Track 5 - Simulation and analysis tools

Giacomo De Pietro (giacomo.pietro@kit.edu)
on behalf of the conveners (Marilena Bandieramonte, GDP, Tobias Stockmanns, Jonas Rembser)

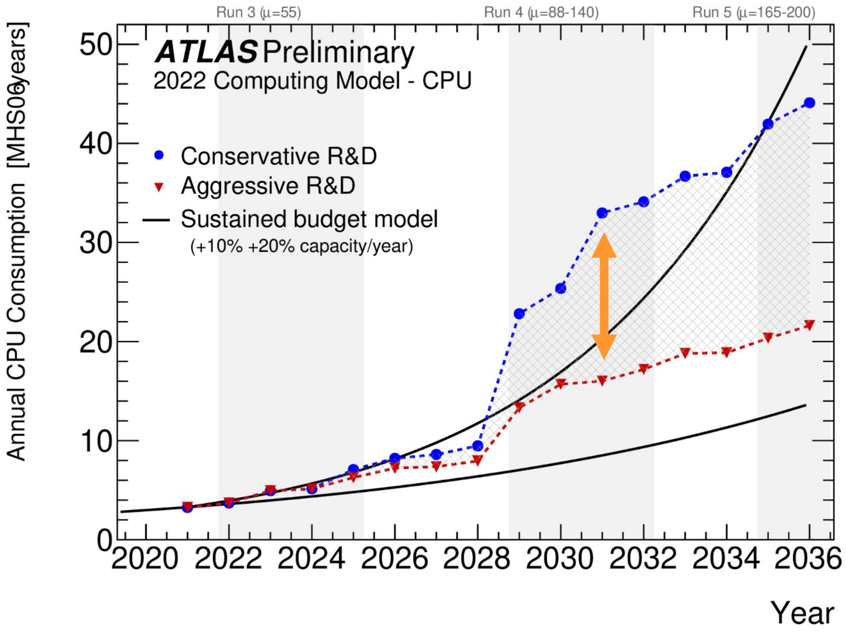
Simulation and analysis tools

- Full list of topics covered in our track:
  - object identification; object calibration; analysis workflows; software for end-user analysis; ML in analysis workflows; lattice QCD; theory calculations; MC event generation; detector simulation; fast simulation (classic and ML); quantum simulation and algorithms.
- 110+ abstracts received:
  - 59 talks (across 10 parallel sessions!)
  - 29 posters
- Impossible to cover all the discussed topics in this (very short) summary!
  - Sorry if I don’t touch your most favorite topic :(

Simulation

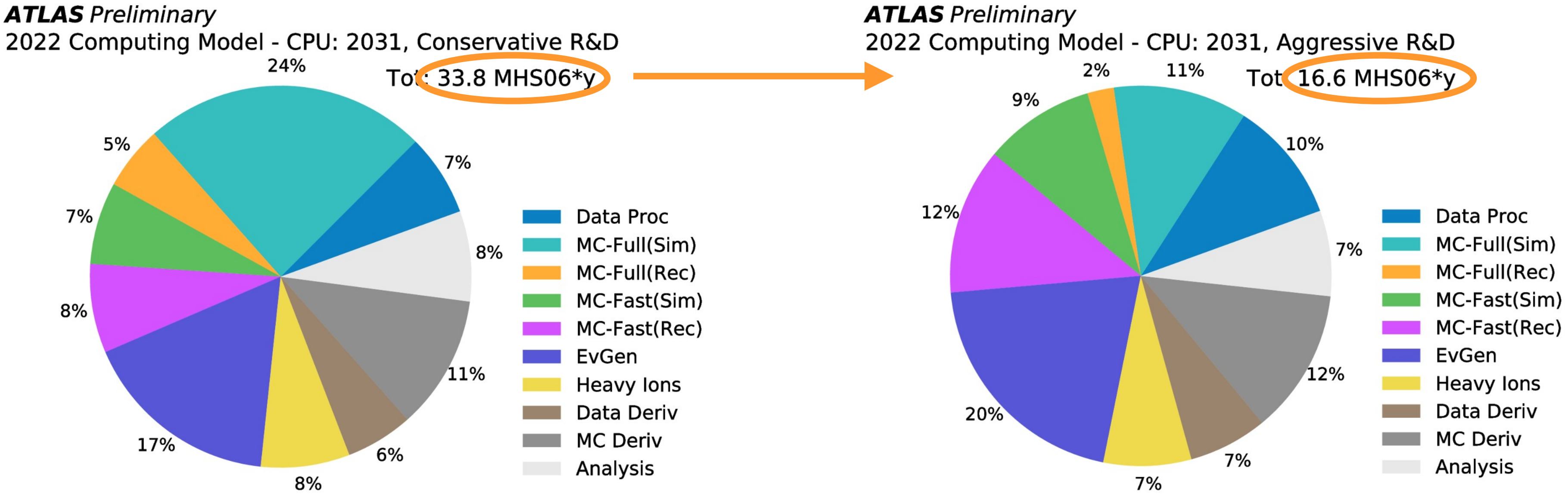
Quest to reduce the CPU consumption

- Community effort is required to reduce the CPU consumption



Quest to reduce the CPU consumption

- Community effort is required to reduce the CPU consumption



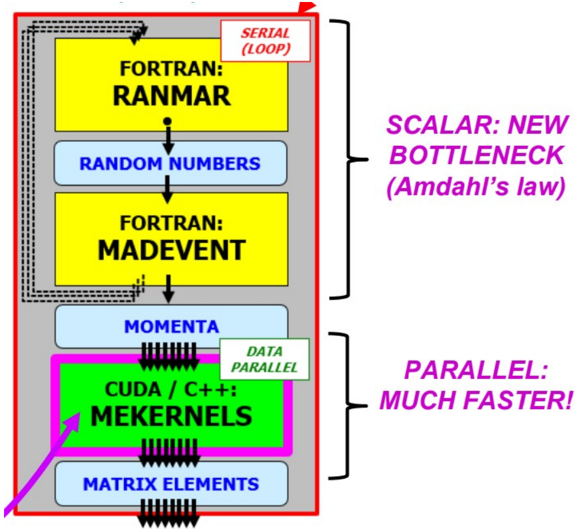
Quest to reduce the CPU consumption

- Community effort is required to reduce the CPU consumption
- And the effort is ongoing on multiple fronts:
  - Efficient event generation
  - Better usage of GPUs
  - Improved geometry descriptions
  - Code optimization
  - Machine learning
  - …

Hard constraint: physics performance must not deteriorate!

Efficient event generation

- Data parallelism for speeding-up the Matrix Element calculation
  - Beneficial also for NLO calculations (though more work is still necessary here)



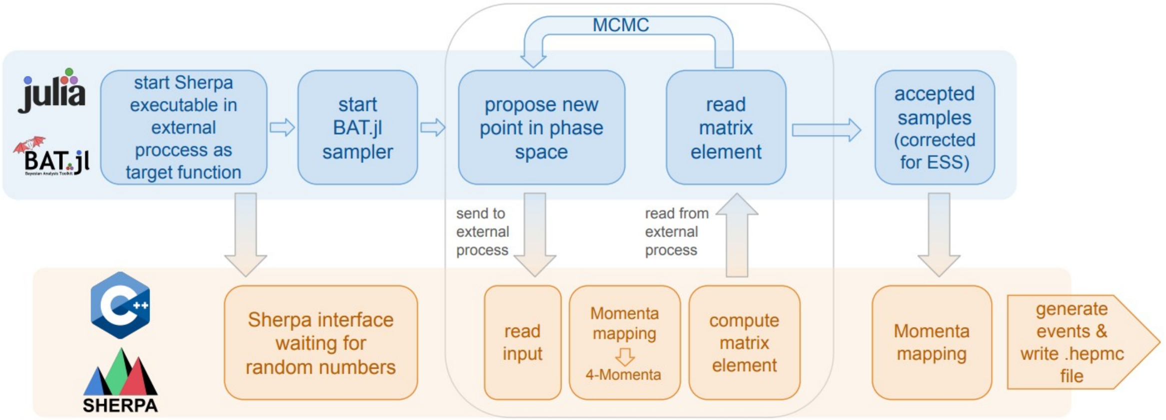
In our work on MG5AMC "CUDACPP" we have targeted data parallelism on both vector CPUs and GPUs from the very beginning!

- We speed up the matrix element (ME) calculation via data parallelism with excellent lockstep processing
  - On NVidia and AMD GPUs in the order of x100 to x1000
  - On vector CPUs we achieve the theoretical speedup limit of x8 for AVX512 SIMD in double precision
- We achieved overall speedups between x3 and x70 (for DY+3j and $gg \to t\bar{t}ggg$) depending on the physics process
  - For many processes (like DY+3j) we speed up the ME so much that the bottleneck is the non-ME component

Efficient event generation

- Data parallelism for speeding-up the Matrix Element calculation
- Improved efficiency of sampling multi-dimensional distributions

$|\mathcal{M}|^2$ is typically multi-modal, wildly fluctuating & computationally expensive



Efficient event generation

- Data parallelism for speeding-up the Matrix Element calculation
- Improved efficiency of sampling multi-dimensional distributions
- Reduction of negative weights in NLO parton shower matching

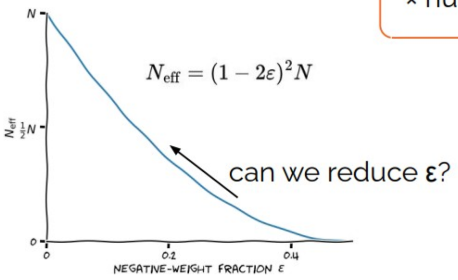
MC generation:

compute cost = cost per CPU-hour

× CPU hours per event → general (q-qb) colour singlet processes now implemented
*Sarmah, Siódmok, **JW*** [arXiv: 2409.16417]

× number of events → due to be included in Herwig 7.4.0

→ pheno studies in progress

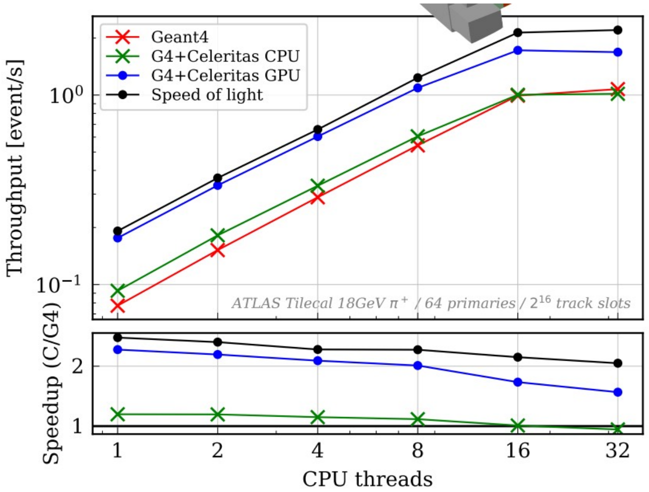


Better usage of GPUs

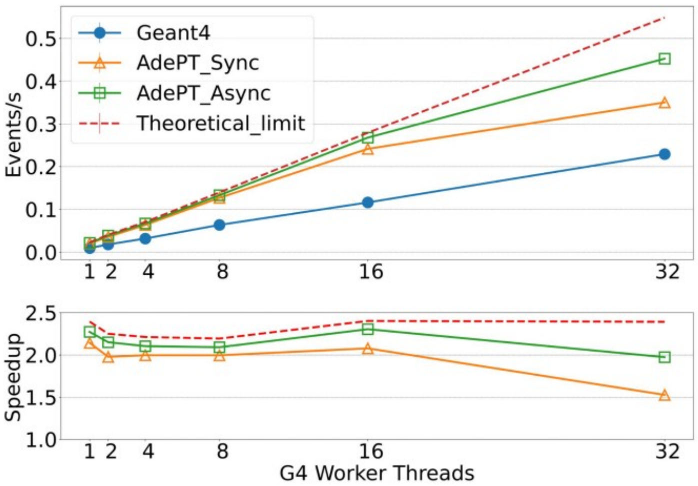
- AdePT and Celeritas R&D projects for efficient EM simulation on GPUs
  - Basic EM physics in good agreement with Geant4
  - Several bottlenecks identified and addressed
  - Ongoing experiments integration and validation

GPU: Nvidia A100
Input: 4 TTBar per thread
Geometry: CMS2018
No magnetic field

ATLAS TileCal test beam results



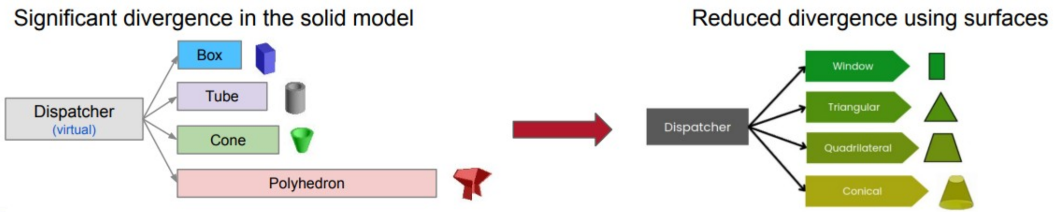
Throughput comparison of AdePT and Geant4



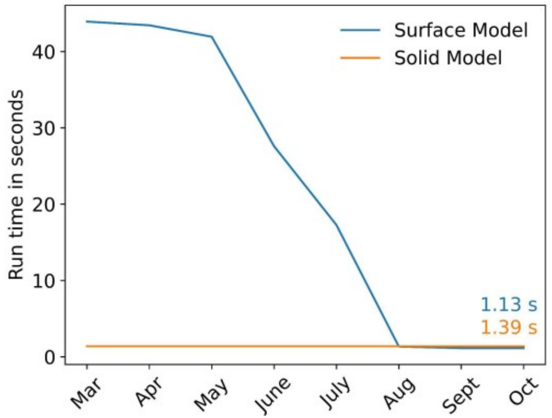
Better usage of GPUs

- AdePT and Celeritas R&D projects for efficient EM simulation on GPUs
- More GPU-friendly surface models from VecGeom

- ~~recursive calls, virtual functions,~~ less complex algorithms ➞ lower register and stack usage ➞ higher occupancy on GPU?
- reduced complexity per surface ➞ lower divergence?
- uncoalesced memory accesses ➞ high latency (intrinsic to geometry)
- State is known by navigation, no pushes required, enables potential use of mixed precision

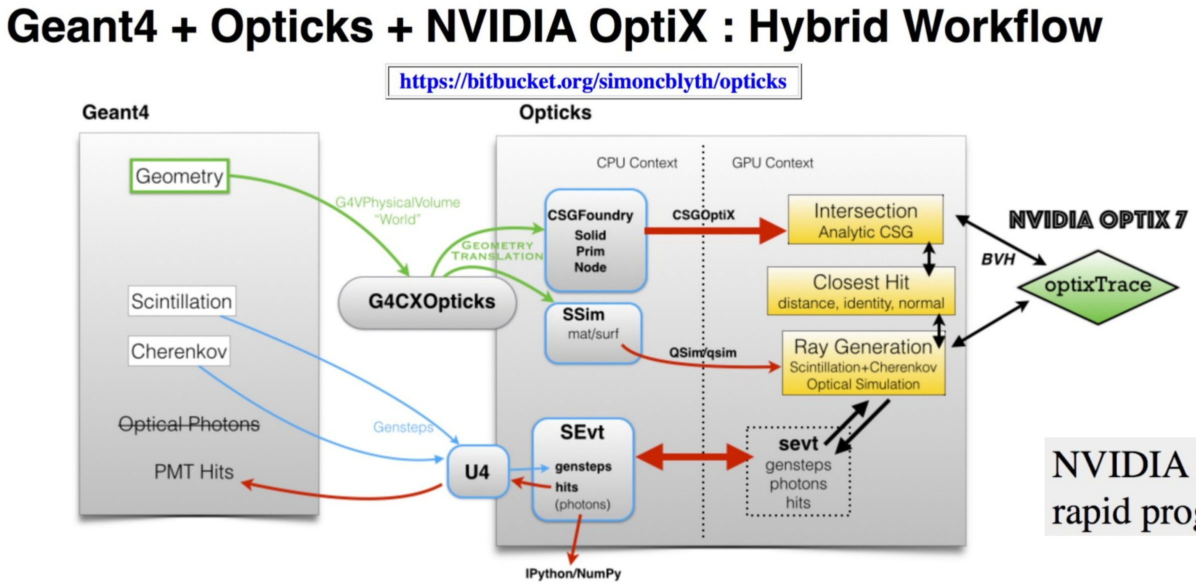


We've come a long way!
Raytracing in CMS TB
~ **40x** speed-up in 6 month



Better usage of GPUs

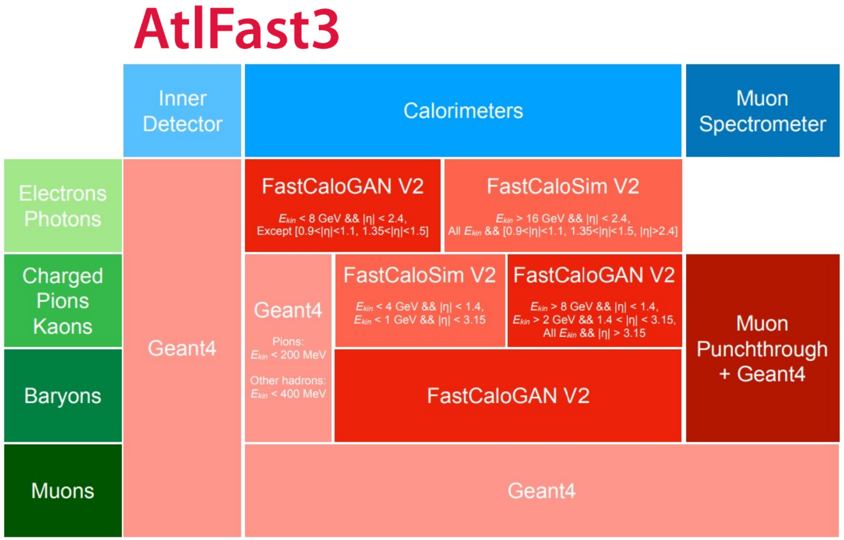
- AdePT and Celeritas R&D projects for efficient EM simulation on GPUs
- More GPU-friendly surface models from VecGeom
- Using NVIDIA (closed source) software for optical photon simulations

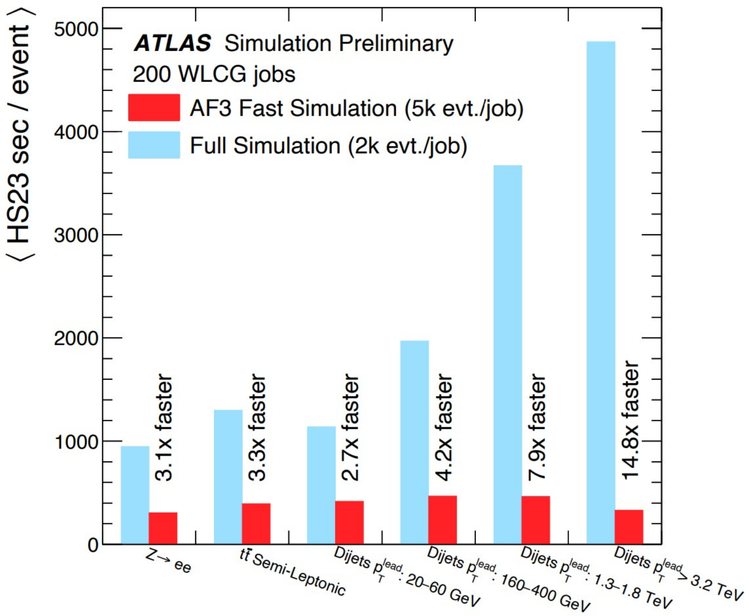


NVIDIA Ray Trace Performance continues rapid progress (2x each gen., every ~2 yrs)

Machine learning

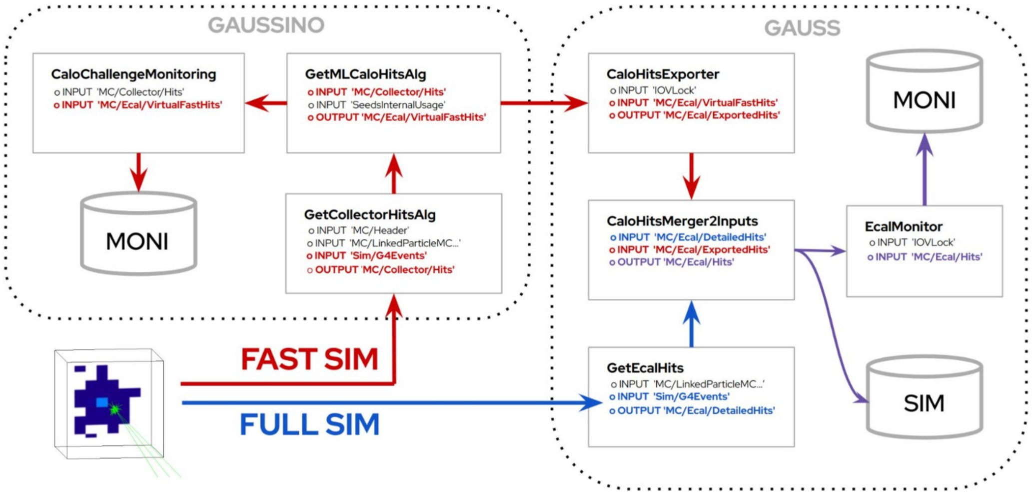
- Machine learning methods are now fully integrated into the standard simulation of many experiments
  - Hybrid usage next to full Geant4 simulation and parametrised modeling

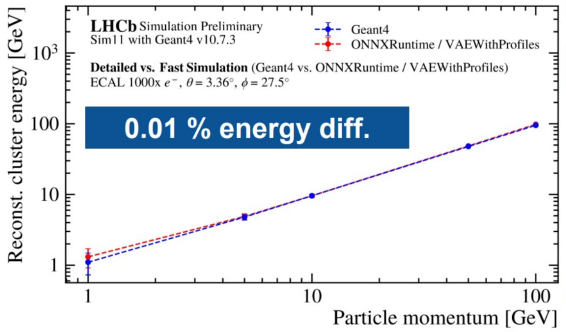


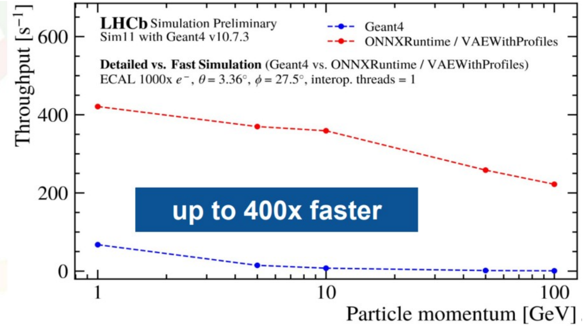


Machine learning

- Machine learning methods are now fully integrated into the standard simulation of many experiments
  - Hybrid usage next to full Geant4 simulation and parametrised modeling



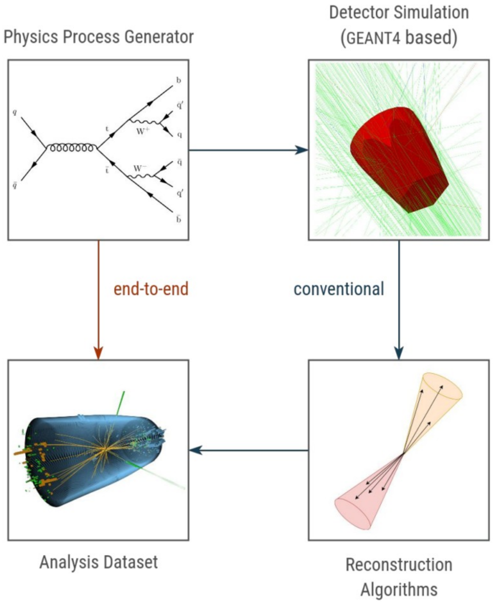


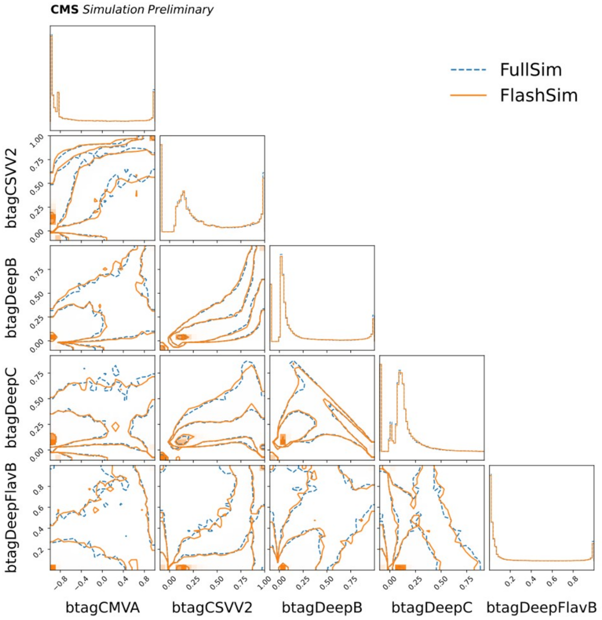


particle gun

Machine learning

- Machine learning methods are now fully integrated into the standard simulation of many experiments
- Promising results from end-to-end simulation (CMS FlashSim)





How fast do we need FlashSim to be

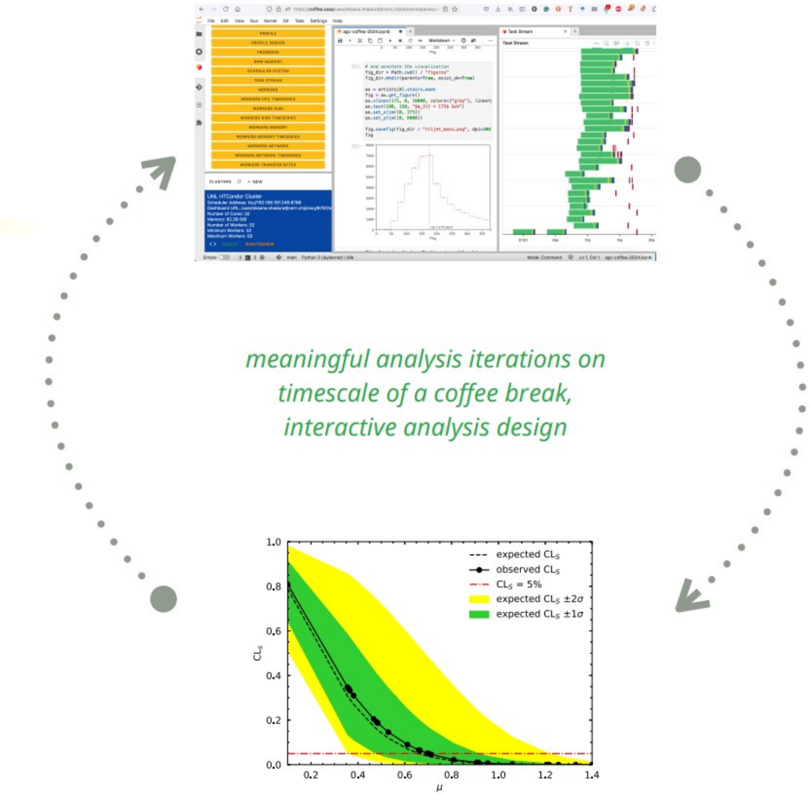
- If you already have generated samples, as fast as possible
- If the generator is very slow, we are easily in the shadow of the generator

What if we can avoid being generator-speed limited by **reusing** generated events?

- Overampling!

Analysis tools

Fast and efficient analysis



Fast and efficient analysis

- Experiment-independent declarative paradigms for analysis

- ***Toolbox*** supporting declarative approach for HEP analysis
- re-implementation from ***NAIL*** (improved modularity)
- stand-alone Python package:
  - ***DAG handling***
  - ***Sample Processing*** : event loop definition
  - ***Interface Dictionary*** : translation of input naming
  - Backend ***processors*** (for event loop):
    1. Basic ***Loop*** *processor* (C++ compiled)
    2. ***RDF****-based processor* (C++ compiled - Multi-thread support)
    3. ***Direct python*** processor

(***Available*** / ***Under development***)

Extension

Multiple-input data formats

Same flow on different-format datasets (input naming translation)

Supported data-formats:

- **NANOAOD** (CMS) ***Full***
- **PHYSLITE** (ATLAS) ***Preliminary***

Extension

Full analysis chain

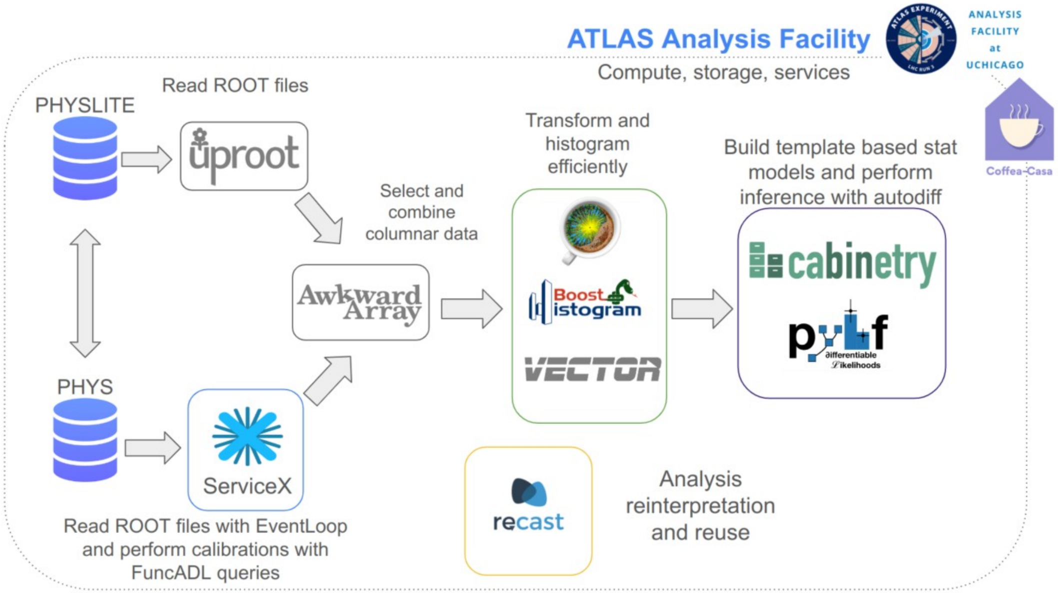
Extend the flow definition to procedure incorporating all the steps needed to extract the result of a complex analysis task

Demonstrator github repository

https://github.com/ICSC-Spoke2-repo/nail-dev

Fast and efficient analysis

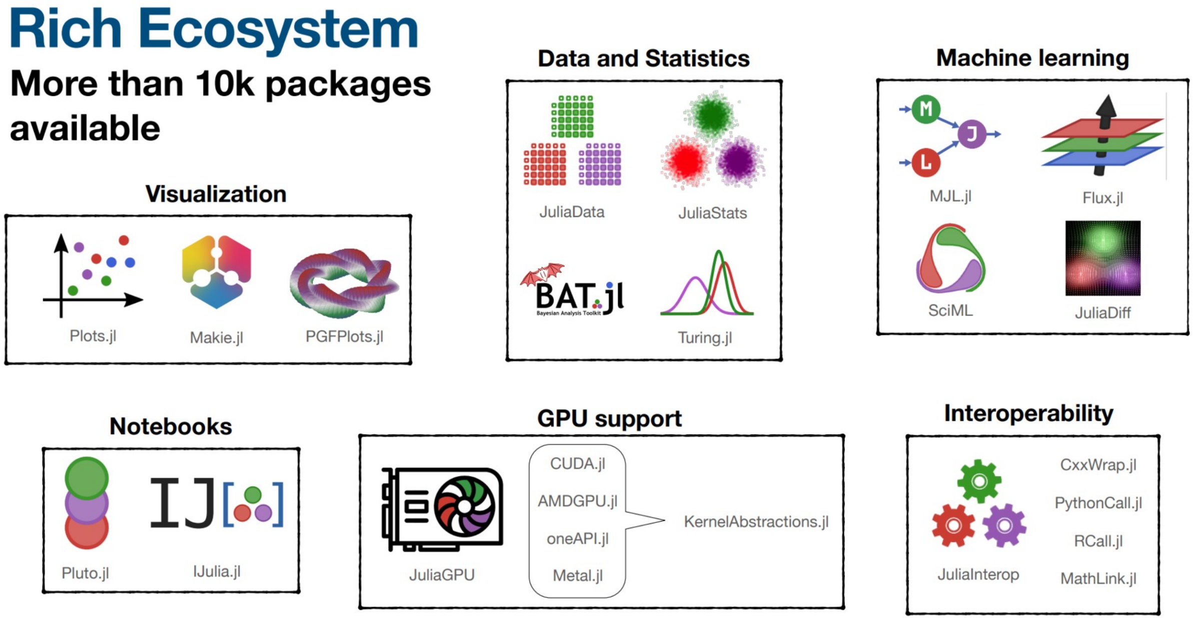
- Experiment-independent declarative paradigms for analysis
- Rich pythonic ecosystem for a columnar analysis pipeline



Components of an ATLAS Analysis Grand Challenge (AGC) demonstrator pipeline (c.f. The 200Gbps Challenge (Alexander Held, Monday plenary))

Fast and efficient analysis

- Experiment-independent declarative paradigms for analysis
- Rich pythonic ecosystem for a columnar analysis pipeline
- What about a full end-to-end analysis demonstrator in Julia?



ROOT

- 2 plenary + 4 parallel talks (in Track 5) from ROOT team

The ROOT Project at the end of Run 3 and towards HL-LHC

ROOT RNTuple & EOS
A holistic approach to data analysis

Benchmark Studies of Machine Learning Inference using SOFIE

New RooFit PyROOT interfaces for connections with Machine Learning

On-the-fly data set joins and concatenations with ROOT's RNTuple

Zero-overhead training of machine learning models with ROOT data

ROOT

- 2 plenary + 4 parallel talks (in Track 5) from ROOT team

ROOT has to evolve to meet the challenges of future scientific computation

- Possible only by **dropping some of its older components or changing some behaviour**
  - E.g. is automatic memory management still needed?
- **Not a revolution, but a piecewise renovation**, leading to a completely new system
  - Not new: code using ROOT today has little to do with the one written in early Run 2
- New components are being introduced and adopted *today*
  - `RNTuple`, `RDataFrame`, new *Pythonic Interface*, completely new `RooFit`, …
- With the early adoption of new components, **the jump to ROOT 7 will not be large**
  - **Some changes will be backwards incompatible**, but a much smaller jump than ROOT5 →ROO6
  - Migration or transition paths will be documented and clearly communicated
- **ROOT 7.00.00 will be released during LS3**, well on time for Run 4 MC productions

ROOT

- 2 plenary + 4 parallel talks (in Track 5) from ROOT team

ROOT has to evolve to meet the challenges of future scientific computation

- Possible only by **dropping some of its older components or changing some behaviour**
  - E.g. is automatic memory management still needed?
- **Not a revolution, but a piecewise renovation**, leading to a completely new system
  - Not new: code using ROOT today has little to do with the one written in early Run 2
- New components are being introduced and adopted *today*
  - `RNTuple`, `RDataFrame`, new *Pythonic Interface*, completely new `RooFit`, ...
- With the early adoption of new components, **the jump to ROOT 7 will not be large**
  - **Some changes will be backwards incompatible**, but a much smaller jump than ROOT5 →ROO6
  - Migration or transition paths will be documented and clearly communicated
- **ROOT 7.00.00 will be released during LS3**, well on time for Run 4 MC productions

Clear direction: reinforce ROOT's presence in the Python ecosystem, prioritising the Python users experience

Thank you to all the speakers of Track 5!