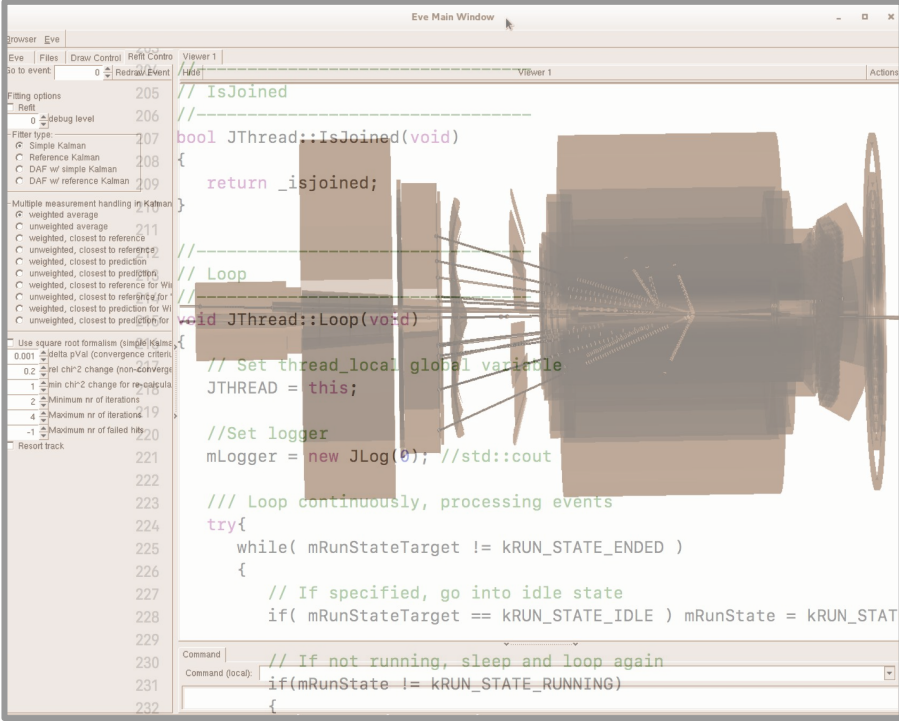


JLEIC Software Status

Status of Simulation and Reconstruction Software at JLEIC

David Lawrence (JLab)

on behalf of the JLEIC Detector and IR Study Group
<https://www.eiccenter.org/content/jleic-detector-and-ir-study-group>



The screenshot displays the 'Eve Main Window' of the JLab software. On the left, a sidebar contains various configuration options such as 'Fitting options', 'Filter type', and 'Multiple measurement handling in Kalman'. The main area is a C++ code editor showing the implementation of a thread class. The code includes methods for checking if the thread is joined, a loop for processing events, and state management. A 3D visualization of a detector component is visible in the background of the code editor.

```
// IsJoined
//-----
bool JThread::IsJoined(void)
{
    return _isjoined;
}

// Loop
void JThread::Loop(void)
{
    // Set thread local global variable
    JTHREAD = this;

    //Set logger
    mLogger = new JLog(0); //std::cout

    /// Loop continuously, processing events
    try{
        while( mRunStateTarget != kRUN_STATE_ENDED )
        {
            // If specified, go into idle state
            if( mRunStateTarget == kRUN_STATE_IDLE ) mRunState = kRUN_STATE_IDLE;

            // If not running, sleep and loop again
            if(mRunState != kRUN_STATE_RUNNING)
            {
                std::this_thread::sleep_for(mSleepTime); //Sleep a minimal amount.
                continue;
            }

            //Check if not enough event-tasks queued
            if(CheckEventQueue())
            {
                //Process-event task is submitted, redo the loop in case we want to buffer
                continue;
            }
        }
    }
```

Simulation and Reconstruction

Near Term Goals/Purpose:

1. Study ability to measure Physics Processes
 - a. Interface to MC Event Generators
 - b. Provide detector responses (Fast MC for acceptance/resolution, ab initio for resolution/backgrounds)

2. Study/Refine Detector Design
 - a. Interface to MC Event Generators
 - b. Access to simulation from alternative designs
 - c. Provide detector responses (Fast MC for acceptance/resolution, ab initio for resolution/backgrounds)

(JLab) Experimental Computing Performance Plan

- **Insure adequate computing resources with \$850K investment in FY18**

- Use local farm for reconstruction, calibration and analysis
- Use distributed resources for MC
- Storage and associated bandwidth scaled to support all resources

- **Open Science Grid**

- GlueX -6 institutions contribute resources
- In a recent 2 week period ~1M core-hours
- Expect yearly 35M-50M core-hours
- Investigating options for CLAS12

- **GlueX reconstruction code at NERSC**

- Scale test in July
- Anticipate 70M core-hours/year

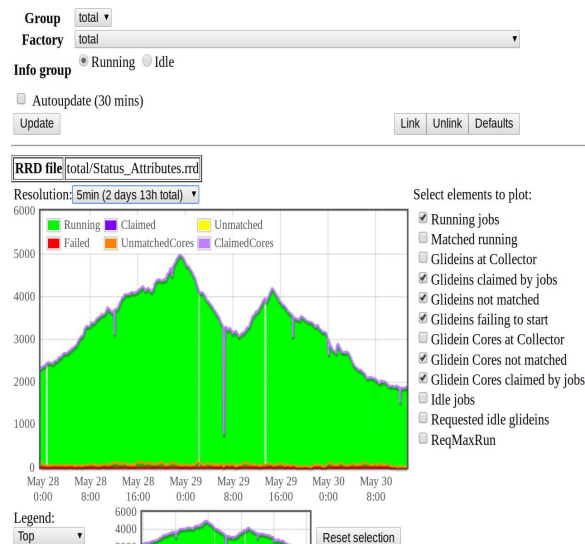
- **Cloud Computing available for bursts**

	Current	FY19	FY20
CPU (M-core-hours/year)	37	70	90
Scratch Disk & Cache Disk (PB)	0.65	1.1	2
Tape (GB/s)	3	5	7
WAN bandwidth (Gbps)	10	10	10

Current and Projected Capacity

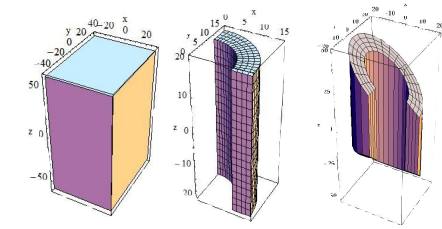
VO frontend status - GluexVO-1_0

[\[Browse \]](#) [\[Group Matrix \]](#) [\[Group Graphs \]](#)

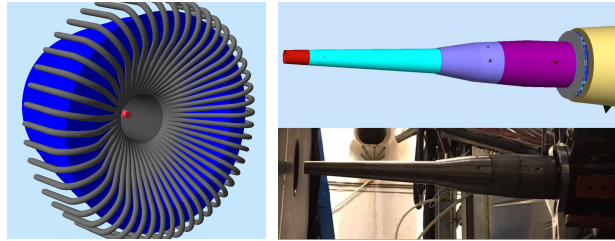


GEMC Framework

Maurizio Ungaro

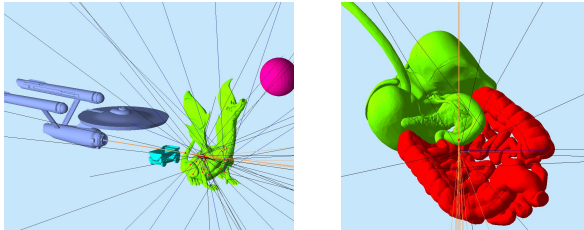


Native
Geant4



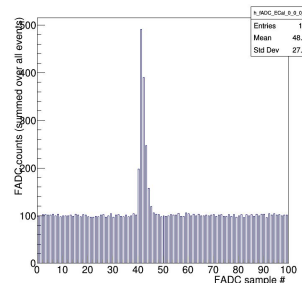
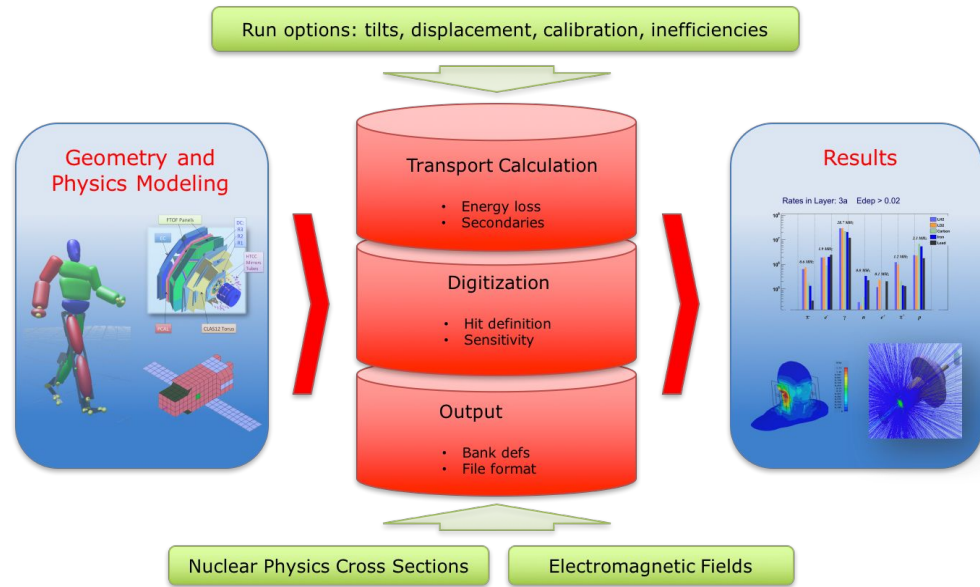
CAD

Many CLAS12 detectors use actual engineering models (STL)

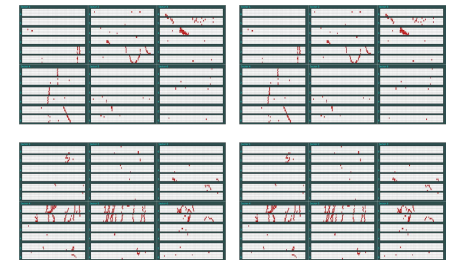


GDML

- Input: Native, CAD, GDML, can be mixed and matched.
- FADC Mode 1 (crate, slot channel)
- Background Merging.
- FAST MC Mode.
- Digitization uses actual CCDB calibration constants.
- Used for CLAS12



GEMC EC FADC Mode 1



Merging of random trigger data with GEMC

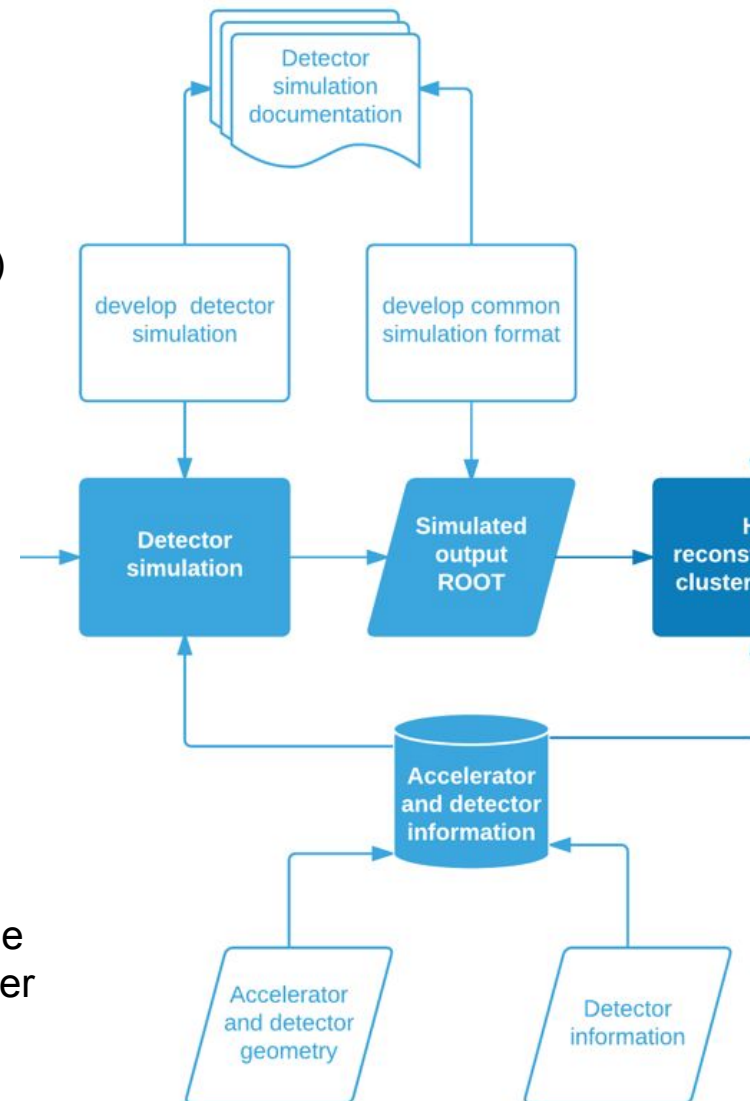
GEMC for JLEIC

Ideal for detector concepts

- application for detector simulations based on Geant4
- reducing the learning curve for Geant4 simulations
- macro language for detector design
- various geometry definitions (GEMC, CAD, gdml, ROOT)
- data card (XML) to steer application, all Geant4 macro commands supported by design
- GUI for interactive sessions
- excellent documentation
- full Geant4 support: adding Geant4 features relatively simple
- transparent in-house development

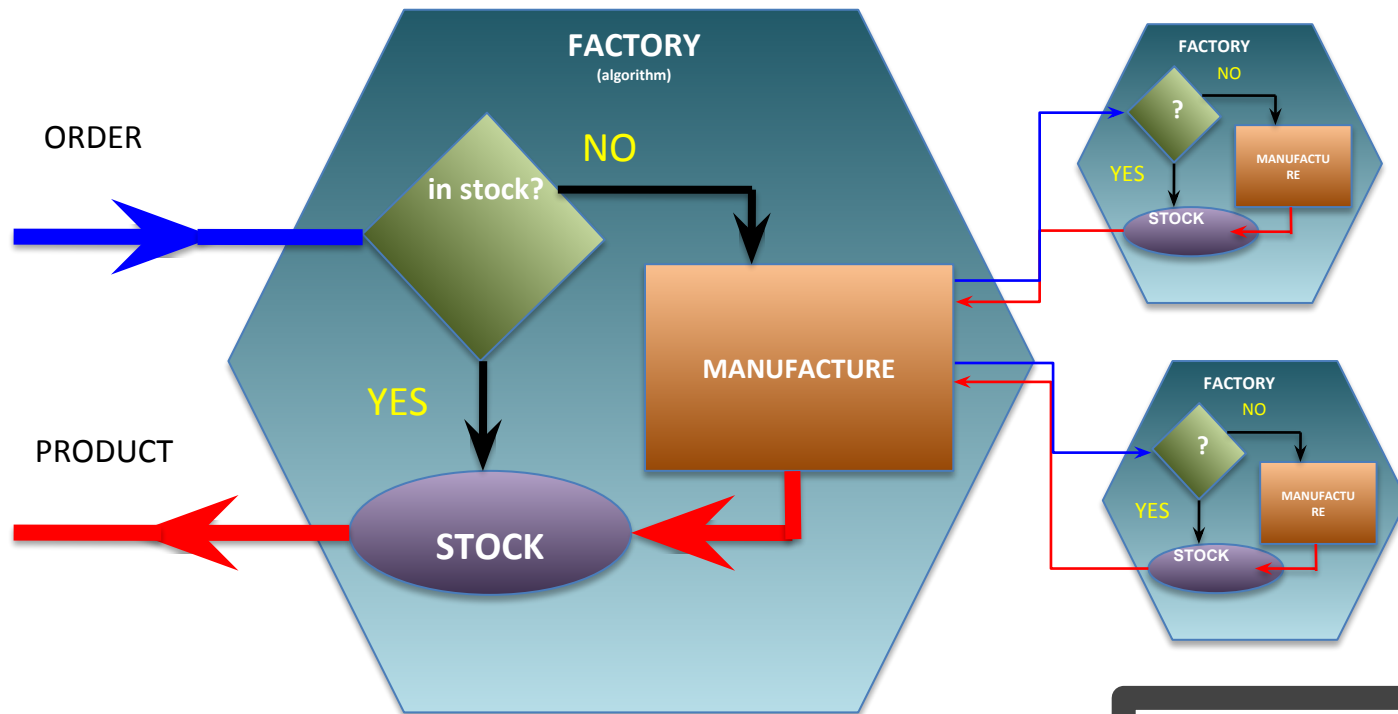
Simulations level

- same application for fast and full detector simulations
- fully adjustable simulation levels, e.g.,
 - only material transport
 - using Geant4 for geometry and physics only in some critical areas and ad-hoc non-Geant4 models in other regions



JANA: C++ Software Framework for Reconstruction Workflow

JANA Factory Model



Data on demand = Don't do it unless you need it
const Stock = Don't do it twice

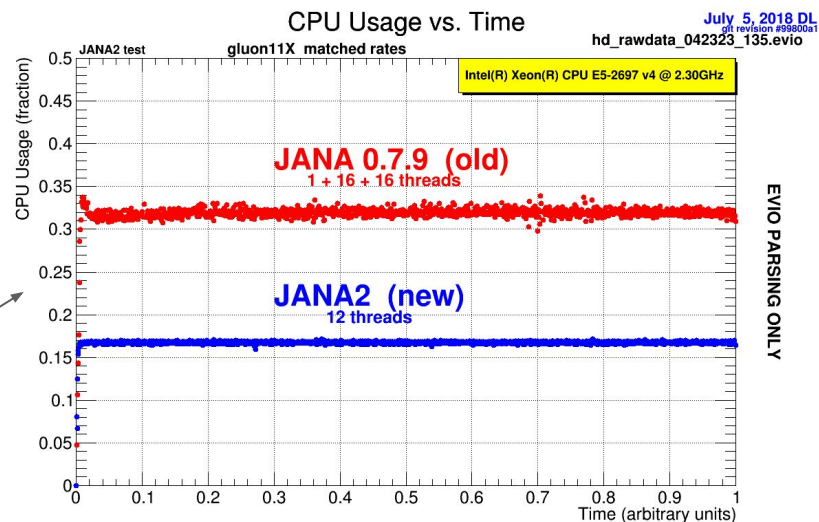
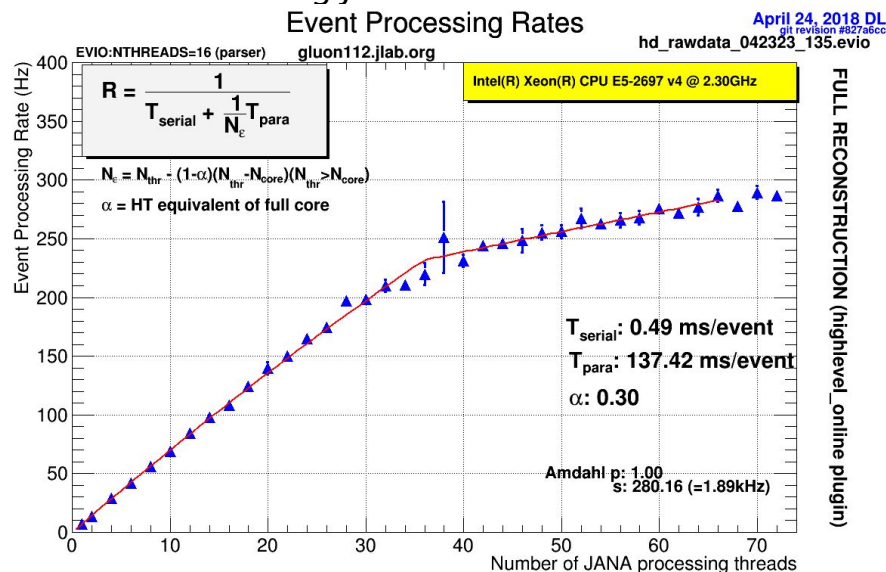
Conservation of
CPU cycles!

```
auto tracks = jevent->Get<DTrack>();  
  
for(auto t : tracks){  
    // ... do something with a track  
}
```


JANA: C++ Software framework for Reconstruction Workflow

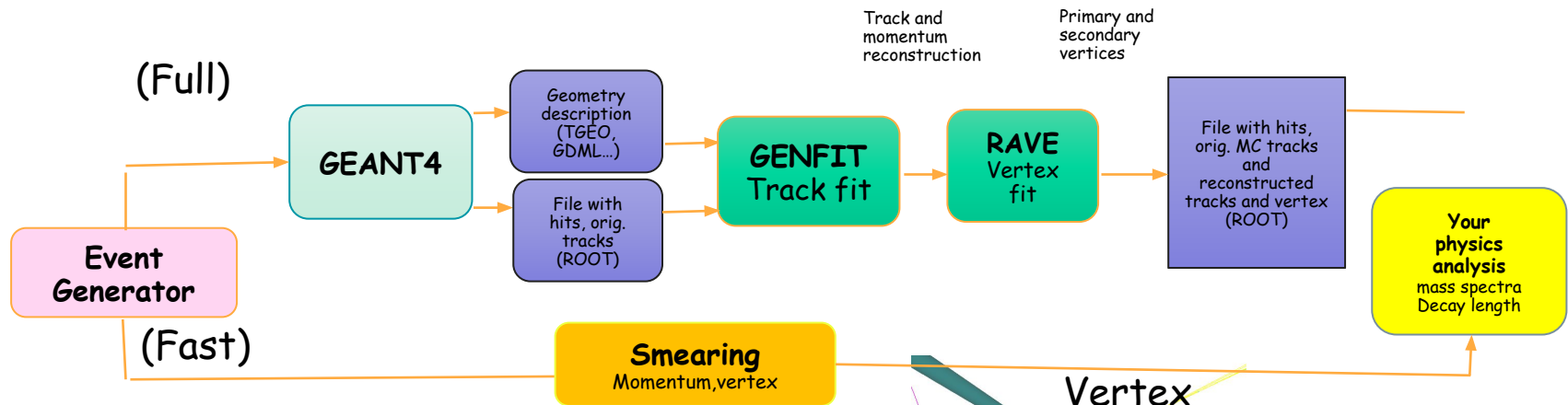
- Multi-threaded
- Modular, user-focused design
- Developed over the past 13 years specifically for 12GeV era of high rate experiments at JLab
- Used for GlueX online DQM, offline reconstruction and L3 trigger system*
- LDRD project for development of JANA2 started in FY18

JANA rate scaling for GlueX Data Reconstruction

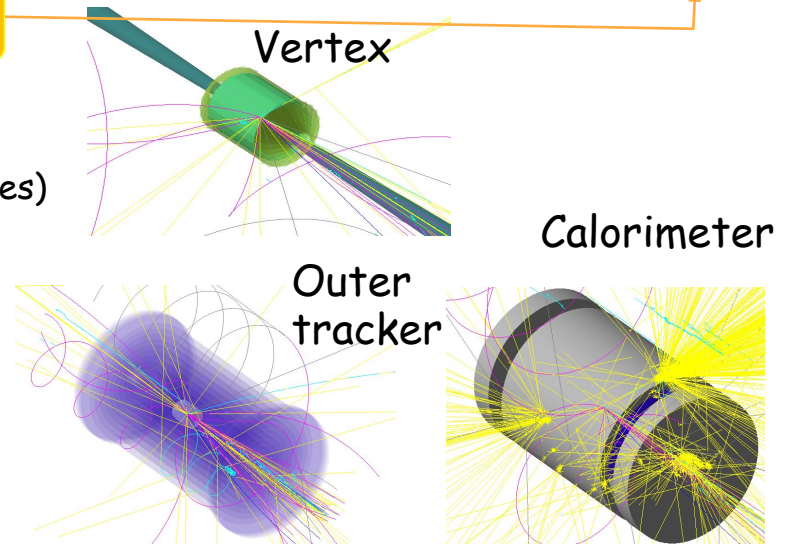


RECONSTRUCTION CHAIN (FOR LDRD)

This chain has been developed to validate tracking and vertex parameters and was used for JLAB LDRD- 1601/1701 project ("Nuclear gluons with charm at EIC") to estimate a detector effect on a charm reconstruction. (Many thanks to Whitney Armstrong, Alexander Kiselev and "software consortium" for ideas and discussions)



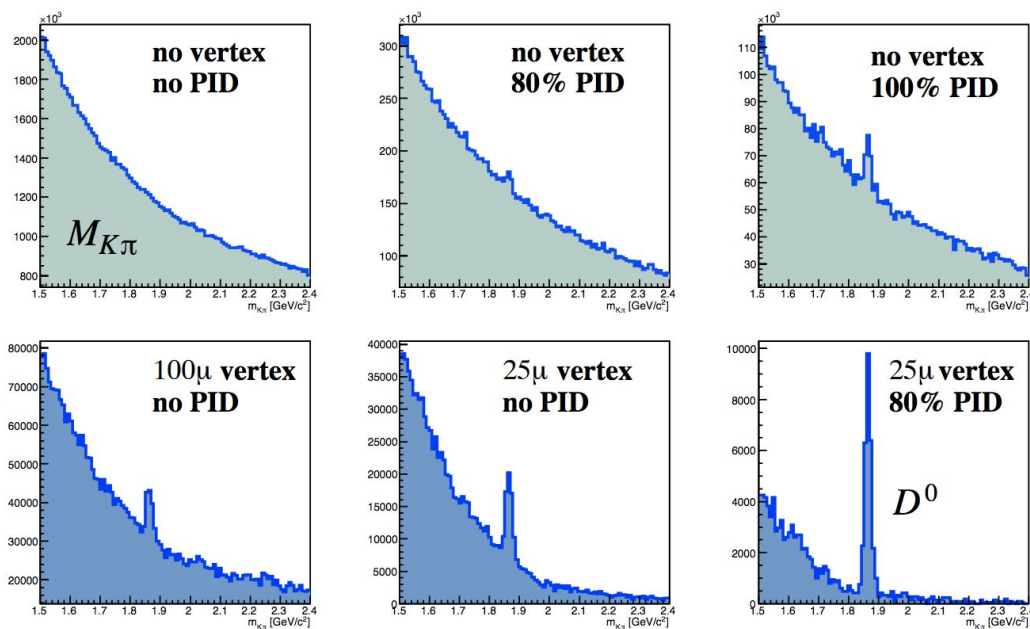
- Geometry created/described **inside GEANT4**, and then distributed via **Root TGEO**
- All parts are connected via intermediate **Root files** (ntuples)
- Event generators (Pythia6, Herwig 6 and 7) HEPMC, Lund format.
- Original (MC) tracks are traced down to analysis
- No pattern recognition ! Only track fit and vtx!



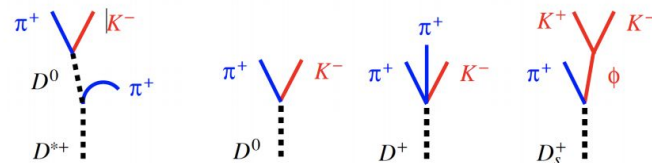
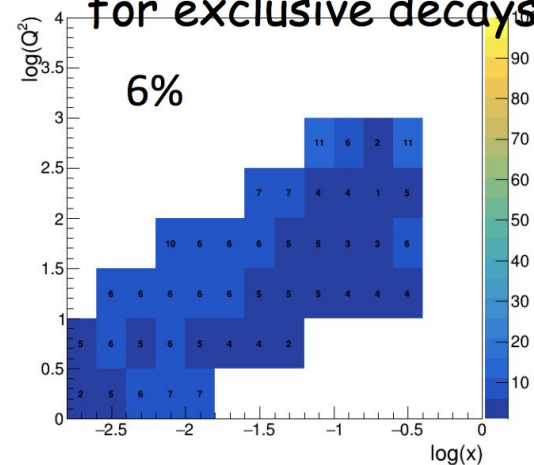
Analysis

- ✓ Process charm (BGF)-only events
- ✓ Process and add all "background" events (all other non -BGF DIS events)
- ✓ Estimate efficiency and set a requirements for detector (PID, vertex, etc)

D^0 on top of DIS background



Charm efficiency
for exclusive decays

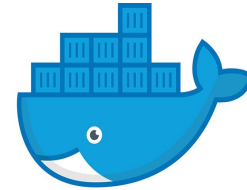




Containers

1. Install Docker or Singularity

2. Run container



docker run -p 6080:6080 -v /my/data/dir:/data -it --rm electronioncollider/jleic:1.0.4
or

singularity shell shub://electronioncollider/jleic:1.0.4 /container/utilities/xstart.csh

3. Point browser to:



http://localhost:6080

Containers

JLEIC Desktop Environment via web browser on host

The screenshot displays a web browser window with the address bar showing 'localhost:6080/vnc.html'. The browser content shows a VNC desktop environment. On the desktop, there are two windows:

- Examples.md - /eic/doc/examples/**: A text editor window showing the contents of a README file. The text includes sections for Introduction, Quick Start, and Details.
- eicuser@ba79a6b26067:/eic/app/jlab**: A terminal window showing the JLEIC environment setup. It lists installed software versions and provides instructions for getting started.

```
JLEIC container example README
Jan. 31, 2018 David Lawrence

# INTRODUCTION
This provides an example for exercising JLEIC simulation software in this container.

# Quick Start
## View Geometry
1.) cd /eic/doc/examples
2.) gemc example.gcard

## Simulate events
1.) cd /eic/doc/examples
2.) gemc -INPUT_GEN_FILE="LUND.pythia-sample.lund" \
        -OUTPUT="evio_sample_out.evio" \
        -USE_GUI=0 \
        example.gcard
3.) evio2root -INPUTF=sample_out.evio

This should produce a file sample_out.root that can be used to browse and plot data.

## Draw Hits
1.) root sample_out.root -x DrawHits.C

Use the mouse to rotate the view.

# DETAILS
## Generated Events
gemc has a built in particle gun that can be configured via the GUI. It will also accept events read from a file in the LUND format. An example file is present in:
/eic/doc/examples

## Accessing Simulated Data
Simulated data is written to an EVIO file and then converted into a ROOT file with the evio2root (see Quick Start above).
```

```
> BANKS      version: 1.3
> CCDB       version: 1.06.02
> CLHEP      version: PRO
> EVIO       version: 5.1
> GEANT4     version: PRO
> GEMC       version: devel
> JANA       version: 0.7.7p1
> MLIBRARY   version: 1.1
> MYSQL      installed in /eic/app/jlab/2.1/Linux_CentOS7.4.1708-x86_64-gcc
4.8.5/mysql/lib
> QT         version: 5.6.2
> ROOT       version: PRO
> SCONS      version: 1.5
> XERCES     version: SYS

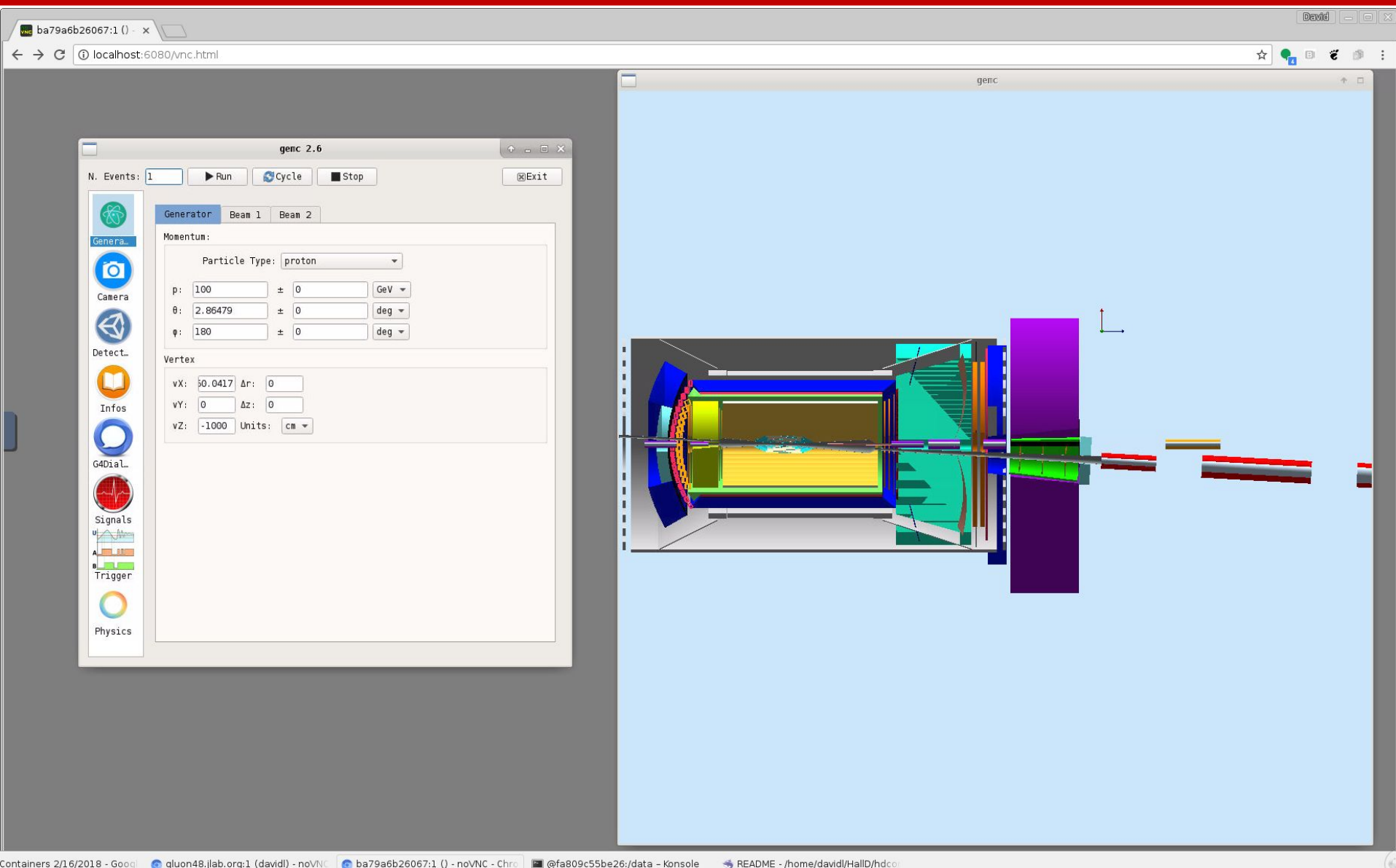
Welcome to the JELIC Container!

To get started, please read the README by typing

less /eic/doc/examples/Examples.md

[eicuser@ba79a6b26067 jlab]$
```


Containers



JLEIC Desktop Environment via web browser on host

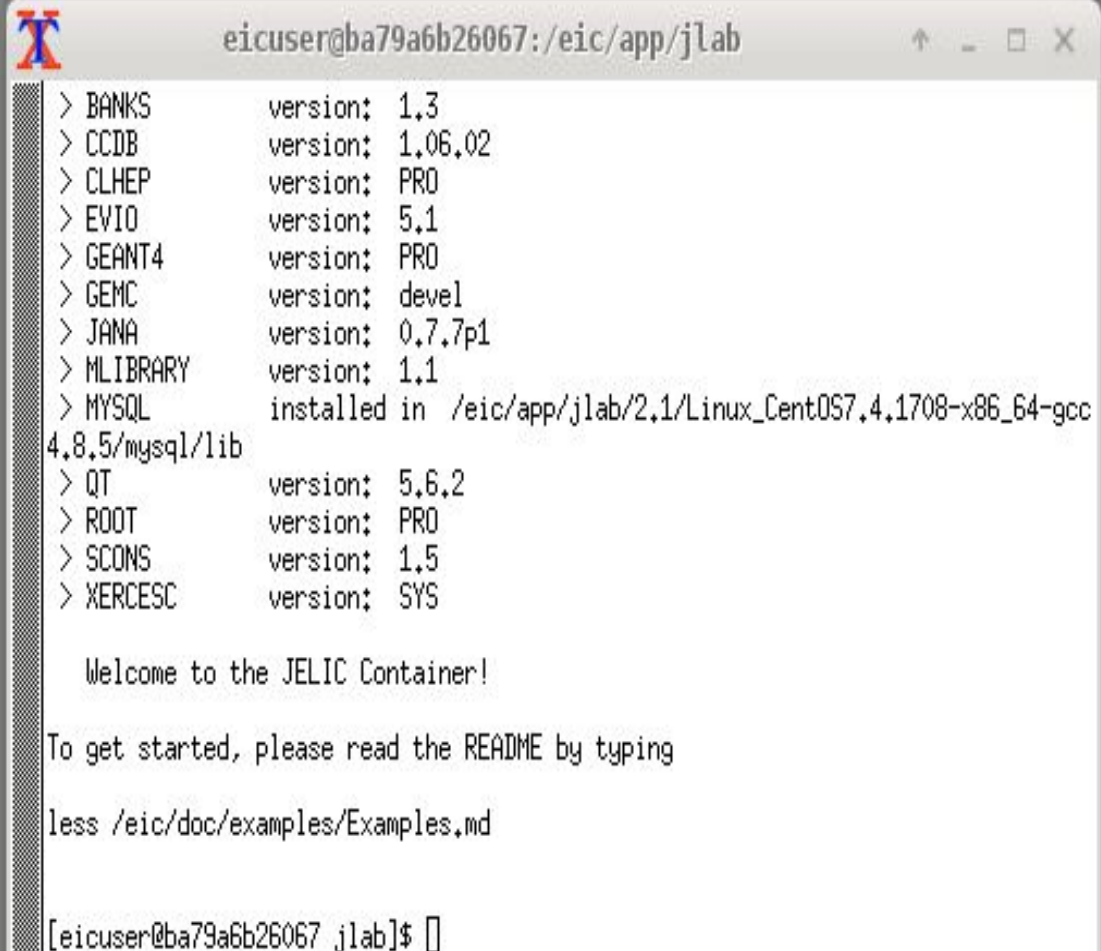


Summary and Future

- Simulation
 - GEMC (Geant4) used for simulation
 - Refining and merging geometries
- Reconstruction
 - Some work completed on tracking/vertexing using GenFit and RAVE
 - Actively integrating reconstruction components into single project using JANA framework
 - Plans to incorporate other ESC software (e.g. ProIO)
- Containers
 - jleic containers published on Docker hub and Singularity hub
 - targeting interactive desktop/laptop use vs. batch

Backups

JLEIC Desktop Environment via web browser on host



```
eicuser@ba79a6b26067:/eic/app/jlab
> BANKS          version: 1.3
> CCDB           version: 1.06.02
> CLHEP          version: PRO
> EVIO           version: 5.1
> GEANT4         version: PRO
> GEMC           version: devel
> JANA           version: 0.7.7p1
> MLIBRARY       version: 1.1
> MYSQL          installed in /eic/app/jlab/2.1/Linux_CentOS7.4.1708-x86_64-gcc
4.8.5/mysql/lib
> QT             version: 5.6.2
> ROOT           version: PRO
> SCONS          version: 1.5
> XERCESC        version: SYS

Welcome to the JELIC Container!

To get started, please read the README by typing

less /eic/doc/examples/Examples.md

[eicuser@ba79a6b26067 jlab]$
```

JLEIC Desktop Environment via web browser on host

JLEIC container example README

Jan. 31, 2018 David Lawrence

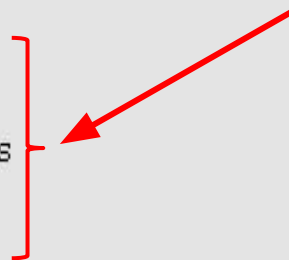
INTRODUCTION

This provides an example for exercising JLEIC simulation software in this container.

Quick Start

View Geometry

- 1.) `cd /eic/doc/examples`
- 2.) `gemc example.gcard`



Simulate events

- 1.) `cd /eic/doc/examples`
- 2.) `gemc -INPUT_GEN_FILE="LUND,pythia-sample.lund" \`
`-OUTPUT="evio,sample_out.evio" \`
`-USE_GUI=0 \`
`example.gcard`
- 3.) `evio2root -INPUTF=sample_out.evio`

This should produce a file `sample_out.root` that can be used to browse and plot data.

JLEIC Desktop Environment via web browser on host

JLEIC container example README

Jan. 31, 2018 David Lawrence

INTRODUCTION

This provides an example for exercising JLEIC simulation software in this container.

Quick Start

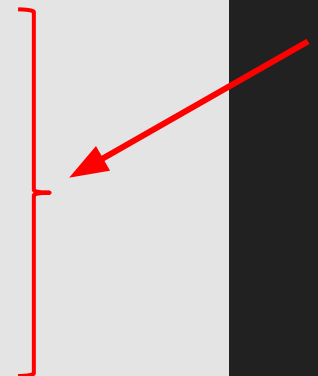
View Geometry

- 1.) `cd /eic/doc/examples`
- 2.) `gemc example.gcard`

Simulate events

- 1.) `cd /eic/doc/examples`
- 2.) `gemc -INPUT_GEN_FILE="LUND,pythia-sample.lund" \
-OUTPUT="evio,sample_out.evio" \
-USE_GUI=0 \
example.gcard`
- 3.) `evio2root -INPUTF=sample_out.evio`

This should produce a file `sample_out.root` that can be used to browse and plot data.



Secure | <https://jeffersonlab.github.io/swcarpentry-jlab-singularity/02-jlabce-container/index.html>

Home Code of Conduct Setup Episodes ▾ Extras ▾ License Improve this page ✎ Search...

Using Singularity at Jefferson Lab

Running the JLab CE container

Overview

Teaching: 5 min
Exercises: 10 min

Questions

- How can we replicate the Jefferson Lab Common Environment on other systems?

Objectives

- Understand how tags are used to version containers.
- Load and use the Jefferson Lab Common Environment on the interactive farm nodes.

Tags: versioning of containers

In the previous episode we downloaded the lolcow container, at `shub://GodloveD/lolcow` and the container was stored with the filename `GodloveD-lolcow-master-latest.simg`. Let's analyze that URL and filename.

- `shub` indicates that the URL points to a Singularity Hub location.
- `GodloveD` is the user who provided this container (David Godlove, if you must know, see for example this [GitHub page](#)).
- `lolcow` is the name of the repository that was used to build this container.
- `master` is the branch from which the container was built.
- `latest` is the tag of the container, with latest for the most recent build.

The tag is commonly used for versioning of containers. By specifying the URL as `shub://GodloveD/lolcow:latest` we can explicitly ask for the latest version of the lolcow container.

Since there is not a lot of versioning one can do on this container, we will first introduce a container where versions ARE important.

Retrieving the Jefferson Lab Common Environment container from Singularity Hub

Now that we have the basics of containers behind us, we can use our first 'useful' physics container: the Jefferson Lab Common Environment container. This container replicates the scientific software suite that is installed on the interactive farm nodes, but packages it up in a nice container.

Of course, there is no real practical need to load the Jefferson Lab Common Environment on a Jefferson Lab interactive farm node, but bear with me for now.

Here is how we download the container:

JLab Software Carpentry Workshop:

EIC container effort
feeding back into
production operations at
JLab