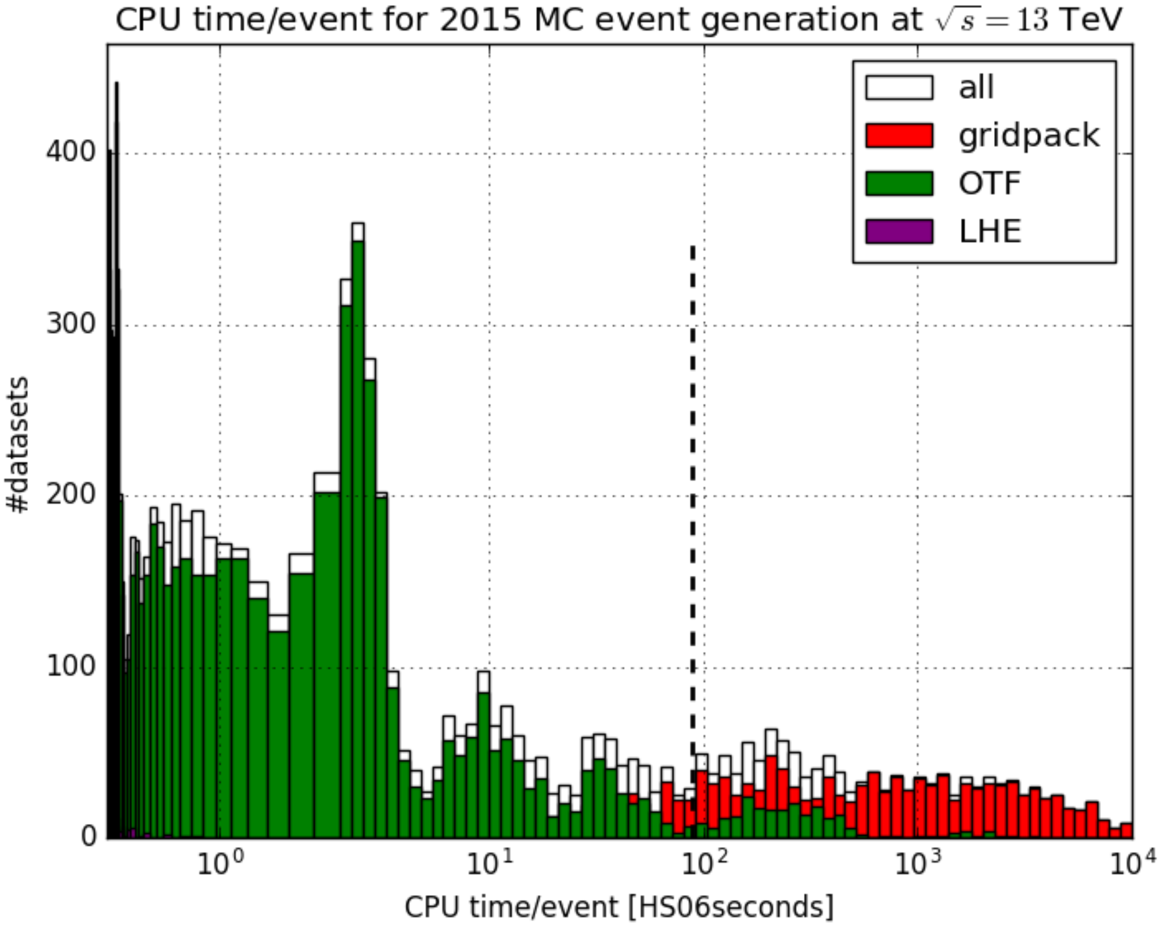
Analytics | Production modes

- The average CPU time required ~90 HepSpec06 seconds/event
- Although most processes require less than the average there are significant tails.



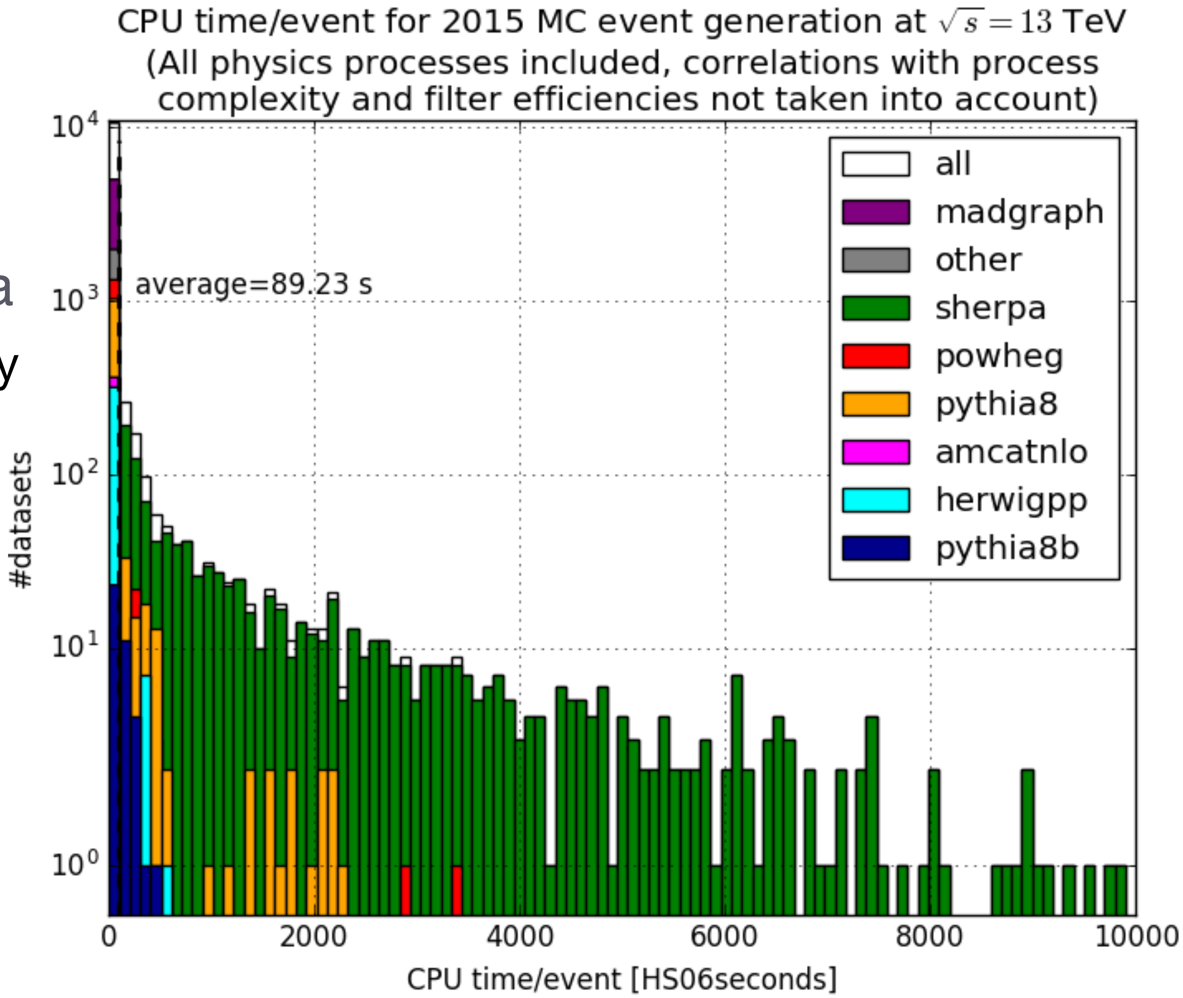
Analytics | Production modes

- The average CPU time required ~90 HepSpec06 seconds/event
- Although most processes require less than the average there are significant tails.
  - Mostly use integration grids to overcome grid CPU limits



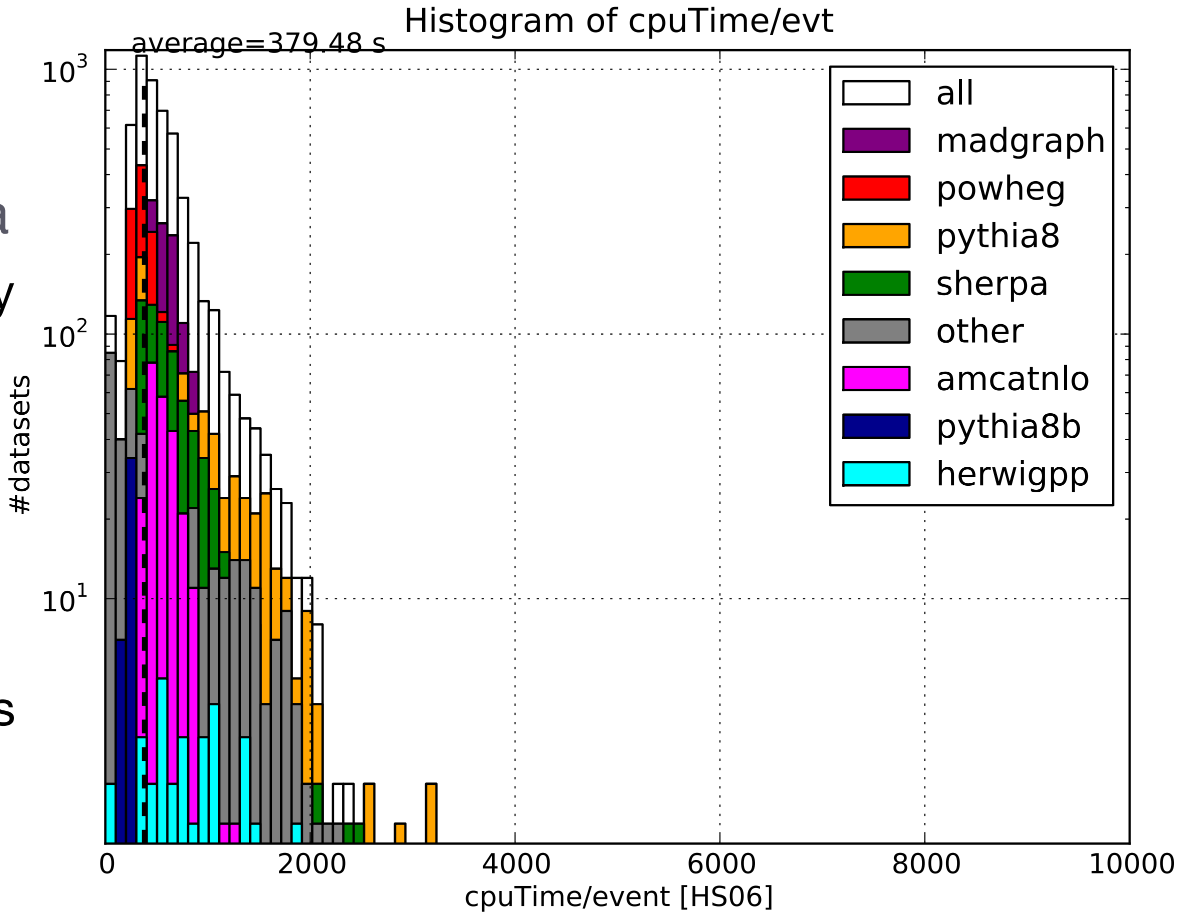
Analytics | Generators

- The average CPU time required ~90 HepSpec06 seconds/event
- Although most processes require less than the average there are significant tails.
  - Mostly use integration grids to overcome grid CPU limits
  - Tails come from Sherpa
    - NLO processes and very low filter efficiencies



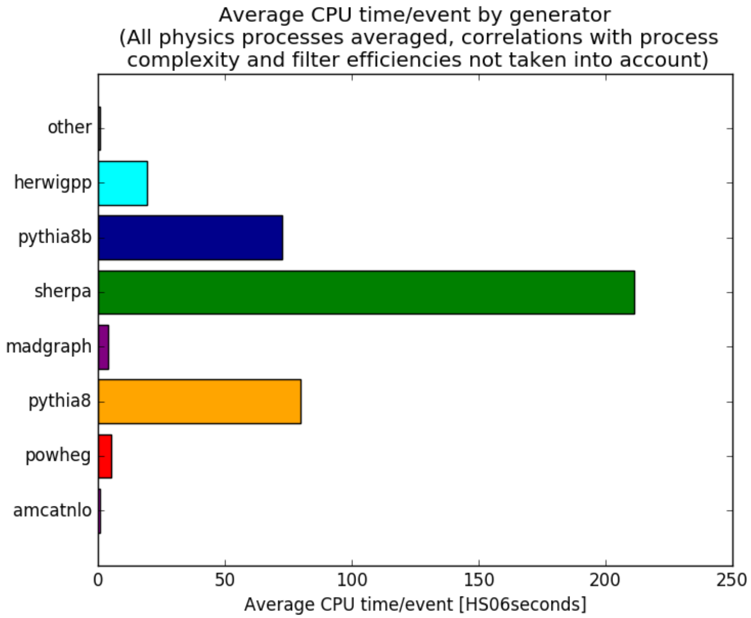
Analytics | Simulation

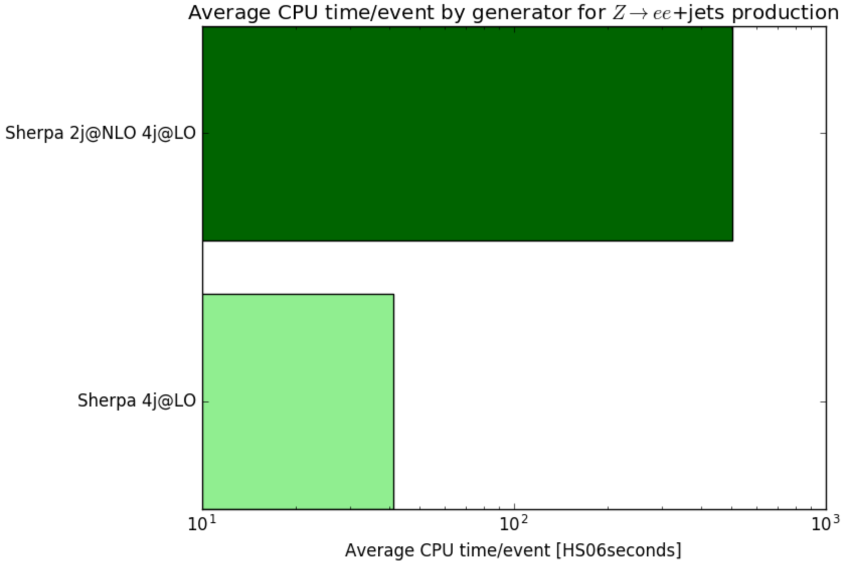
- The average CPU time required ~90 HepSpec06 seconds/event
- Although most processes require less than the average there are significant tails.
  - Mostly use integration grids to overcome grid CPU limits
  - Tails come from Sherpa
    - NLO processes and very low filter efficiencies
  - Starts becoming more CPU intensive than full detector simulation
    - Unsustainable, with significantly more events required at high lumi.
    - HPC can help here.

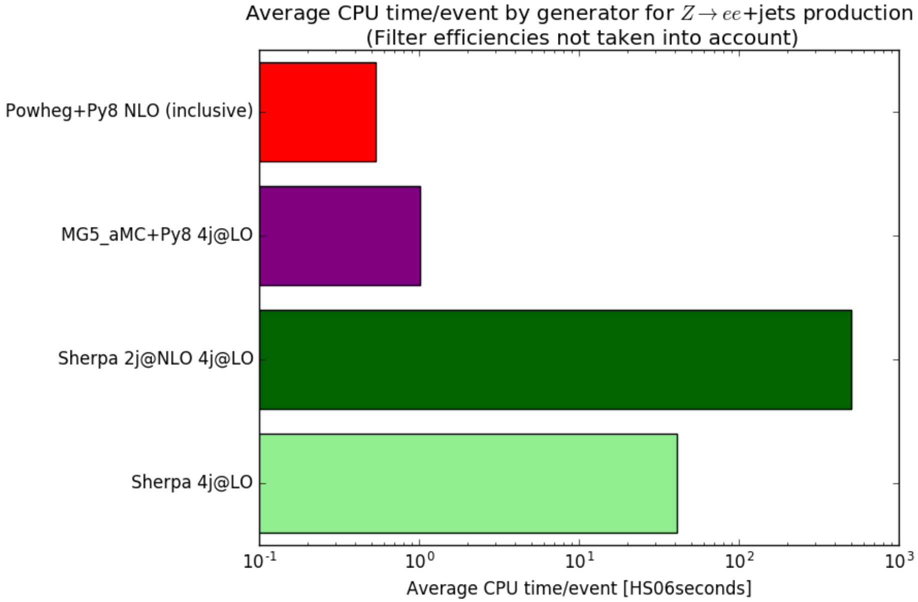


Analytics | Generator overview

- Average CPU time/event
- Sherpa has the highest CPU time/event. But this is dominated by the fact that it is generating the most CPU intensive process with some of the lowest filter efficiencies.
  - NLO significantly more CPU consuming than LO.
  - B- and C-hadron filters have large effect.







Other technical issues

- Use of filtering for generation
  - ME level cuts necessary on e.g. $p_T(V)$, H_T to get sufficient coverage of the phase space.
  - These significantly increase the CPU consumption in these regions and compounded with other filters become close to unmanageable.
  - If it is possible to have smarter generator treatment of these cuts it would help significantly.

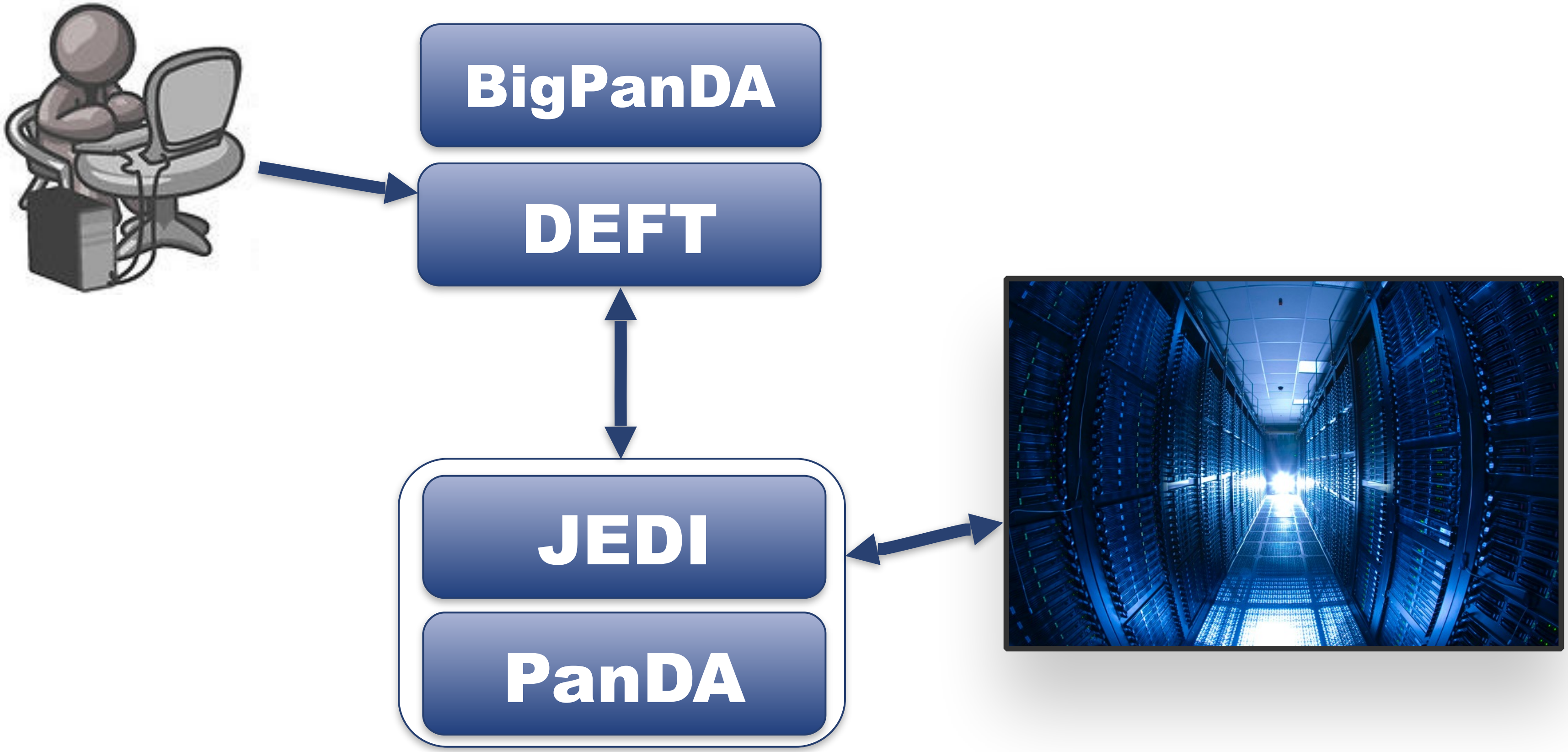
- Use of LHE weights for systematic variations
  - Now started to be used more frequently in ATLAS
  - Careful validation including several closure tests has been performed
    - Still some things to be checked for the PDF weights.
  - Not possible for all generators
    - Only for some processes in Sherpa 2.2 are available

Summary

- MC production consumes a large proportion of all ATLAS grid resources
  - Event generation is a significant fraction of that.
- Software integration is quite flexible
  - Introduction of OTF and integration grid running modes.
- Comprehensive validation procedure in place
  - Observed discrepancies are reported back to MC authors.
- CPU consumption is getting quite critical for some samples.
  - No major bottlenecks so far, but we are probably in an unsustainable situation → ~10 times the current statistics could cause some significant difficulties
  - Better use of HPC facilities could help here.

Back-ups

ProdSys2



BigPanDA

DEFT

JEDI

PanDA