Announcement: State of Software Survey

The EICUG Software Working Group (SWG) is repeating its “State of Software” survey from 2021 [1, 2].

In its updated survey, the SWG is collecting information on the community’s specific software tools and practices during the call for detector collaboration proposals. Everyone who has been involved in the simulation efforts for the proposals or the related physics and detector studies is invited to participate in the survey:

https://www.surveymonkey.com/r/EICUG-SWG-StateOfSoftware-2022

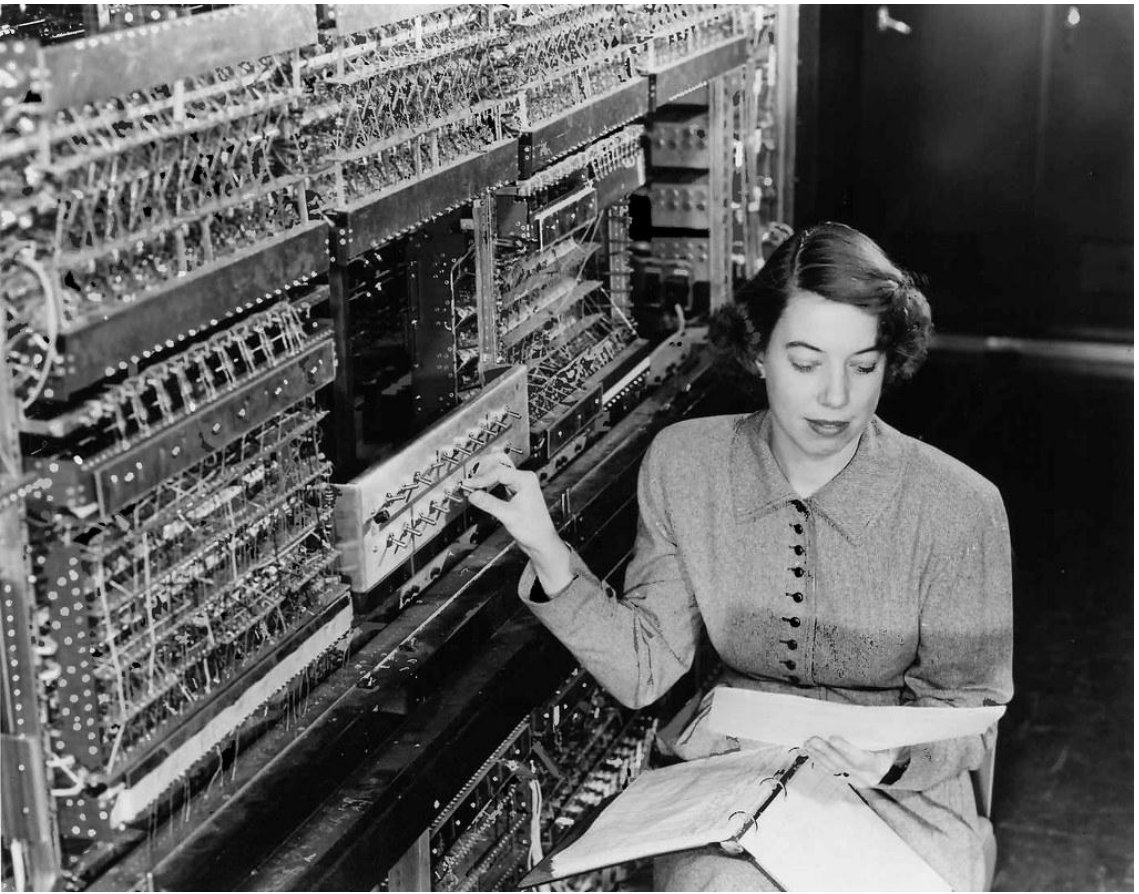
The “State of Software” survey should take at most five minutes to complete and will close on February 13.

Thank you very much for your support,

Wouter Deconinck, Markus Diefenthaler, Sylvester Joosten, and Kolja Kauder on behalf of the SWG.

[1] https://eic.github.io/activities/ucd.html

[2] https://indico.jlab.org/event/420/#16-eic-user-survey-and-focus-g



ATHENA

Software & Computing

Lessons and next steps

Wednesday 2022-02-03

The Software and Computing WG Conveners:
Andrea Bressan (University of Trieste and INFN) ,
Dmitry Romanov (Jefferson lab) ,
Sylvester Joosten (Argonne National Laboratory) ,
Whitney Armstrong (Argonne National Laboratory) ,
Wouter Deconinck (The University of Manitoba)

Philosophy*:* Let's prepare for our future at the EIC!

ATHENA

- **Build forward-looking team of developers to ensure the long-term success of the EIC scientific program in software & computing.**
- Focus on modern scientific computing practices
  - Strong emphasis on modular orthogonal tools.
  - Integration with HTC/HPC, CI workflows, and enable use of data-science toolkits.
- Avoid "not-invented-here" syndrome, and instead leverage cutting-edge CERN-supported software components where possible.
  - Build on top of mature, well-supported, and actively developed software stack.
  - Externalize support burden where possible.
- Actively work with the EICUG SWG to help develop and integrate community tools for all collaborations.

ATHENA Software Stack

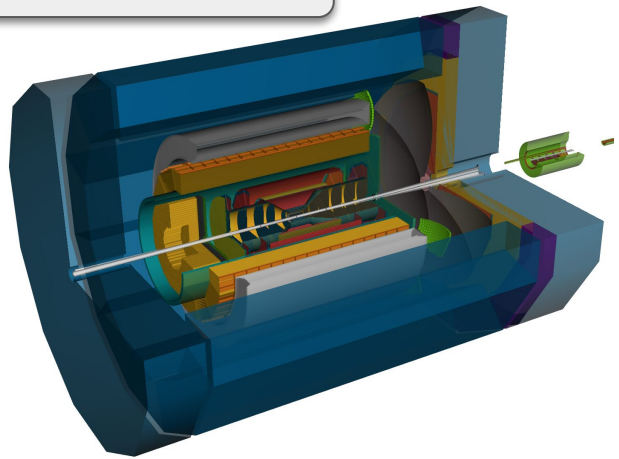
- Detailed detector geometry description in **DD4HEP**, which steers the Geant4 simulations
- Reconstruction framework (**JUGGLER**) built on top of **GAUDI**, leveraging **ACTS** for tracking and **Tensorflow** for AI.
- Modular components communicate through a robust, flat data model (**EICD**, implemented using **PODIO**).
- Leverage dedicated GitLab server (**eicweb**) with CI backend for reproducible container builds (using **Spack**), and automated tests and benchmarks.

MCEG

DD4Hep,
Geometry,
Geant4 (DDSim)

Juggler,
Gaudi - processing
ACTS - tracking,
Tensorflow - ML
Custom algorithms

Analysis/Benchmarks



Highlights

- We took some risk starting from scratch, but it paid off!
- Performant modern simulation/reconstruction toolkit, fully operational in 4 months!
- Setup the heart of a powerful toolkit aimed far beyond the detector proposal.

DD4hep as single source of geometry

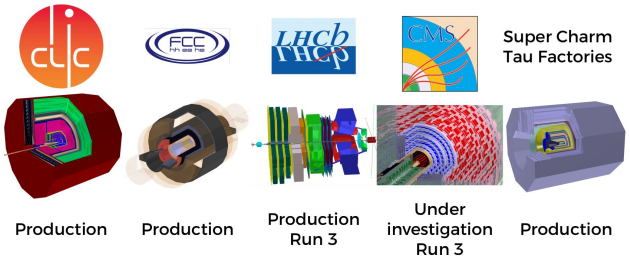
Parametrized geometries

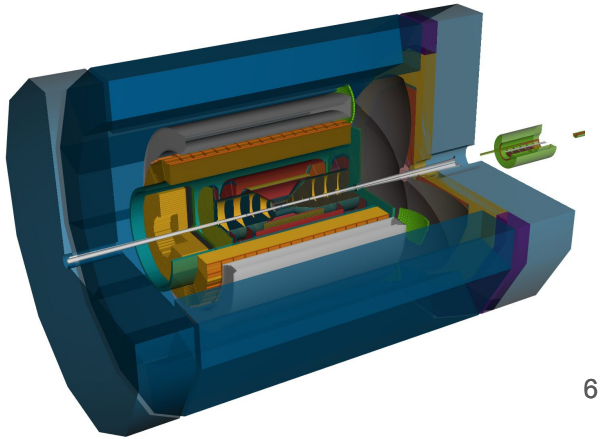
- Geometry configuration happens through XML files: easy to edit for beginners!
- Learning curve to design a proper detector parametrization a bit steeper, but easy to overcome under expert guidance.
- Worth it:
  - Well designed parametrization easy to maintain.
  - Parametrized geometries show their power when designing and optimizing a detector: were able to manage four major design iterations for ATHENA (*Acadia, BigBend, Canyonlands, DeathValley*), and many smaller optimizations.
- jsROOT geometry browser invaluable for both beginners and experts.

Development with DD4hep

- Very positive interactions with DD4hep developers:
  - Joined some of their regular meetings
  - ATHENA had multiple pull requests merged into main DD4hep GitHub repository, very fast turnaround!
- Long-term plan for custom DD4hep plugins (NPDet): minimize amount of custom code we need to maintain ourselves: either push upstream (to DD4hep) or downstream (to specific detector implementation).
  - ***DD4hep developers happy to include ATHENA plugins (eg. for Cherenkov detectors) in official DD4hep***

The DD4hep Community

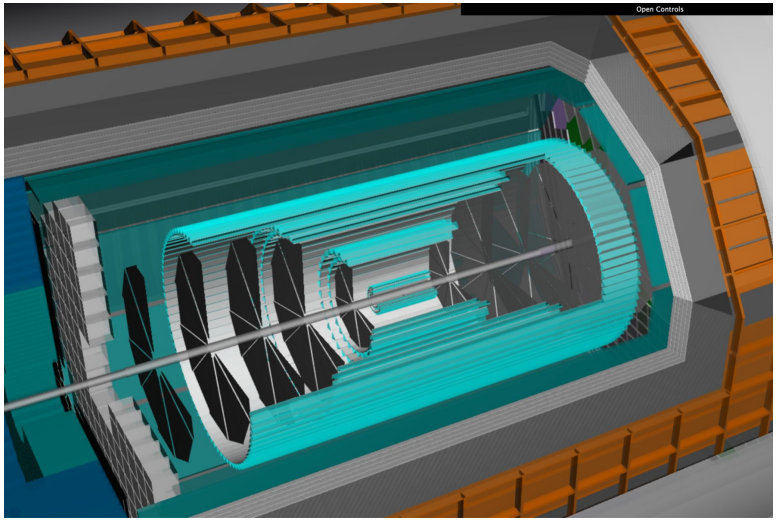




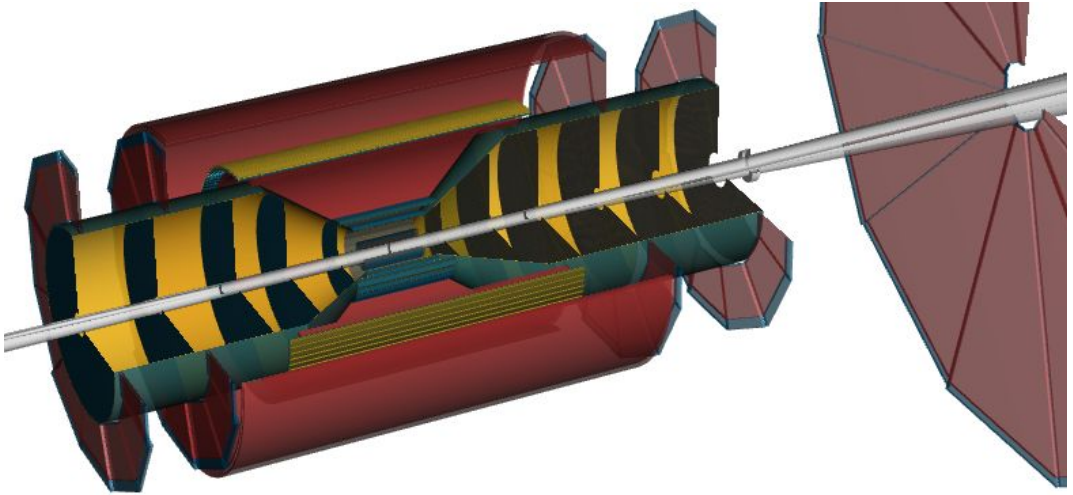
Detailed geometry implementation

EXAMPLE: Tracking Systems

Acadia

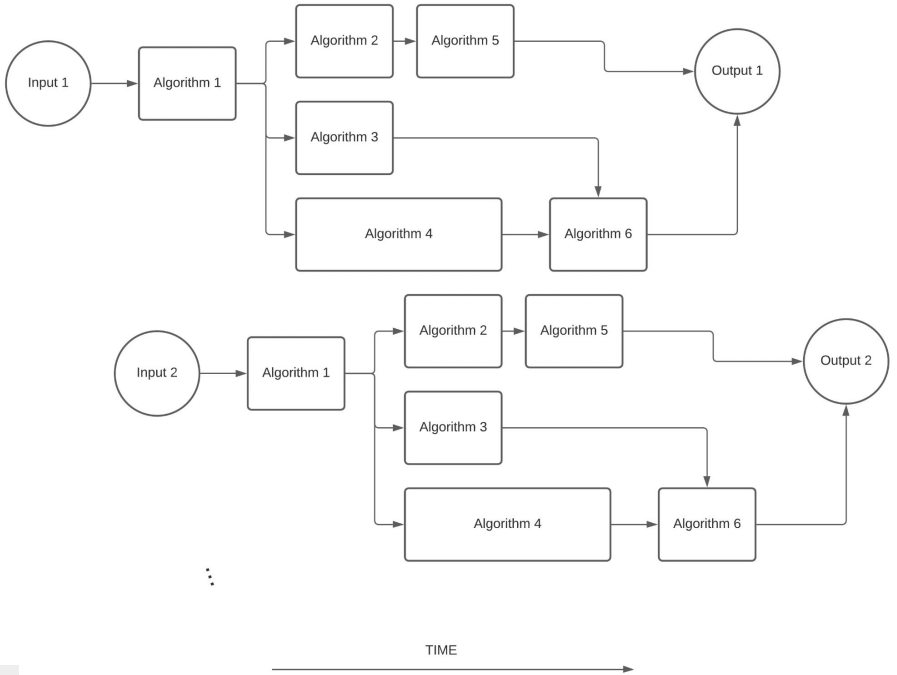


DeathValley



Gaudi enables highly flexible concurrent workflows

- All Juggler algorithms reentrant by design
  - Support concurrent processing, heterogeneous environments from the beginning
  - Highly modular
  - Easy to integrate with external toolkits (ACTS, tensorflow, …)
  - ***Reentrant algorithms easy to write, debug, validate, and compose, even by beginners!***
- Reconstruction: steered through a simple python script → trivial to reconfigure reconstruction for different detector layouts, or for subsystem-only reconstructions
  - Algorithms designed to be as small as possible to allow easy composition/substitution/optimization
  - Even true for the event scheduler! Concurrency in Gaudi enabled by swapping out the event scheduler (can use concurrent, parallel, single-threaded, …).



We are well prepared to run heterogeneous environments (using eg. GPUs, TPUs, …) in our reconstruction chain!

Tracking with ACTS at the design stage

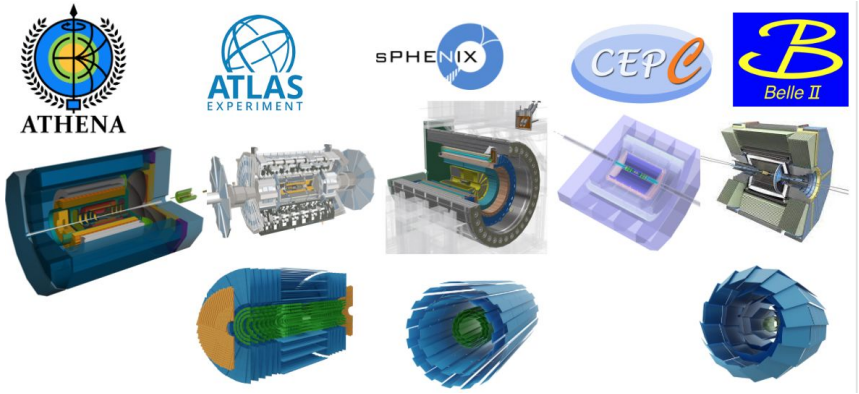
Very early on we had issues with tracking due to an ACTS version change (upgrade to v8). This was ultimately caused due to our lack of communication with the ACTS developers. Since then, we instituted (bi)weekly meetings with the ACTS team. With this in place, we went from v8 all the way to v16 without any major issues!

Lesson: communication (and even collaboration) with upstream developer team invaluable. We waste time and resources when working in isolation!

Highly positive experience: the ACTS team treated us as first-rate "clients" of their project

- Regular meetings with the ACTS team invaluable
- ATHENA commits to upstream ACTS repository now part of the main codebase (less for us to maintain!)
- Close collaboration was crucial, allowed us to stay close to the latest releases in rapidly evolving codebase.

The ACTS Community



First collaboration to use ACTS throughout the entire prototyping stage

- We were running ACTS for all our productions!
- Important feedback to ACTS developers on automation (eg. material maps).
- ***EIC-unique problems will solved together with the ACTS team (eg. dealing with off-axis geometries for the B0 tracker).***

EIC Data Model (eicd)

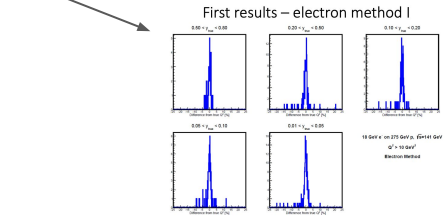
- Well-defined **flat data model** defined in a single yaml file. No external libraries needed to load data. No hard lock-in to any file format (eg. ROOT).
- Enables collaborators to see the big picture over implementation details.
- Encouraged generalization and consistency (eg. weight, likelihood, probability in IRT)
- ***Flat data model enabled collaborators to use data in unforeseen ways (good!):***
  - python-only analysis (no ROOT)
  - python event display
  - VR event display
  - offline SIDIS fast/full analysis framework could integrate easily with full reconstruction
  - …
- Similarities with EDM4hep (part of key4hep)
  - ***First meeting with key4hep last Friday: move towards: Lot of enthusiasm from HEP community to collaborate! Consider adopting EDM4hep in-line with our software philosophy.***

Showcase: *Enabled collaborators to think about the big picture (versus implementation details)*

1. Define data model

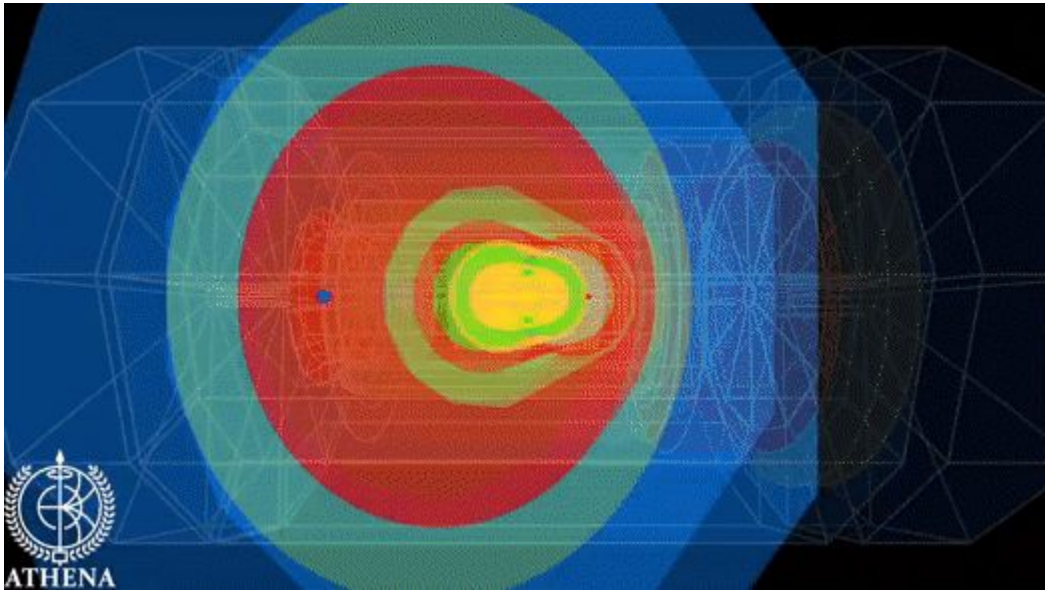
```
527   ## ============================================================================
528   ## Kinematic reconstruction
529   ## ============================================================================
530
531   eic::InclusiveKinematics:
532     Description: "Kinematic variables for DIS events"
533     Author: "S. Joosten, W. Deconinck"
534     Members:
535       - float             x               // Bjorken x (Q2/2P.q)
536       - float             Q2              // Four-momentum transfer squared [GeV^2]
537       - float             W               // Invariant mass of final state [GeV]
538       - float             y               // Inelasticity (P.q/P.k)
539       - float             nu              // Energy transfer P.q/M [GeV]
540       ## Spin state?
541       ## phi_S?
542       - eic::Index        scatID          // Associated scattered electron (if identified)
```

2. Write appropriate algorithm(s) and benchmarks.
3. Results!



Showcase: Modern toolkit enables creativity

ATHENA 3D VR display created by PhD student Sean Preins (UCR)



Structuring the toolkit as a set of modular, orthogonal tools (independent geometry description, independent detector simulation, independent data model, …) enabled our collaborators to make use of our environment in creative new ways … a big win in our book!

Automated Workflows at eicweb

GitLab server (eicweb.phy.anl.gov)

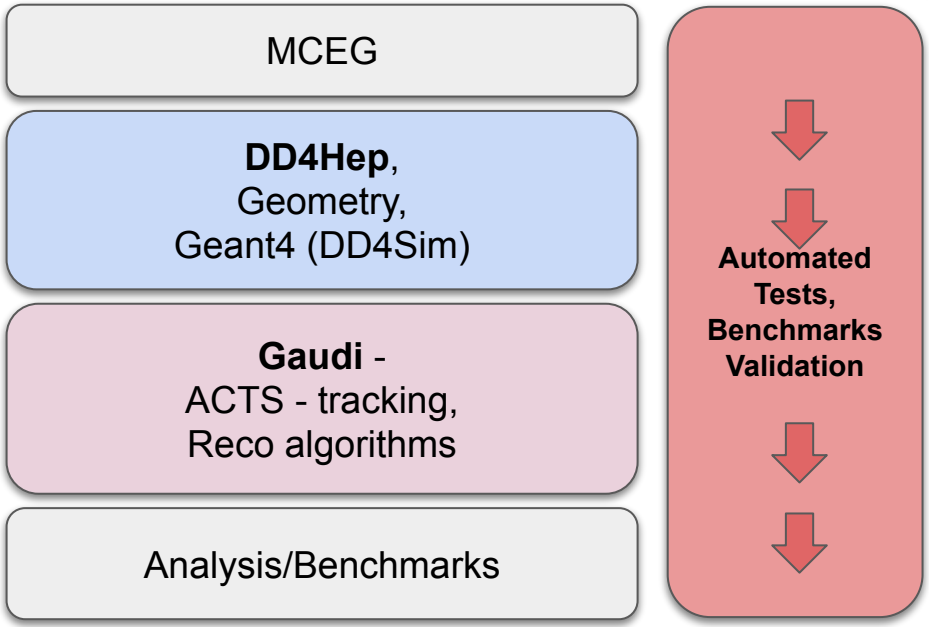
- mirrored on github.com/eic
- continuous integration
- dedicated build cluster

Runs automatically on each user commit, executing workflows running multiple tests, benchmarks and analysis

Automated containers

Both Docker and Singularity images are created nightly or on demand (commit) providing:

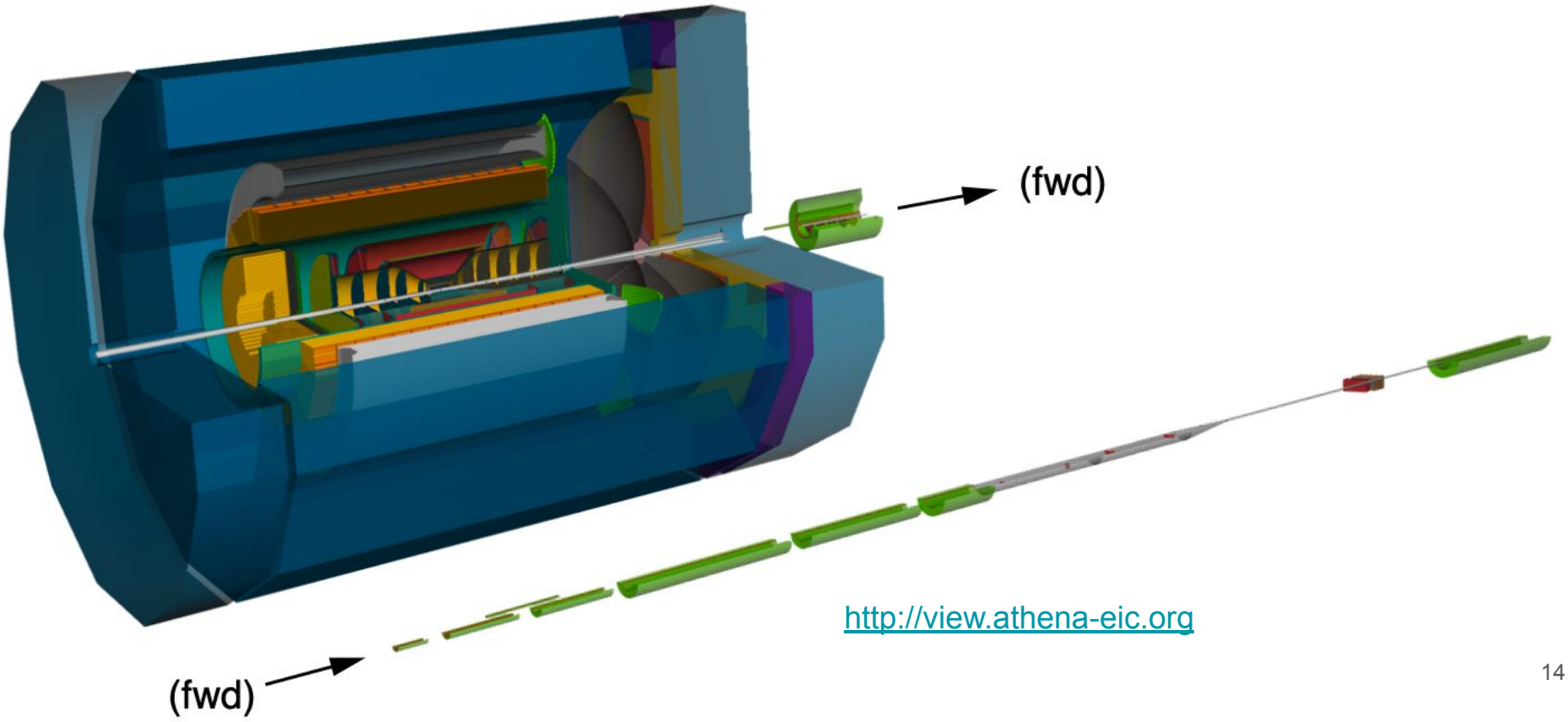
- reproducibility,
- production level images
- latest updates for those working locally



Why use a custom GitLab server?

- CI system loses benefits with shared, queued jobs
- Our collaboration controls users, yet access to HPC

Automatic Visualization



http://view.athena-eic.org

Still easy to get started locally… in only 1 line!

Step 1: `curl -L get.athena-eic.org | bash`
Step 2: `???`
Step 3: `Profit`

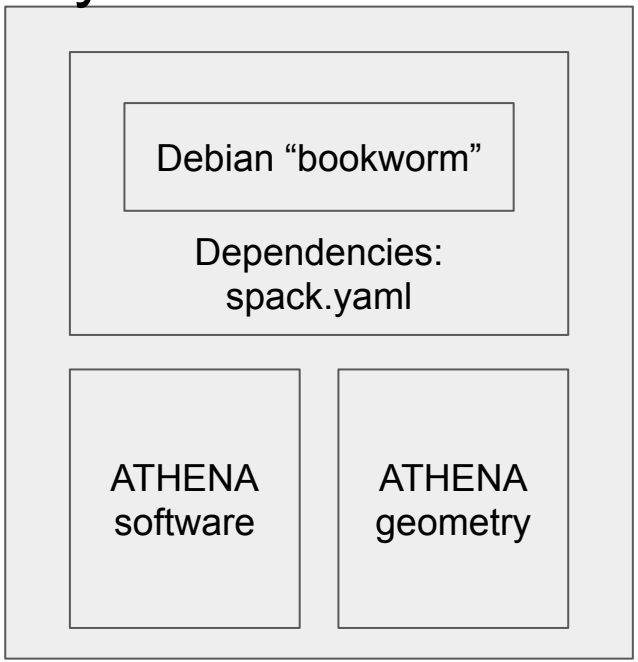
- Uses images on `/cvmfs` when available, downloads singularity sifs otherwise.
- Rolling out seamless container updates to end users
- At the same time basis of scalable computing on OSG: same containers are used everywhere.
- Note: In principle not even needed to look at data (flat format!)

Approach worked robustly during the entire proposal period. Biggest challenge was making people believe it was really that simple!

User support is paramount!

- Initial **tutorials** were crucial to get the collaboration started with a new toolkit (and help develop it further): https://eic.phy.anl.gov/tutorials/eic_tutorial/
- **Regular office hours** (3x/week) were extremely productive to support the collaboration.
- Office hours also provided valuable contact time when everyone was working from home, much appreciated by many involved.

Deployment With Containers: Layout



ATHENA software layer (updates: ~days):

- git clone of key repositories, cache busting with commit hashes through GitLab API
- all software also in EICUG SWG eic-spack repository, aim to use spack environment

ATHENA geometry layer (updates: ~days):

- multiple geometries inside each container
  - streamlined distribution of large containers when only geometries are different
- calibration and configuration artifacts defined by url, stored in container cache by hash (FileLoader plugin)
  - url is only pulled when not found in cache, i.e. only when modifying geometry
  - upstreaming to dd4hep planned

Ensure consistent detector setups for production runs.

Able to deploy manicured modern environment many very different computer systems (including OSG!)

Anatomy Of ATHENA Jobs

This worked really well to support very different computer systems!

Input:

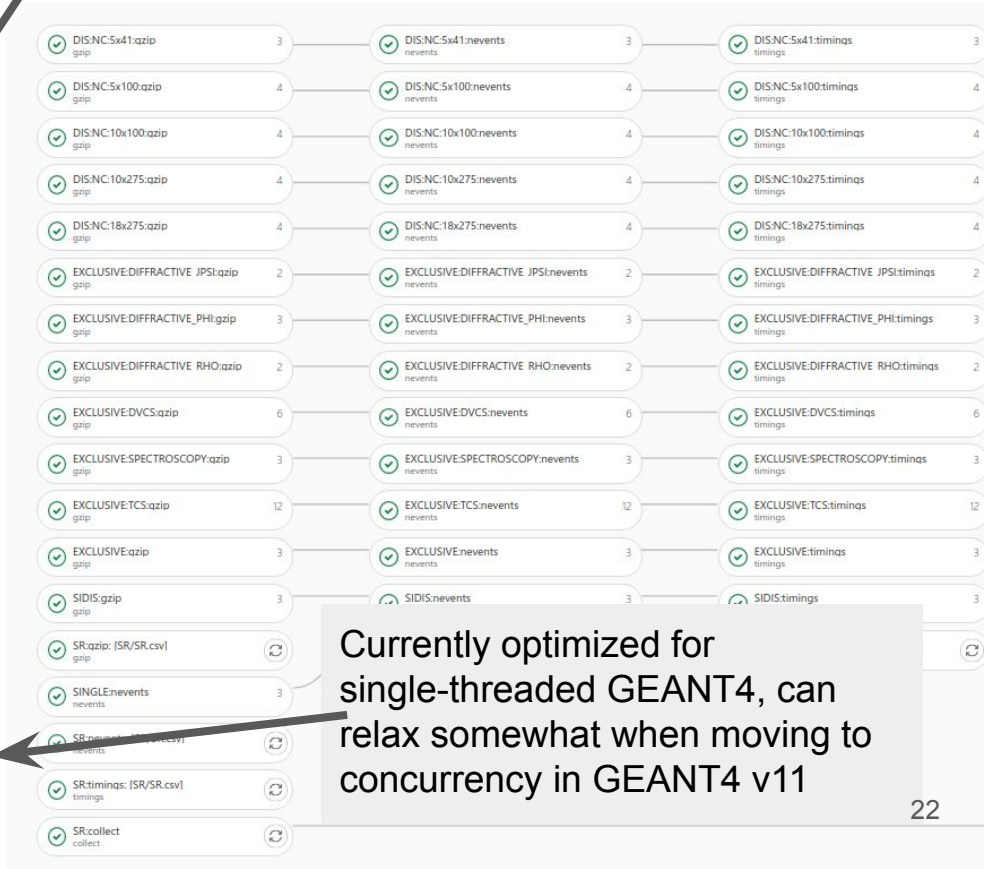
- S3: `mc cp` of HepMC v3 files (gzipped)
- condor: transfer DD4hep gun steer file

Benchmarking on eicweb as part of CI:

- running test, smoke tests, sanity checks, time-per-event determination into csv
- target time: slurm: ~20 hrs, condor: ~2 hrs

Job submission:

- identical syntax for slurm and condor
  - Automatic retrieval of csv artifacts, automatic job strategy determination
  - No user code is needed: all submission support is available on CVMFS
- memory request: 2 GB, typical use 1.5 GB



Currently optimized for single-threaded GEANT4, can relax somewhat when moving to concurrency in GEANT4 v11

Lessons Learned Running Large Productions

Slurm at Compute Canada and JLab

What worked well:

- predictability: equal specs on all nodes
- efficiencies of ~infinite network storage

What could be better

- lack of S3 access on compute nodes require pre-staging inputs and post-treatment of outputs: mitigated with cronjobs
- improvements in data transfers to their final destination, e.g. S3 or writable xrootd
- network access segmentation between DTN, interactive nodes: mitigated with cronjobs

HT-Condor on OSG

What worked well

- scalability to ~30k simultaneous jobs
- collaboration with OSG support staff
- delayed instantiation for job rate limiting

What could be better

- jobs getting vacuumed on certain clusters: mitigate with `Requirements` clauses excluding sites
- large variation (factor 10) in job durations due to spec variations: not mitigated, just resubmit jobs that time out at 19 hrs

Wishlist For Next Months

Maintenance tasks:

- Upgrades of underlying dependencies: moved to spack 0.17.1, and now using ACTS 16.0.0 (January 13, 2022) in production
- Find current limits on OSG job throughputs (known limits were all mitigated, i.e. S3 access)

Exploration tasks:

- **GPU acceleration** for ACTS as a proof of concept for enabling this more widely
- **Switch to task-based scheduler** in gaudi for default productions
- Leverage full software pipeline for prototype testing (integration with streaming DAQ).
- Further explore user-centered design together with the EICUG SWG
- Explore points of overlap and **collaboration with the key4hep project** (which independently converged on a similar toolkit).
    - ***First meeting last Friday was very positive!***

Takaway points

Some takeaway points from our experience

- Absolutely possible to migrate a userbase to a new software environment while retaining (or improving!) productivity.
- A modern containerized approach can work well in highly distributed production environments. We ran the ATHENA productions on 7 distinct HTC and HPC sites.
- Actively working with the teams behind community software projects is *highly* productive.
- Working closely with the users (regular contact time between users/developers) is paramount.
- Modern productivity tools (CI, issue trackers, development boards, slack, …) absolutely work!
- Many users are happy to be trained in modern approaches.
- Keeping the toolkit modular and orthogonal allows for direct involvement of domain experts, and enables unforeseen creativity.

Overarching philosophy:

- Focus on modern scientific computing practices
- Use what is already available, do not reinvent the wheel
- Actively work with outside organizations

Questions?