

ATHENA

EIC SWG Meeting Lessons Learned

Wednesday 2022-01-26

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!

- **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.

Highlights

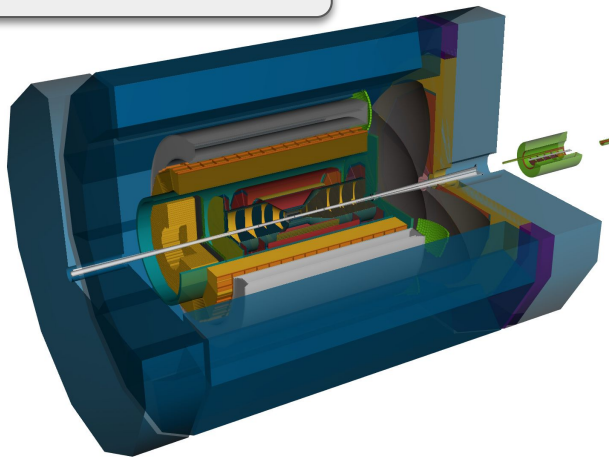
- We took some risk starting from scratch, but it paid off!
- Performant modern simulation/reconstruction toolkit, fully operational in 4 months!
- Setup the heart of a powerful toolkit aimed far beyond the detector proposal.

MCEG

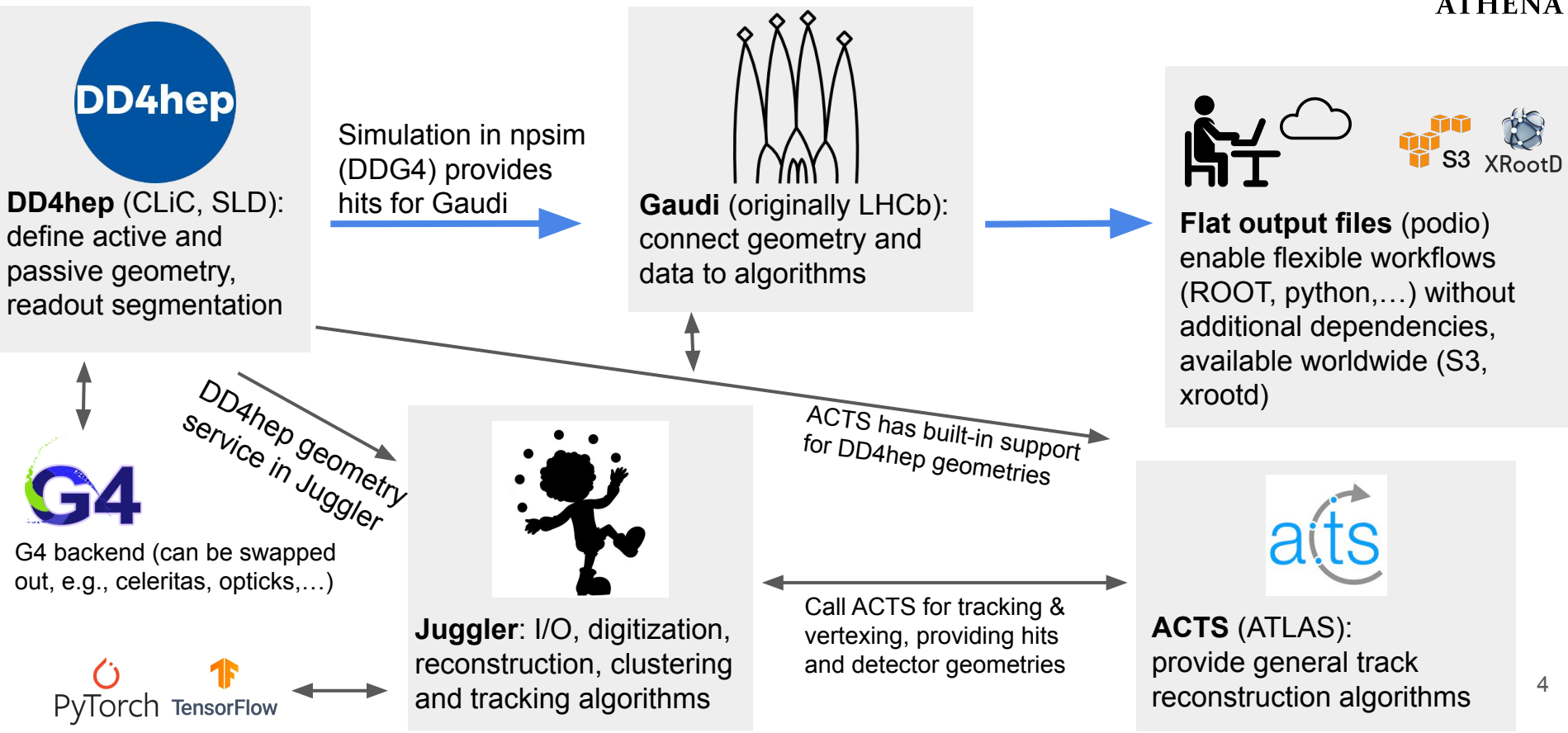
DD4Hep,
Geometry,
Geant4 (DDSim)

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

Analysis/Benchmarks



ATHENA Software Ecosystem: Emphasis On Modularity



DD4hep as single source of geometry



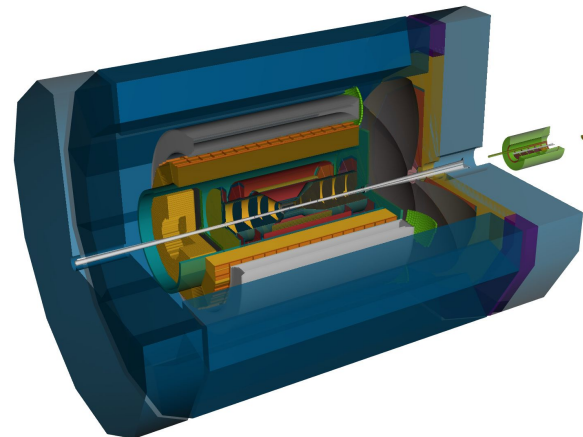
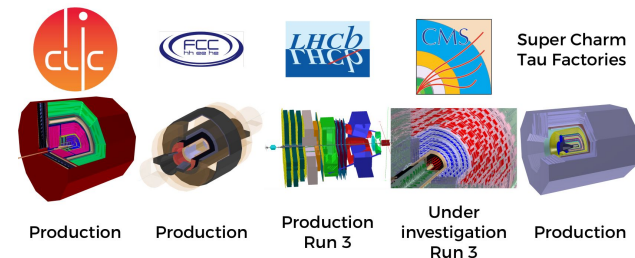
Parametrized geometries

- 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).

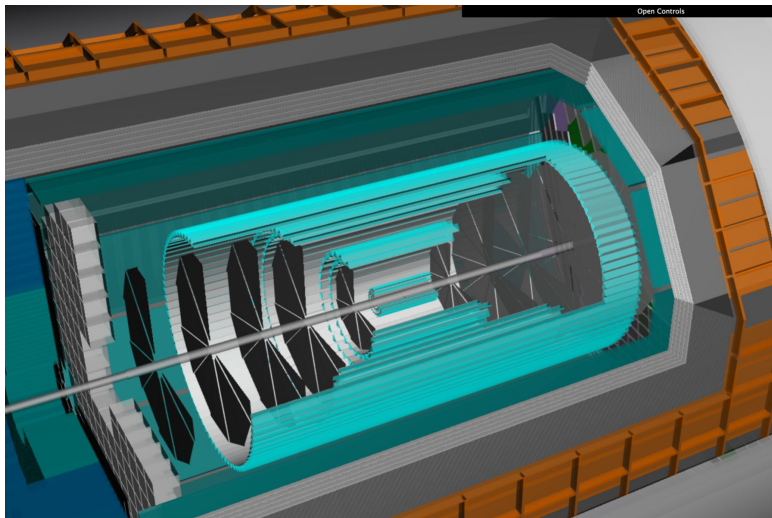
The DD4hep Community



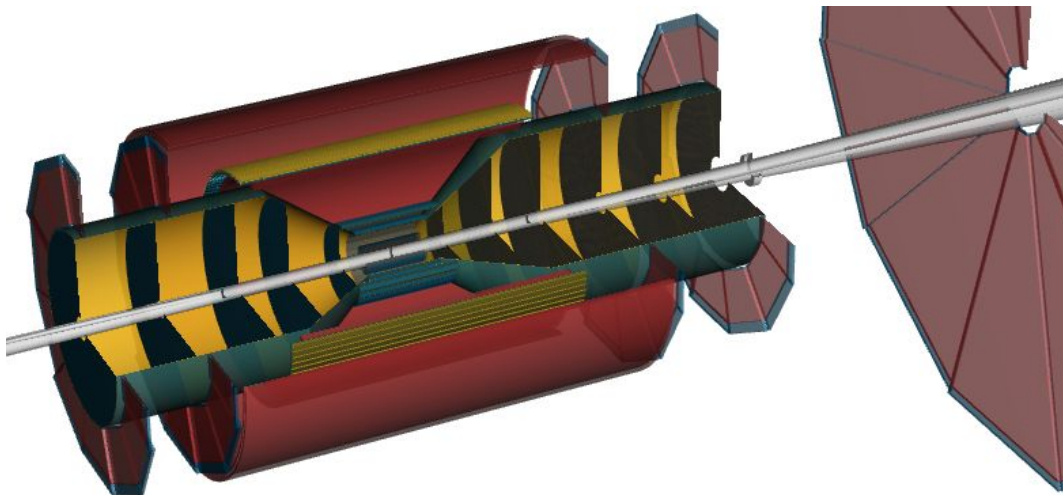
Detailed geometry implementation

EXAMPLE: Tracking Systems

Acadia

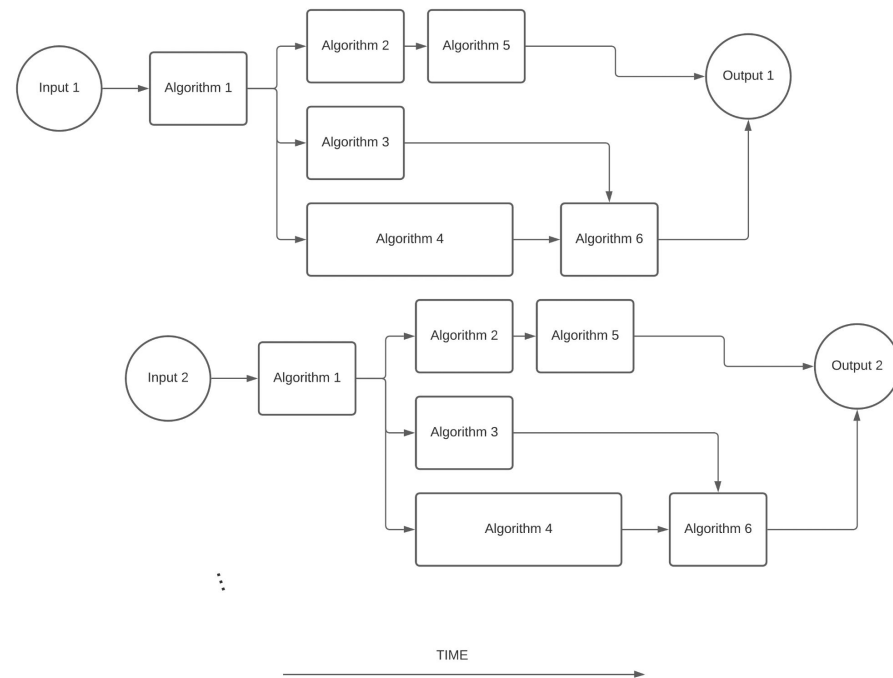


DeathValley



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, ...).

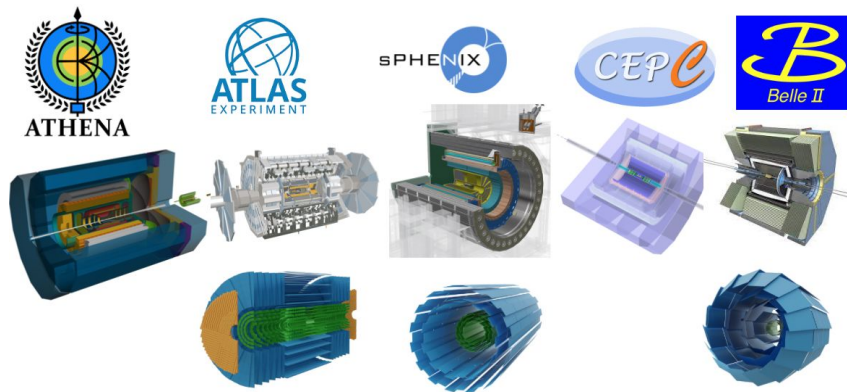


Tracking with ACTS at the design stage

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 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)
- PODness of 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), with some different design choices (relational versus pure POD), will follow up with key4hep team to discuss interoperability and/or collaboration.

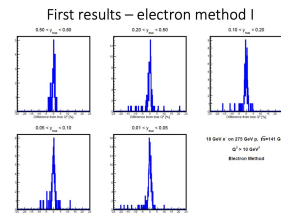
Showcase: Enabled collaborators to think about the big picture (versus implementation details)

1. Define data model

```

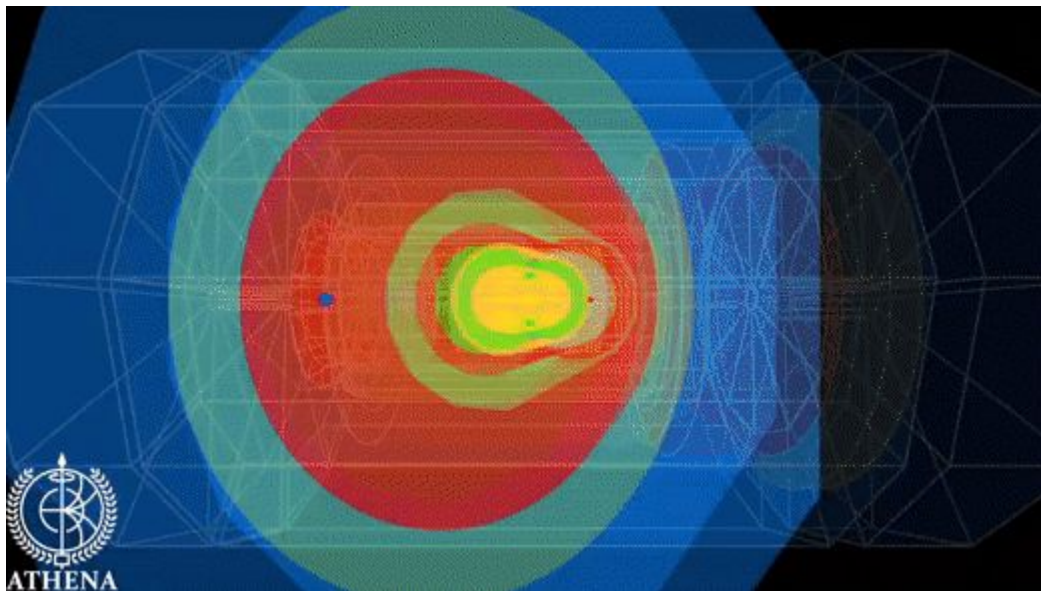
527  ## =====
528  ## Kinematic reconstruction
529  ## =====
530
531  eic::InclusiveKinematics:
532    Description: "Kinematic variables for DIS events"
533    Author: "S. Joosten, W. Deconinck"
534    Members:
535      - float      x           // Bjorken x (Q2/2P.q)
536      - float      Q2          // 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)
  
```

- Write appropriate algorithm(s) and benchmarks.
- 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)

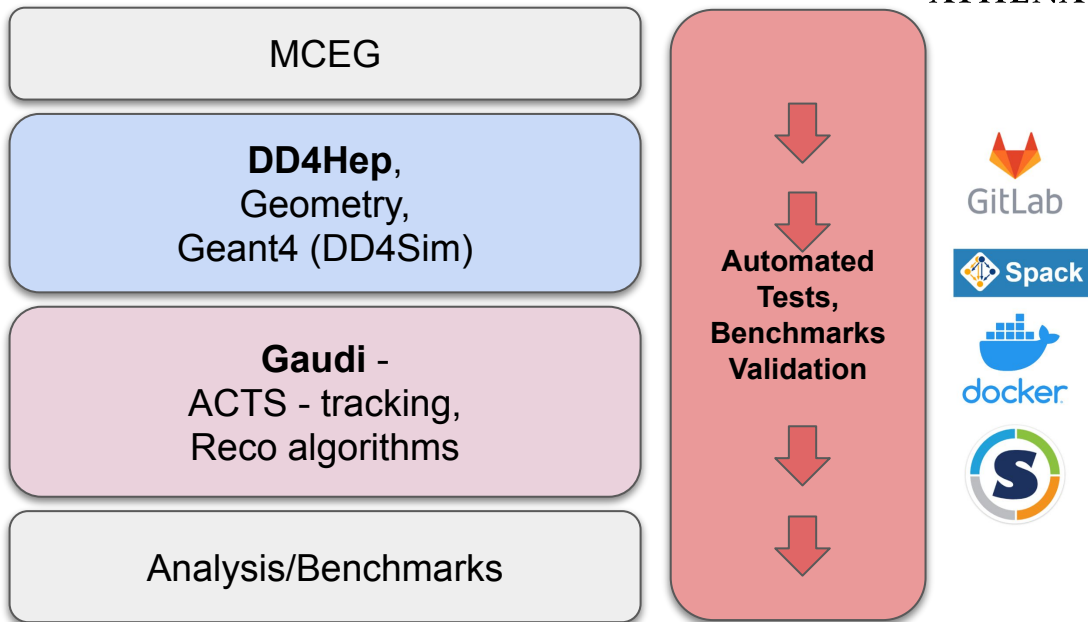
- mirrored on github.com/eic
- continuous integration
- dedicated build cluster

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

Automated Workflows at eicweb

High Level Requirements

- Failures should show up in minutes, but benchmarking may take longer.
- Merge request cannot be held up by checks for more than ~two hours.
- Modular framework with large dependencies (root, geant4, ACTS) results in large containers, which must be distributed efficiently to CI nodes.
- Access for new users at institutions without MOU/NPUA (e.g. EIC India) without a lengthy approval process.
- Compliance with data management, export controls (private repositories).

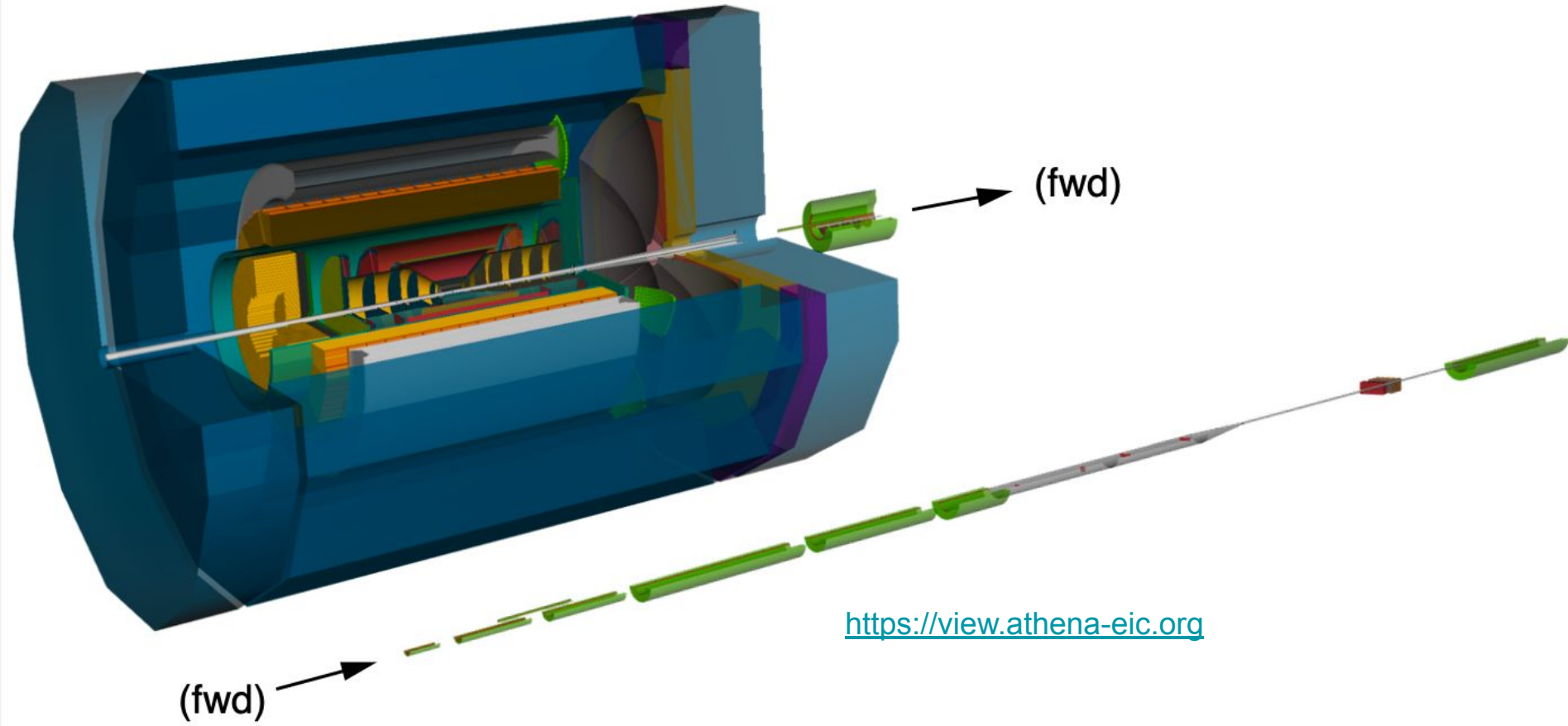
Infrastructure at ANL

- Dedicated servers with 1500+ job slots to run ~10-minute to 3-hour long CI pipelines of ~50 to ~100 individual steps for commits and merge requests. (Github free tier: 2k mins / month)
- Docker build server of 128 cores and high bandwidth to distribute to nodes with modified gitlab-runners running unprivileged singularity containers.
- Integrated development environment with kubernetes for container shells.
- Full control over users accounts, no lab account required.



This infrastructure was not available on short notice at either BNL or JLab.

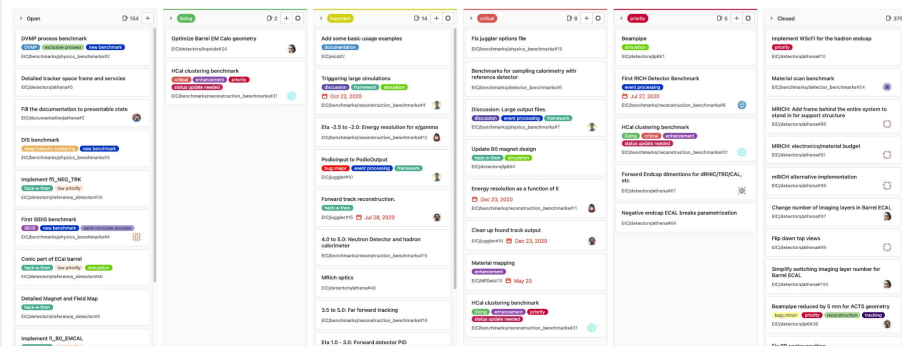
Automatic Visualization



<https://view.athena-eic.org>

Thoughts on CI-centered workflows

- **Continuous (and pain free) upgrade path** made possible by automated benchmarks and analyses for validation. This enabled us to stay up-to-date with our core software packages and pick up improvements where needed (we went from ACTS 8 → 16 in the last half year!)
- Automatic testing catches issues before they become problems
- CI workflow enabled rapid incremental development while ensuring resilience.
- Merge-request workflow coupled with extensive CI chain made it effective to on-board new developers (this worked well to get many new people productive while under time pressure!):
 - Start with empty benchmark
 - Implement reconstruction algorithm in Juggler
 - Implement reconstruction benchmark and merge both once functioning
 - Write analysis (or analysis benchmark!)



- Centralized development board and issue trackers were invaluable to effectively manage development tasks
- They greatly aided discoverability, making it much easier for people to find/check out a task

Test	Docs	Collect	Flavor	Deploy	Download	Config	Initiate	Data Job	Simulate	Calibrate	Benchmarks	Collect	Deploy	Trigger	Download	Config	Initiate	Run
test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01	test_run_01
test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02	test_run_02
test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03	test_run_03
test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04	test_run_04
test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05	test_run_05
test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06	test_run_06
test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07	test_run_07
test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08	test_run_08
test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09	test_run_09
test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10	test_run_10
test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11	test_run_11
test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12	test_run_12
test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13	test_run_13
test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14	test_run_14
test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15	test_run_15
test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16	test_run_16
test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17	test_run_17
test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18	test_run_18
test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19	test_run_19
test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20	test_run_20

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.

Software & Computing Philosophy

Encourage Upstream Contributions

- Requirements of well-formed HepMC as input has resulted in real improvements to multiple MCEGs used by EIC community.
- Various upstream contributions to DD4hep, ACTS, Spack, uproot,...

Encourage Social Coding

- CI platform provides the incentive for developers to commit code frequently: achieving data management and analysis preservation goals.
- Merge request reviews ensure higher quality code and build developer skills.

Enable Access Without Restrictions

- ATHENA collaboration members include INFN, EIC India, LBL, ANL,...
- Data 'publicly' available through BNL S3 and publicly available through JLab xrootd.
- Flat data structures (i.e. could be a csv), stored as ubiquitous ROOT trees without need for data structure libraries.
- Support for uproot using numpy library (awkward not needed).

Approaches Under Evaluation

- Rucio for data management
- Reana for analysis workflows

Deployment With Containers: Layout

Core operating system (updates: ~6 months):



- debian “bookworm” slim (2021-12-20)
- gcc-11 and clang-13 toolchains

Dependencies (updates: ~2 months, rolling):



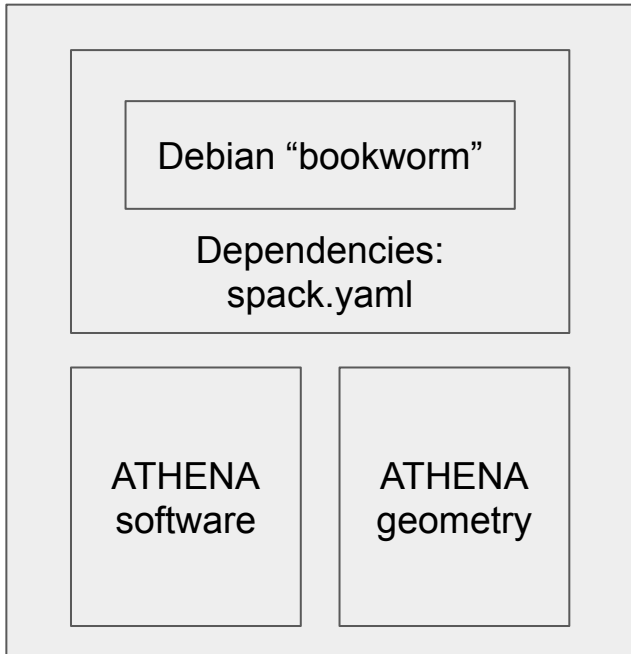
- containerized spack environment (0.17++)
- binary buildcache packages written to docker buildkit cache mount in CI jobs
- “thin” repository of custom packages:
 - bugfix patches, new versions, new packages awaiting upstream acceptance (see useful pattern below)
 - high point ~20 pkgs, now back to 7

Debian “bookworm”

Dependencies:
spack environment

```
from spack.pkg.builtin.acts import Acts as BuiltinActs
class Acts(BuiltinActs):
    version('16.0.0', commit='9bd86921155e708189417b5a8019add10fd5b273', submodules=True)
    version('15.1.0', commit='a96e6db7de6075e85b6d5346bc89845eeb89b324', submodules=True)
    version('15.0.0', commit='0fef9e0831a90e946745390882aac871b211eaac', submodules=True)
```


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

Deployment With Containers: Versioning & Distribution


Container versioning:

- nightly (ATHENA master branches)
- stable (released ATHENA versions)
- unstable (merge requests)

Container registries:

- eicweb (GitLab): all containers, private
 - actually mostly for singularity containers
- Docker Hub: nightly, stable, stable-\$(date)

CVMFS singularity sandboxes:

- 
- The OSG logo, consisting of three overlapping, stylized, orange and yellow geometric shapes resembling a stack of books or a series of parallel lines.
- using OSG ~6 hour synchronizations, to `/cvmfs/singularity.opensciencegrid.org`
 - distribution to clients on OSG, users at large lab facilities, end users with CVMFS

User access to containers:

- goals: quick, transparent to user
- `curl -L get.athena-eic.org | bash`
- `./eic-shell`

If CVMFS found:

- use auto-updating sandbox image

If CVMFS not found:

- singularity pull sandbox image
- `./eic-shell --upgrade`

Other features: use gpfs, automatic bindpath detection, use singularity from `/cvmfs/oasis.opensciencegrid.org`

Anatomy Of ATHENA Jobs

Input:

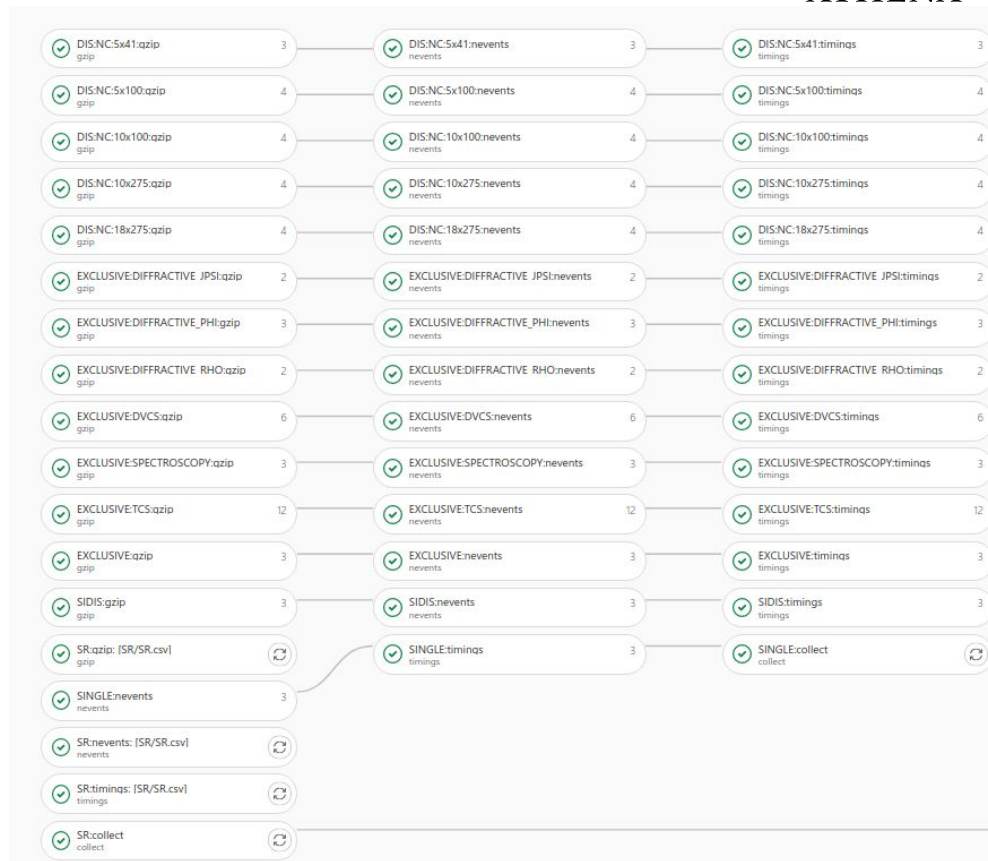
- S3: mc cp of HepMC v3 files (gzipped)
- condor: transfer DD4hep gun steer file

Benchmarking on eicweb as part of CI:

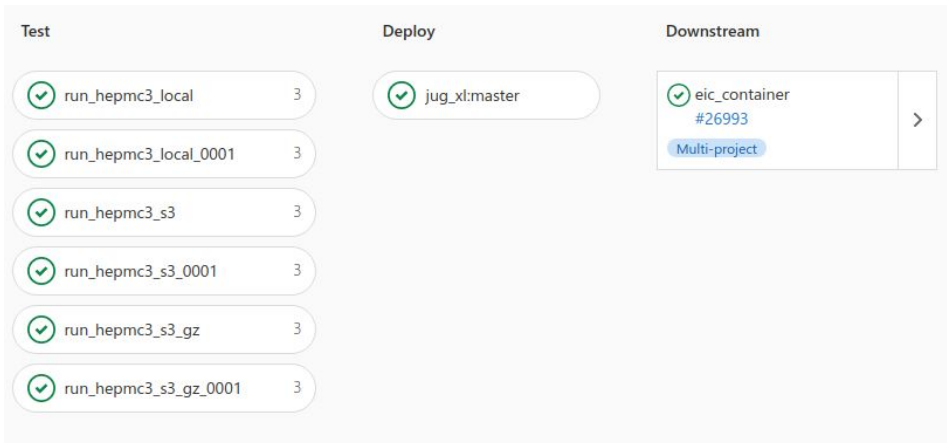
- running test, smoke tests, sanity checks, time-per-event determination into csv
- target time: slurm: ~20 hrs, condor: ~2 hrs

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



Anatomy Of ATHENA Jobs



Structure on job node:

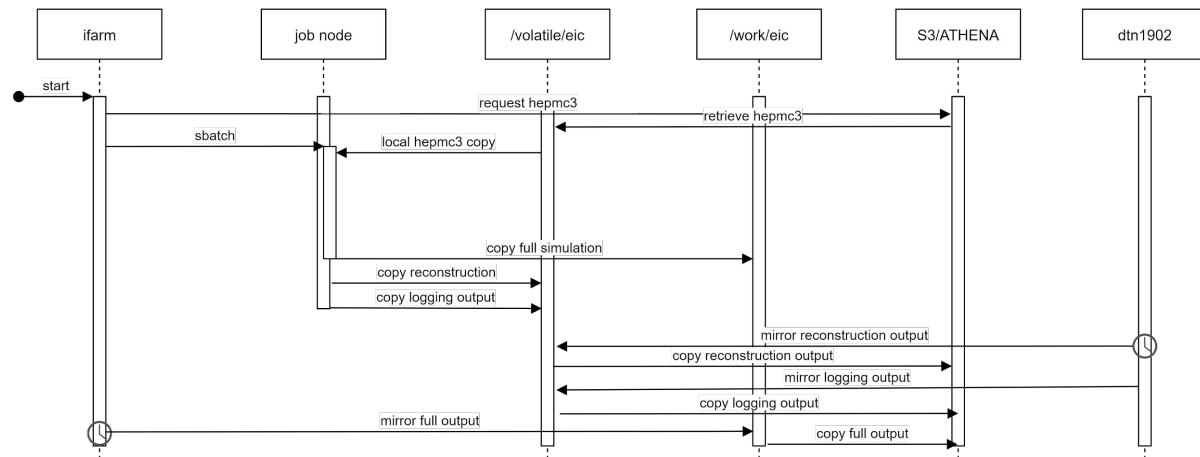
- S3 retrieval (inside the job, not using OSG transfer_input_file, no pre-signed urls)
- run simulation (artifacts downloaded as needed, or found in container cache)
- S3 upload of full simulation podio output
- run reconstruction (artifacts downloaded as needed, or found in container cache)
- S3 upload of reconstruction eicd output

Interactions with OSG on best practices:

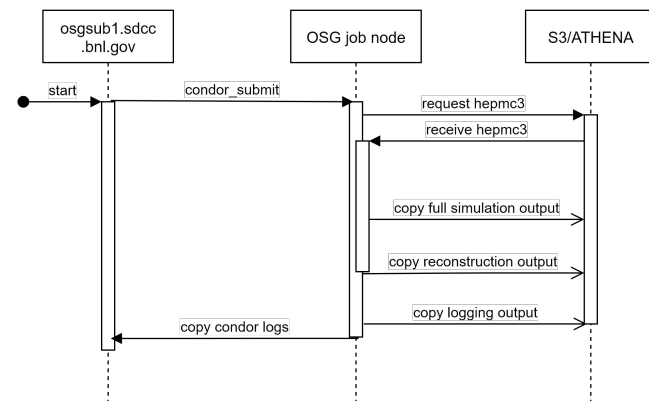
- hold release based on log parsing
- secrets transmission to nodes (env.sh)
- delayed job instantiation (max_idle)
- queue from csv
- reporting misbehaving nodes

Operational Benefits of OSG Jobs

Running at JLab (capacity 25k job slots, 14% for EIC)



Running on OSG (capacity 50k)



Typical Full Simulation & Reconstruction Timings

Typical memory used: ~1.5 GB/process

Typical fixed overhead: ~60 s/process

Single particle initial states:

DIS NC ($Q^2 > 1$, $Q^2 > 10$, $Q^2 > 100$, $Q^2 > 1000$)

- 5x41: 1.3 s/ev, 1.8 s/ev, 2.5 s/ev
- 10x100: 2.3 s/ev, 2.9 s/ev, 3.9 s/ev, 5.2 s/ev
- 18x275: 5.6 s/ev, 6.1 s/ev, 7.6 s/ev, 10.2 s/ev

SIDIS

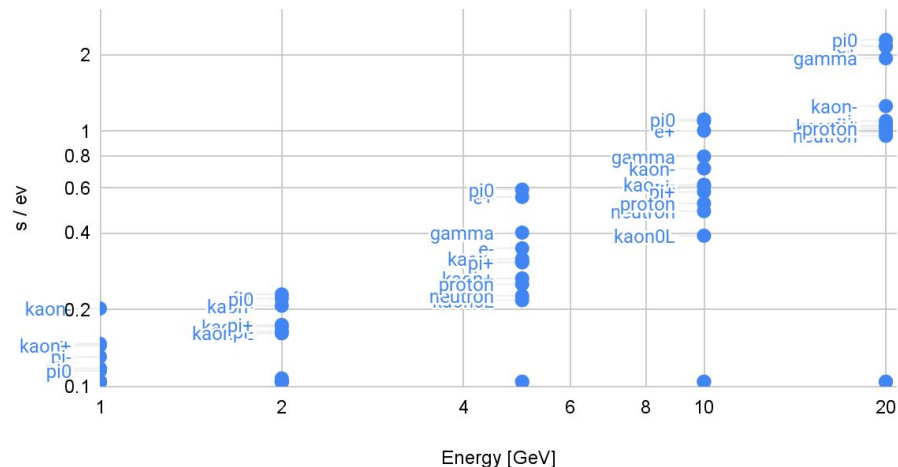
- Lambda 18x275: 7.4 s/ev
- eAu 18x110: 2.7 s/ev

Diffractive J/psi

- eAu: 3.8 s/ev

Dual EPYC 7H12 (256 threads, 512 GB, 2 GB/thread)
HEP-SPEC06 ~20/thread (no actual benchmark available, based on NIKHEF Physics Data Processing Facility Dual EPYC 7H12 without hyperthreading)

Barrel



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).

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?