CLAS Software

D.P. Weygand

Outline

- Brief history of CLAS12 Software
- Current Status/ Future Plans
  - Framework (ClaRA)
  - Simulation
  - Reconstruction
    - Tracking
    - Calorimetry
    - Cerenkov Counters
    - TOF

CLAS12 Software Goals

Robust & Fault Tolerant

OnLine Calibration

Architecture designed to Last Decades through Language evolution

Fast & Efficient Event Reconstruction

Easy Access to Reconstructed data

Conference-Ready analysis in Months

CLAS12 Software Evolution

Project	Physics Goal	Method	Result
MOMRES	calculate momentum and angular resolution of CLAS12 detector	analytical calculation of momentum resolution using location and accuracy of measurement points and location of multiple scattering material, in an inhomogeneous magnetic field	detector accuracy specifications adequate to achieve physics specifications
FASTMC	simulate experiment counting rates and resolutions	parameterize the acceptance and momentum resolution of the detector as simple functions and look-up tables	verified experiment's feasibility, many received PAC approval
GEMC	simulate realistic events including background	GEANT4-based code with realistic detector geometry and hit digitization and realistic background hit simulation	produced realistic events to develop reconstruction codes
SOCRAT	verify MOMRES results on resolution; feasibility of $L = 10^{35}$	ROOT-based code using a Hough transform for pattern recognition and a Kalman-filter fitting routine	full "3-d" tracking results verified good resolution and efficiency at $L=10^{35}$
CLARA	Unified framework for physics data processing	Service Oriented Architecture Cloud computing	CLAS12 event reconstruction in a cloud
SOT	demonstrate tracking within the CLARA framework	JAVA- based code broken into several 'services'	reproduced SOCRAT results; achieved high event throughput

CLAS12 Software Goals

- Challenges of the physics data processing environment
  - Long lifetime, evolving technologies
- Complexity through simplicity
  - Build complex applications using small and simple components.
- Enhance utilization, accessibility, contribution and collaboration
  - Reusability of components
  - Integration of legacy and /or foreign components
  - On-demand data processing.
  - Location independent resource pooling.
- Multi-Threading
  - Effective utilization of multicore processor systems.
- Software agility
  - Alter software application dynamically
  - Robust and fault tolerant
- Production Processing
  - Dynamically add and remove resources

Framework: Current Status

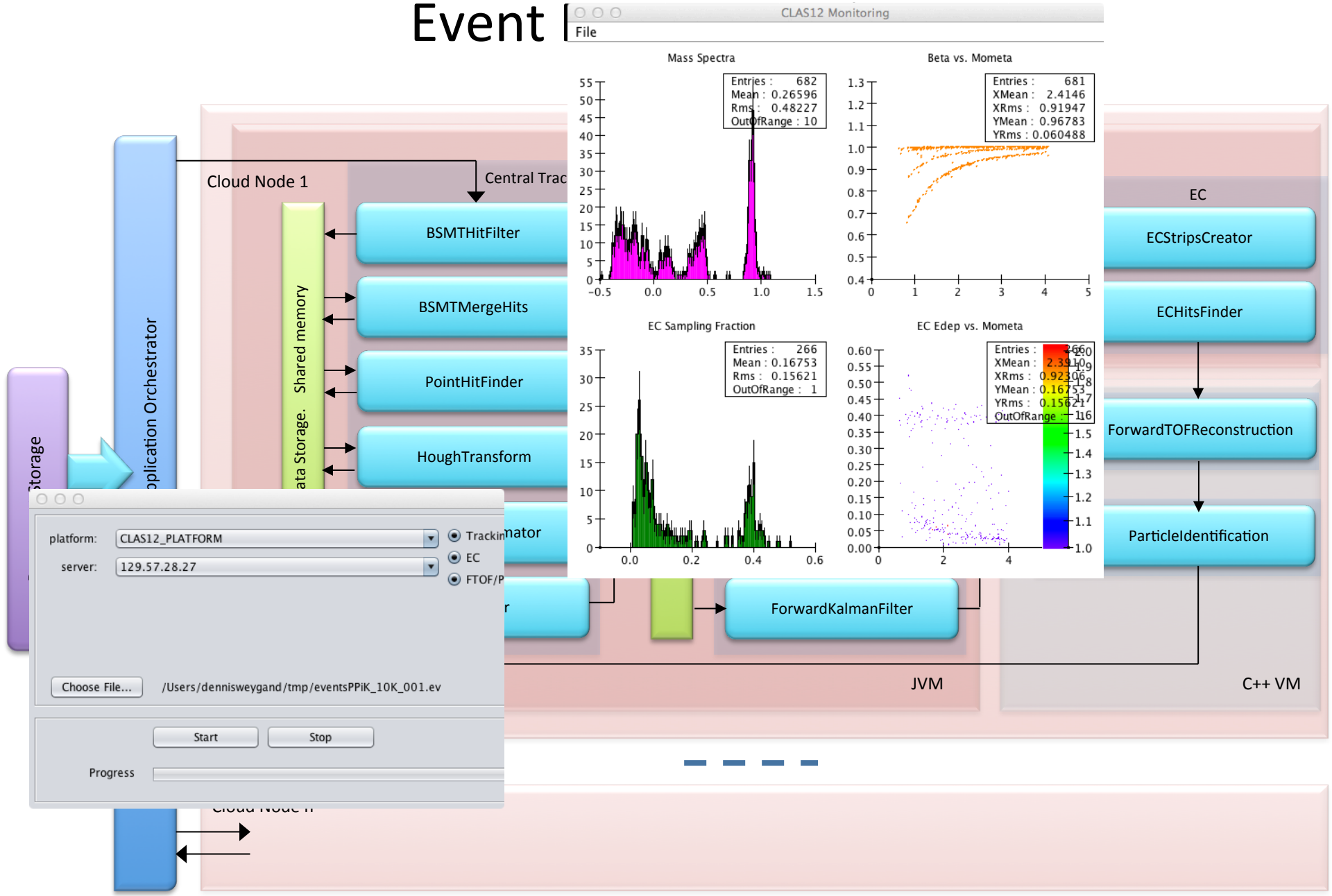
- SOA is architecture choice for CLaRA to leverage cloud computing
- Data processing major components as services.
- Application design based on services
- ClaRA supports both traditional and cloud computing models.
  - Centralized batch processing
  - Distributed cloud processing

Current Status

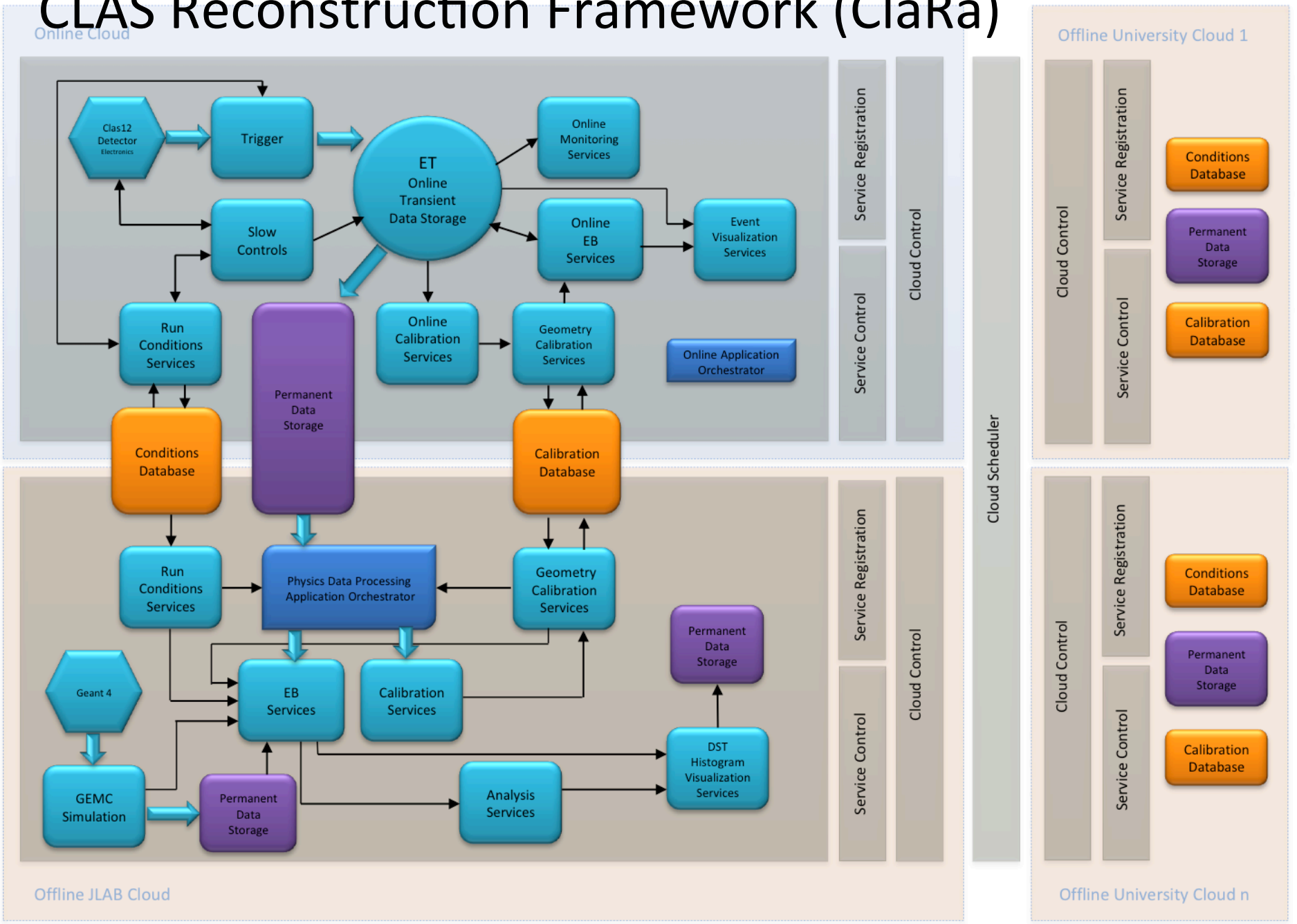
- A multi-treaded analyses framework, based on SOA
- PDP application based on specialized services
  - Small, independent
  - Easy to test, update and maintain
  - Building and running PDP application does not require CS skills.
- List of applications has been developed using the framework
  - Charge particle tracking
    - Central
    - Forward
  - EC reconstruction
  - FTOF reconstruction
  - PID
  - Detector calibration
  - Event Building
  - Histogram services
  - Database application
    - Calibration services
    - Geometry and run conditions services

ClaRA supports both traditional and cloud computing models and if need be we are ready for cloud deployment.

Event

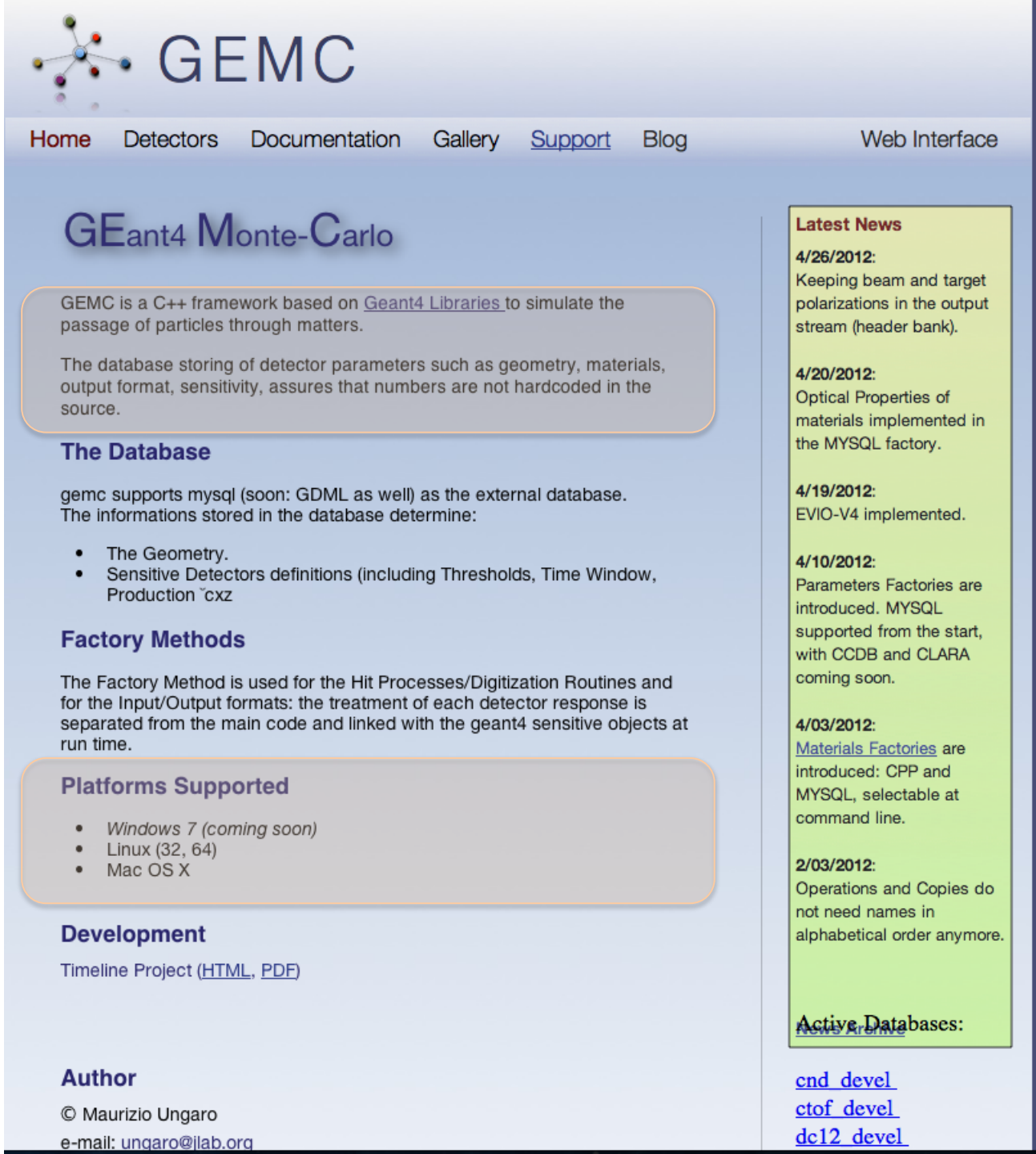

CLAS12 Monitoring
File
Mass Spectra
Entries : 682
Mean : 0.26596
Rms : 0.48227
OutOfRange : 10
Beta vs. Mometa
Entries : 681
XMean : 2.4146
XRms : 0.91947
YMean : 0.96783
YRms : 0.060488
EC Sampling Fraction
Entries : 266
Mean : 0.16753
Rms : 0.15621
OutOfRange : 1
EC Edep vs. Mometa
Entries : 266
XMean : 2.3910
XRms : 0.92306
YMean : 0.16753
YRms : 0.15621
OutOfRange : 1
Cloud Node 1
Central Trac
Shared memory
BSMTHitFilter
BSMTMergeHits
PointHitFinder
HoughTransform
EC
ECStripsCreator
ECHitsFinder
ForwardTOFReconstruction
ParticleIdentification
ForwardKalmanFilter
JVM
C++ VM
platform:
CLAS12_PLATFORM
server:
129.57.28.27
Choose File...
/Users/dennisweygand/tmp/eventsPPiK_10K_001.ev
Start
Stop
Progress

CLAS Reconstruction Framework (ClaRa)



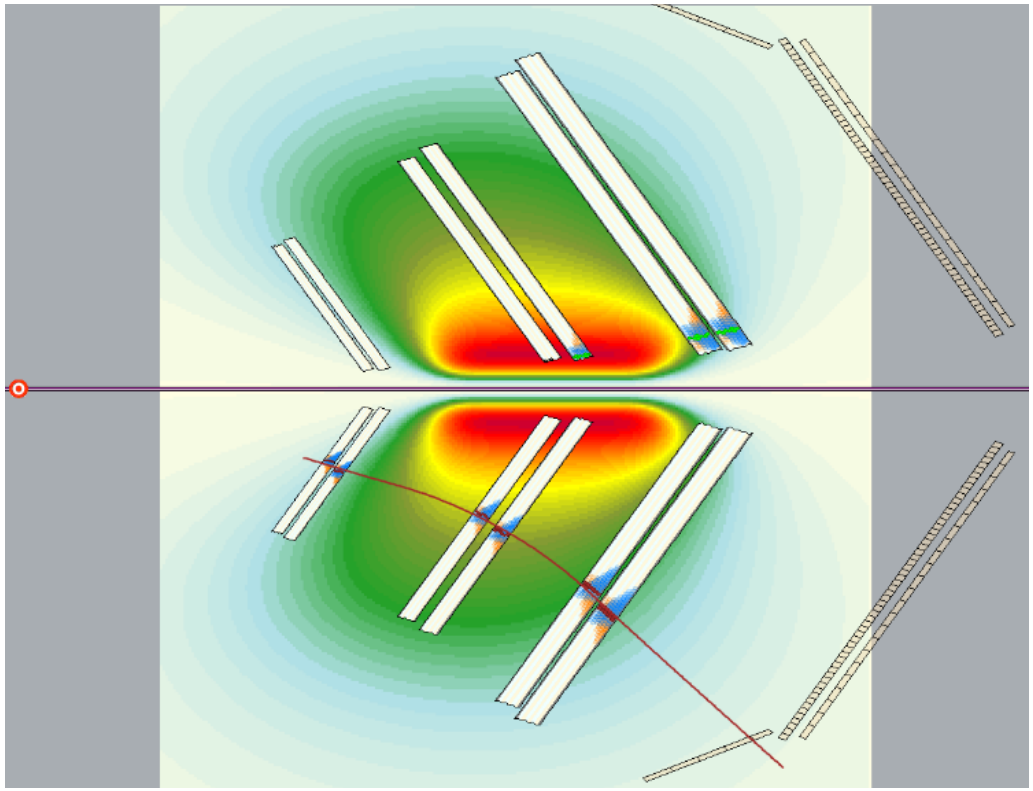
GEMC: geant4 based Monte Carlo Event Simulation

Forms the foundation of the software effort- Reconstruction code development is based on the simulations of GEMC



Components of CLAS 12 Tracking

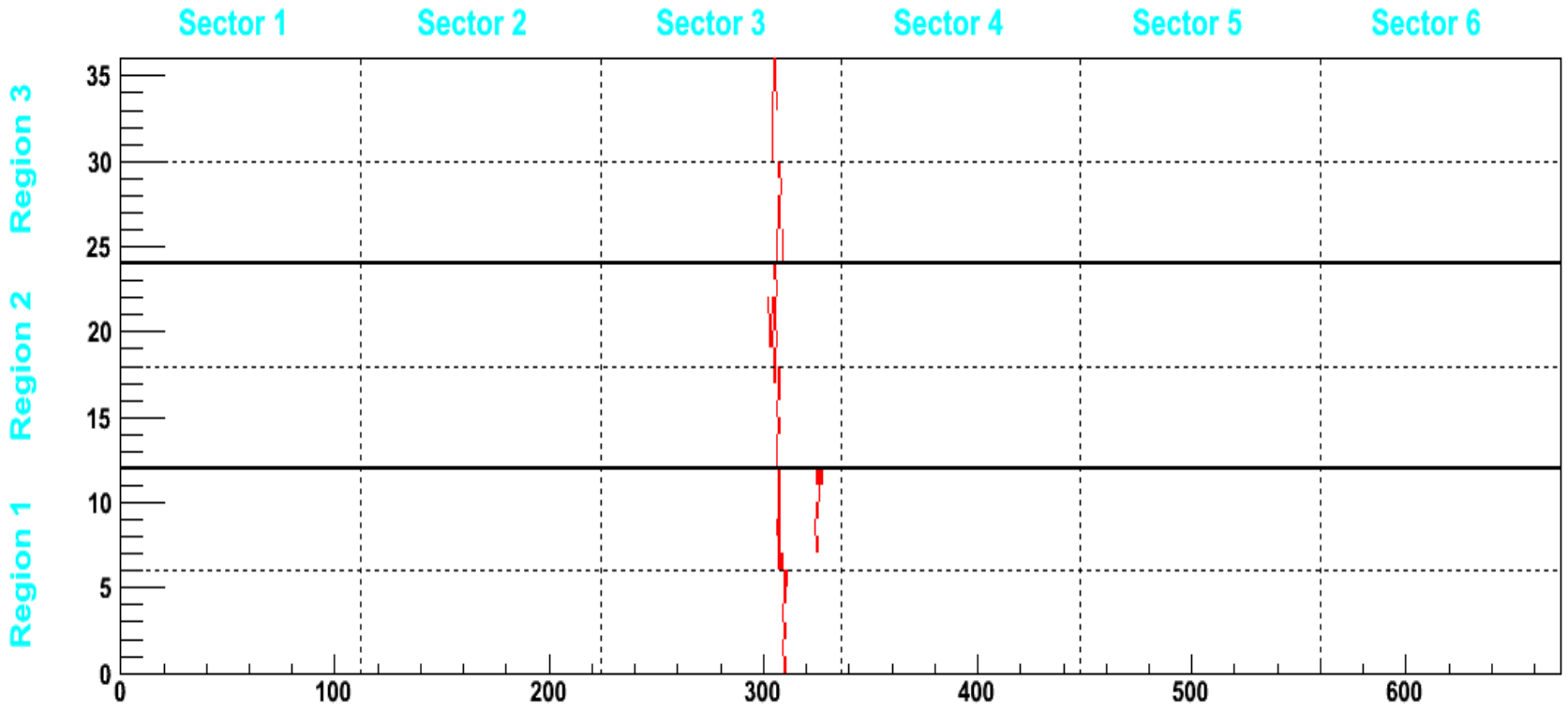
- Forward Tracking
  - ☞ Toroidal Magnetic field bends charged tracks in/out



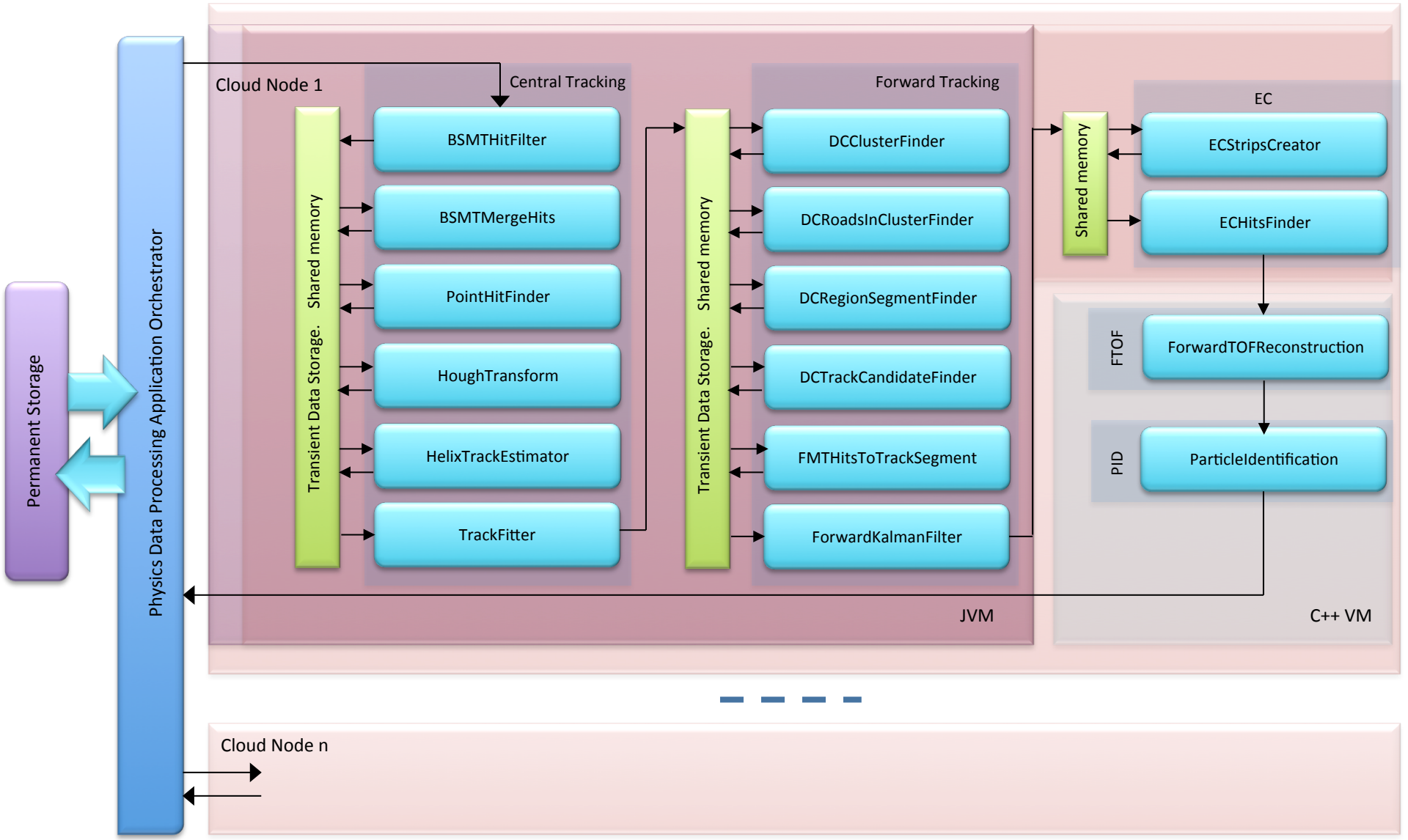
  - – Drift Chambers
  - – Forward Vertex Tracker
  - – Forward Time of Flight detector

Socrat (ROOT)

Starting point (uncorrelated background, just for illustration):



Generation II SOT (OO/SOA)

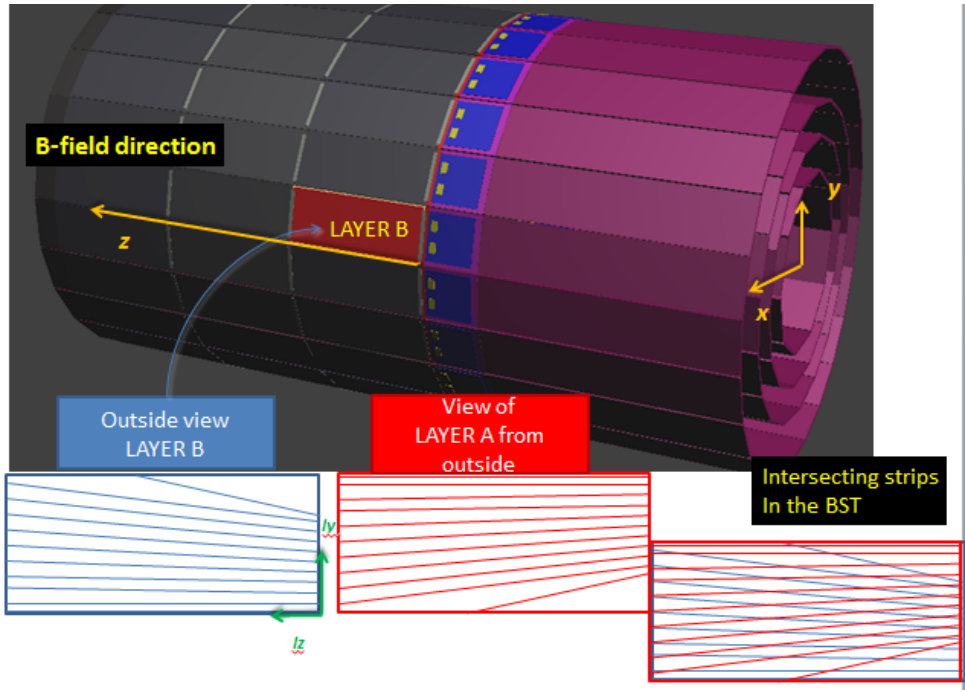


Tracking: Status and Future Plans

➢ Central Tracking

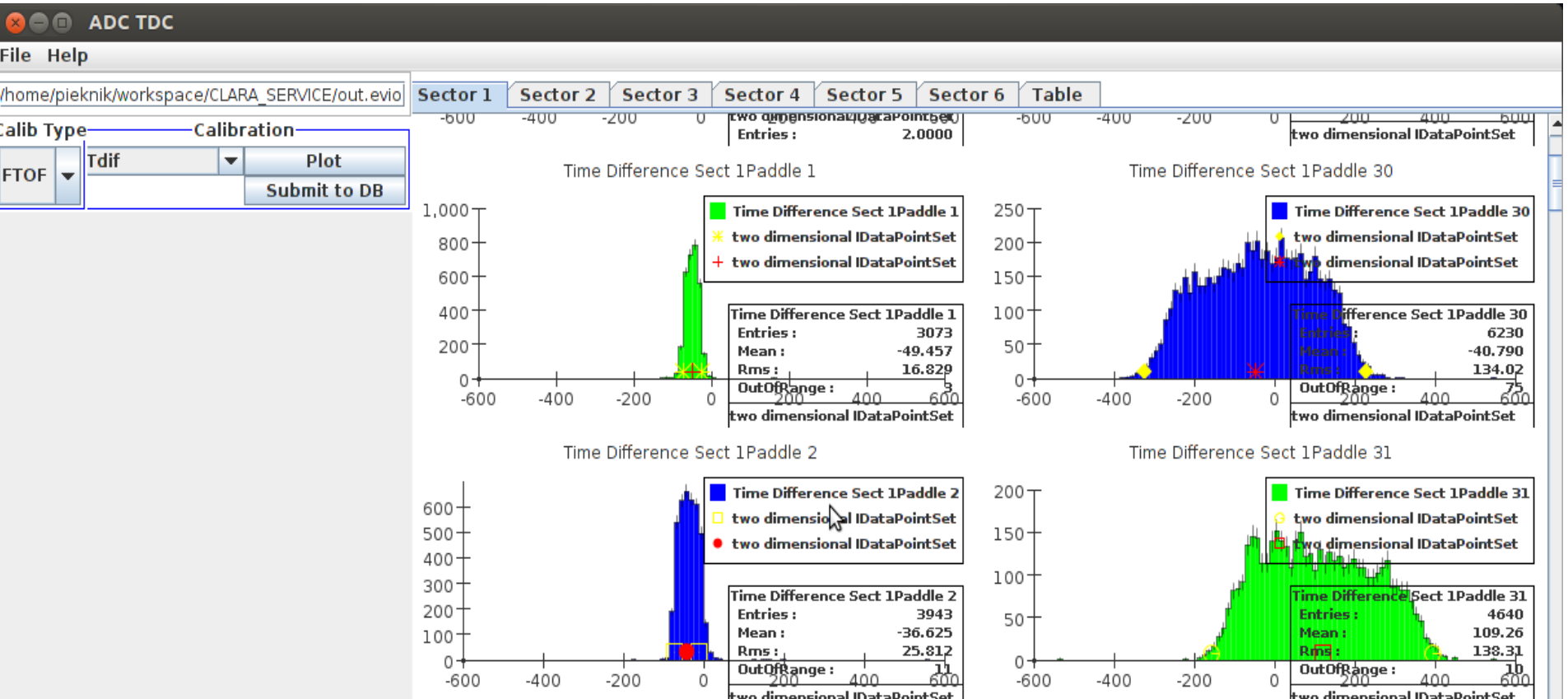
Generation III

- Hit Recognition
  - — Central tracker strip ``intersection'' determination
    - — not on the same plane
    - — i.e. trajectory-dependent
    - — hence use iterative algorithm to improve hit position accuracy based on track's angle of intersection with BST planes

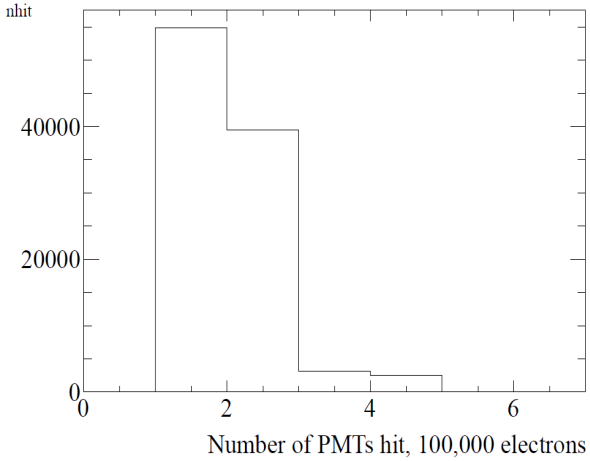


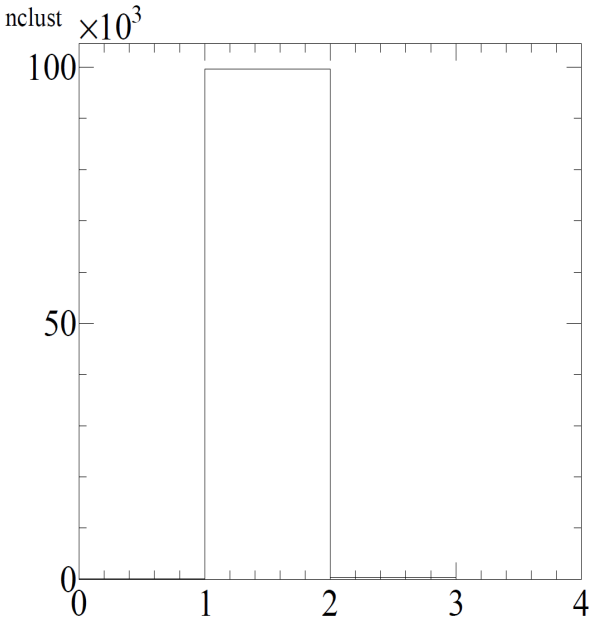
- Pattern Recognition
  - — Hough Transform (✓)
  - — Geometrical linking algorithm, Look-up tables, … (to be implemented and tested)
- Track Fitting
  - — Kalman Filter (✓)
  - —Global Fitting Methods (to be implemented and tested)

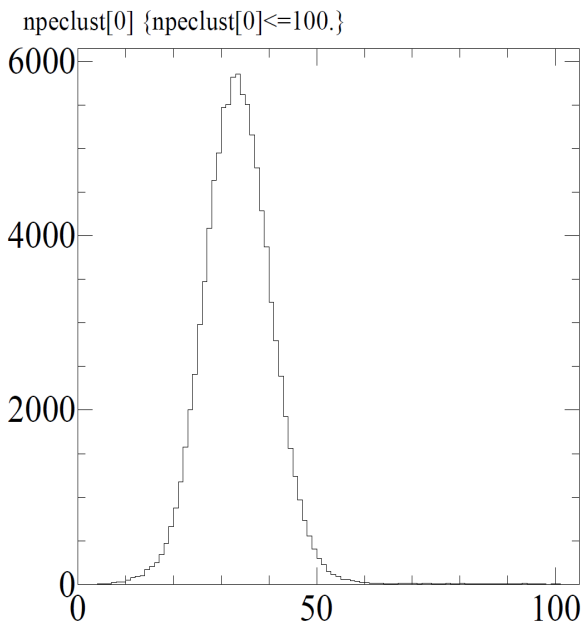
CTOF/FTOF Calibration/Monitoring Suite

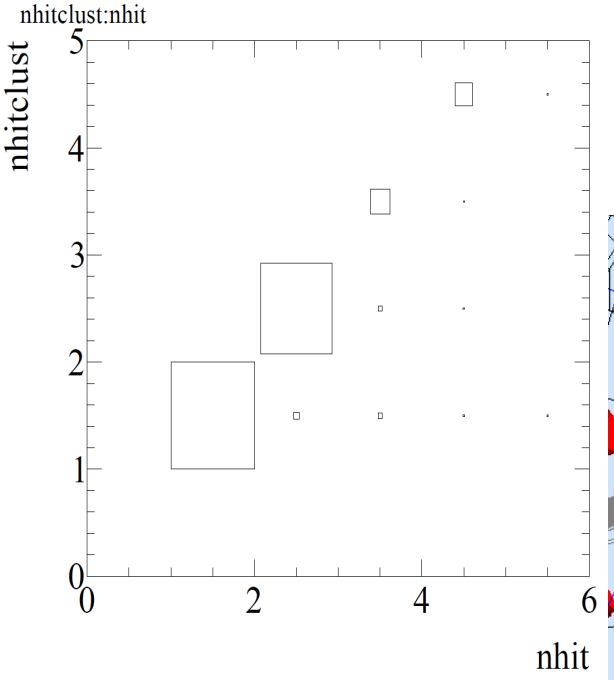


High Threshold Cerenkov Counter



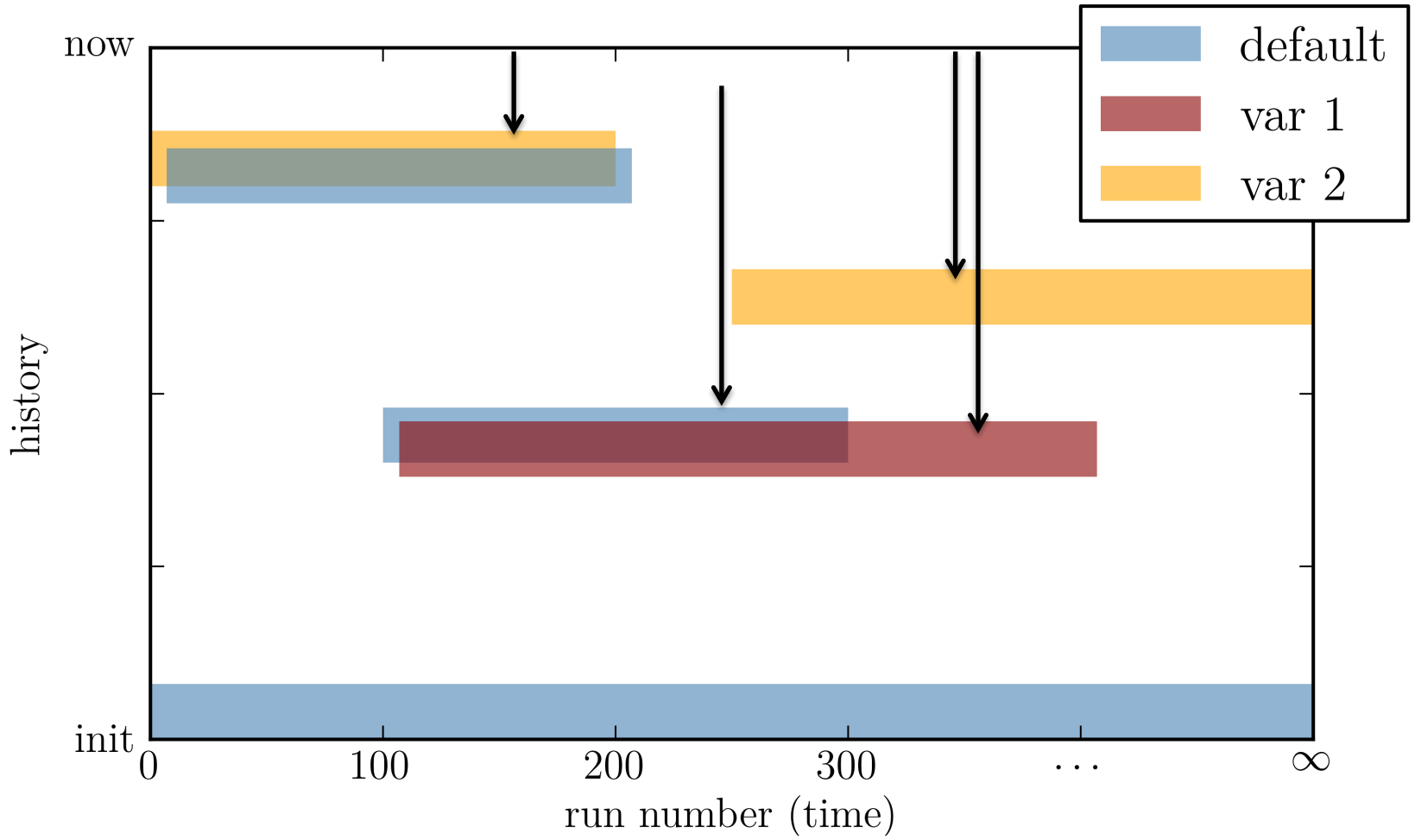




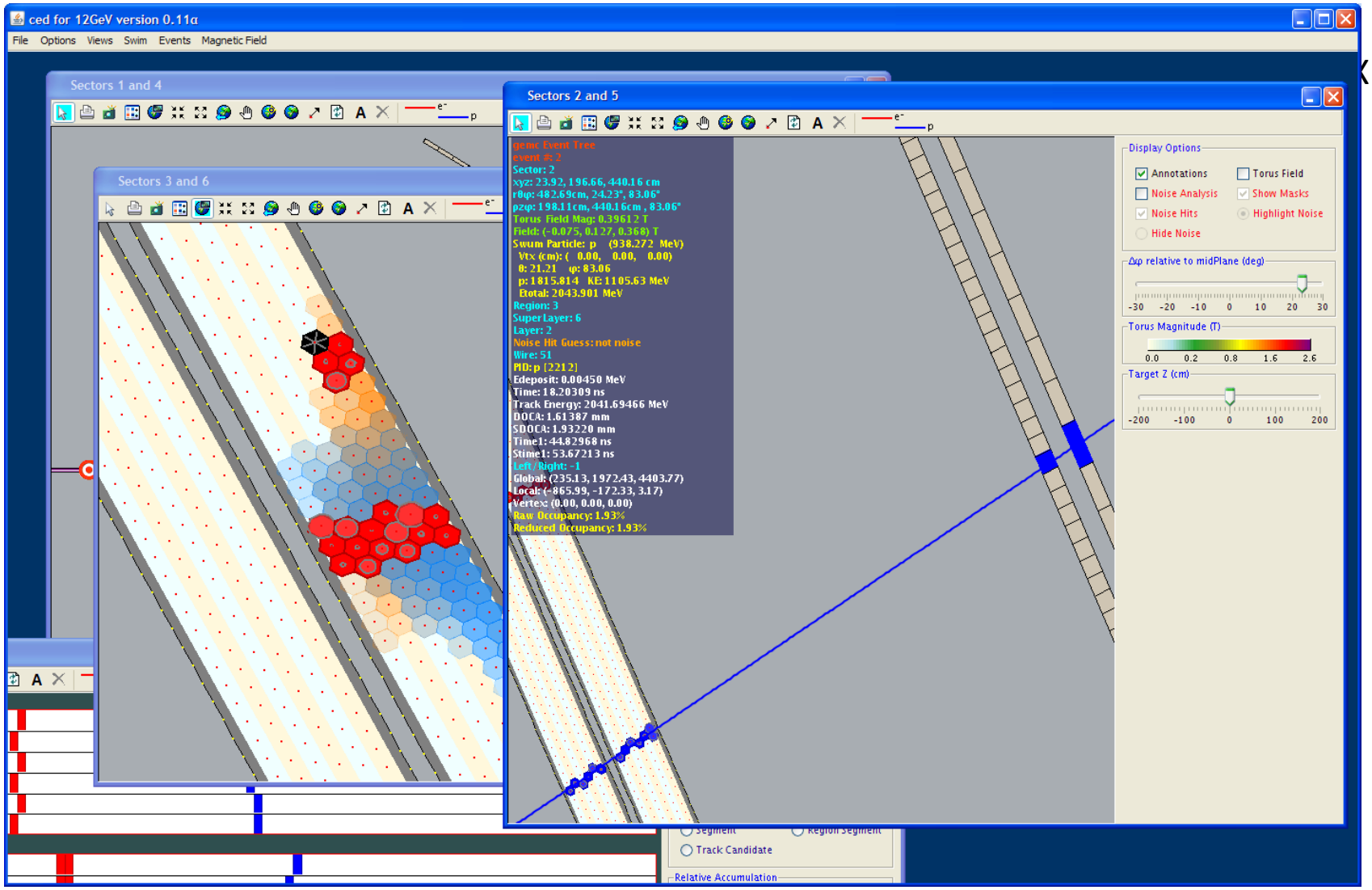


- Group all hits into clusters starting from the hit with maximum amplitude, add nearest neighbors, recursively search all cluster hits for nearest neighbors
- Behavior controlled by hit, maximum, and cluster thresholds and timing cut.
- Algorithm tested with simulated data, works exactly as intended.

Calibration and Conditions Database



Individual Event Display (ced)



Production Clas12 Physics Data Processing

Online calibration
Online reconstruction and data quality checks
High level triggers

Offline Calibration
First pass event reconstruction on JLAB local cloud
DST production at JLAB (Tape/disk)
DST distribution to university clouds

Software Tools

Languages: C++, Java, Python

Code management: Subversion

Code Development:

Standard IDE's: Eclipse and NetBeans

Jlab Tools

EVIO

CCDB: mysql based calibration and conditions (and geometry) database (Shared with GlueX)

bCNU/jevio Event Display
(Shared with GlueX)

HEP/NP Data analysis

Root

CLHEP: C++ tools for HEP

jHepWork/freeHep

COLT: Java implementation of CL

JAIDA

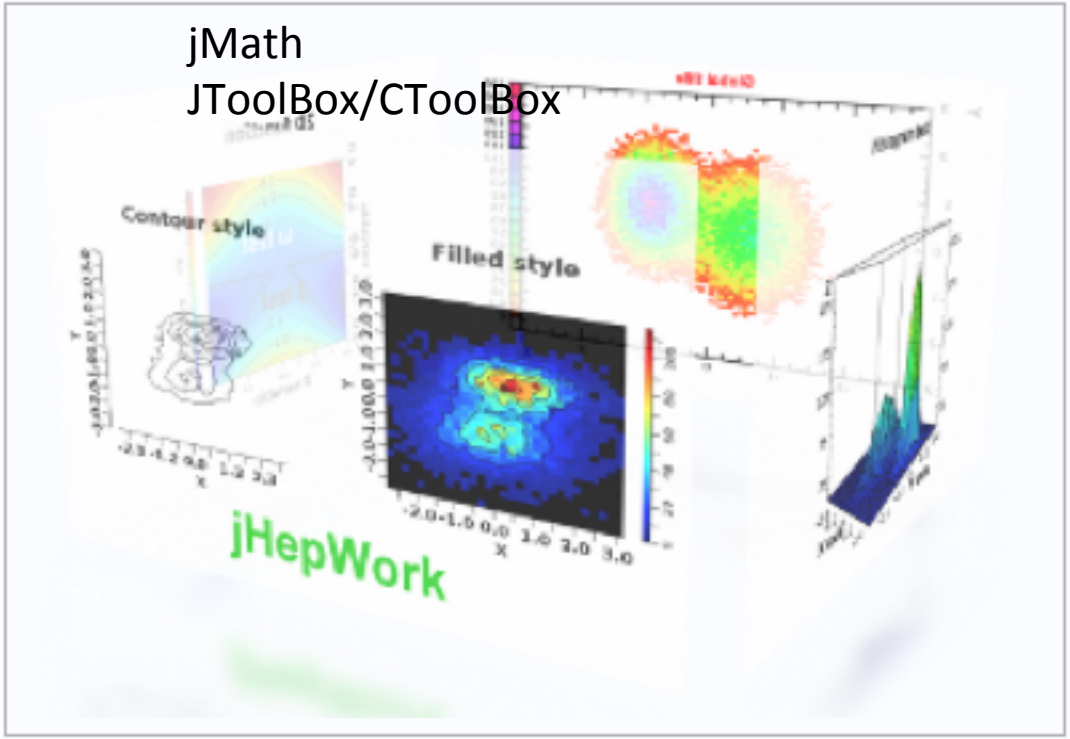
NumPy: Python Mathematical to

SciPy: Python Numerical tools

Geant4: Particle tracking through

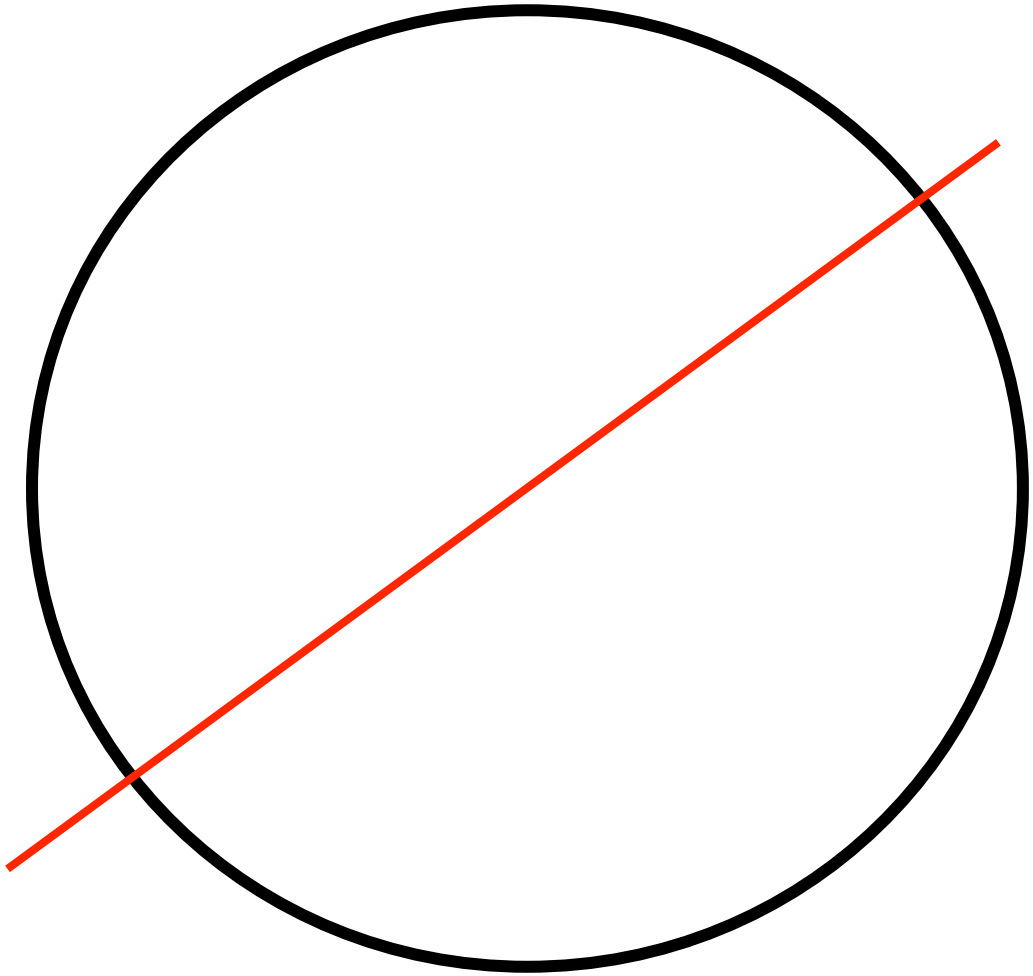
jMath

JToolBox/CToolBox



Summary

- Simulation (GEMC)
- Reconstruction
  - Tracking
  - Calorimetry
  - Cerenkov Counters
  - TOF
- Reconstruction Framework (ClaRa)
- Individual Event Display (ced)
- Code Management (Subversion)
- Software Component Management



Service Oriented Architecture

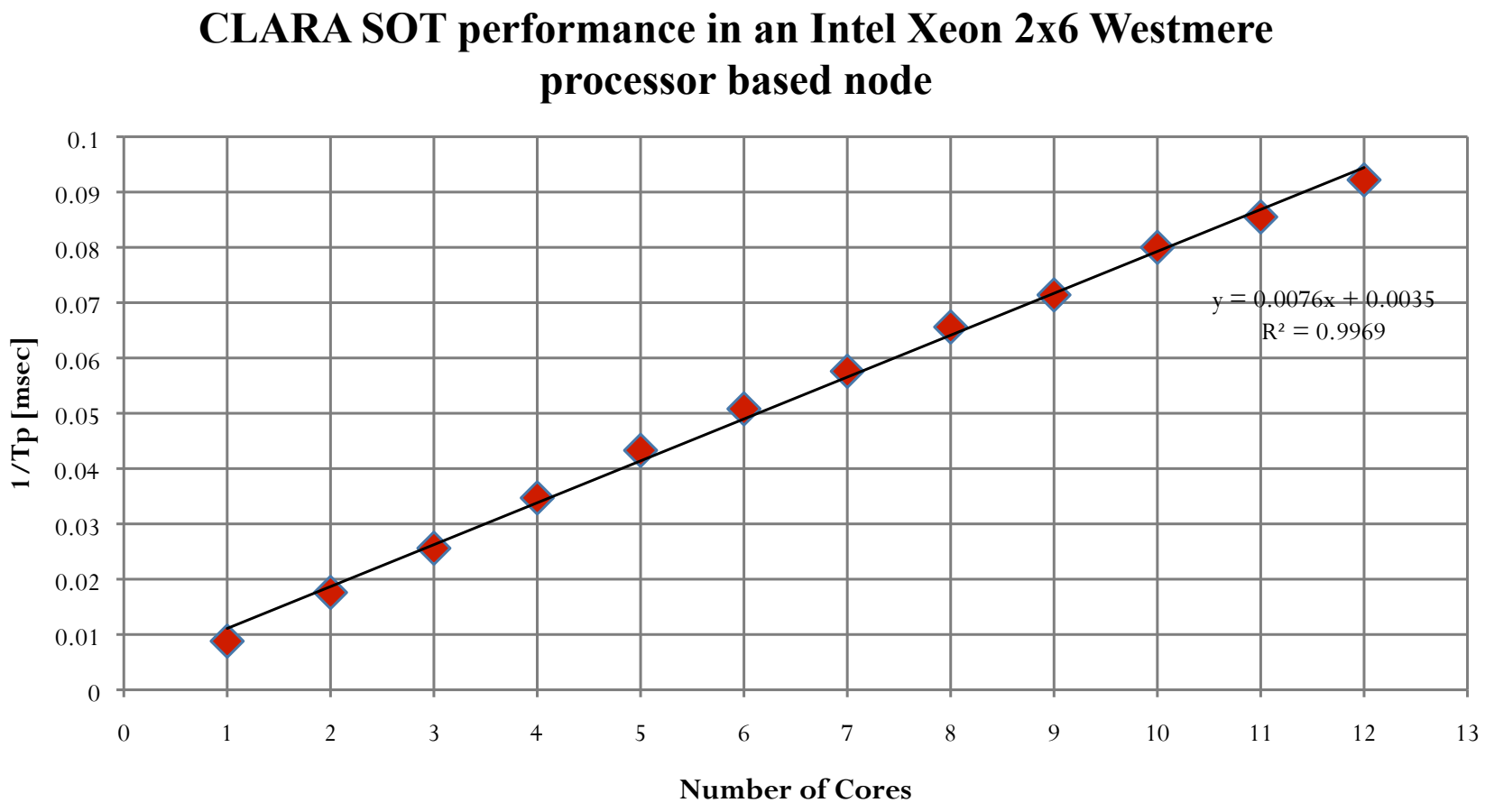
Services are unassociated, *loosely coupled* units of functionality that have no calls to each other embedded in them. Each service implements a single function. Rather than services embedding calls to each other in their source code, they use defined protocols that describe how services pass and parse messages using description metadata.

Services are associated by *orchestration*. In the process of orchestration the developer associates software functionality (the services) in a non-hierarchical arrangement using a software tool that contains a complete list of all available services, their characteristics, and the means to build an application utilizing these sources.

SOA aims to allow users to string together fairly large chunks of functionality to form *ad hoc* applications that are built almost entirely from existing software services. The larger the chunks- the fewer the interface points required. However*, very large chunks of functionality may not prove sufficiently granular for easy reuse.*

Place Holder for Current Performance Plots

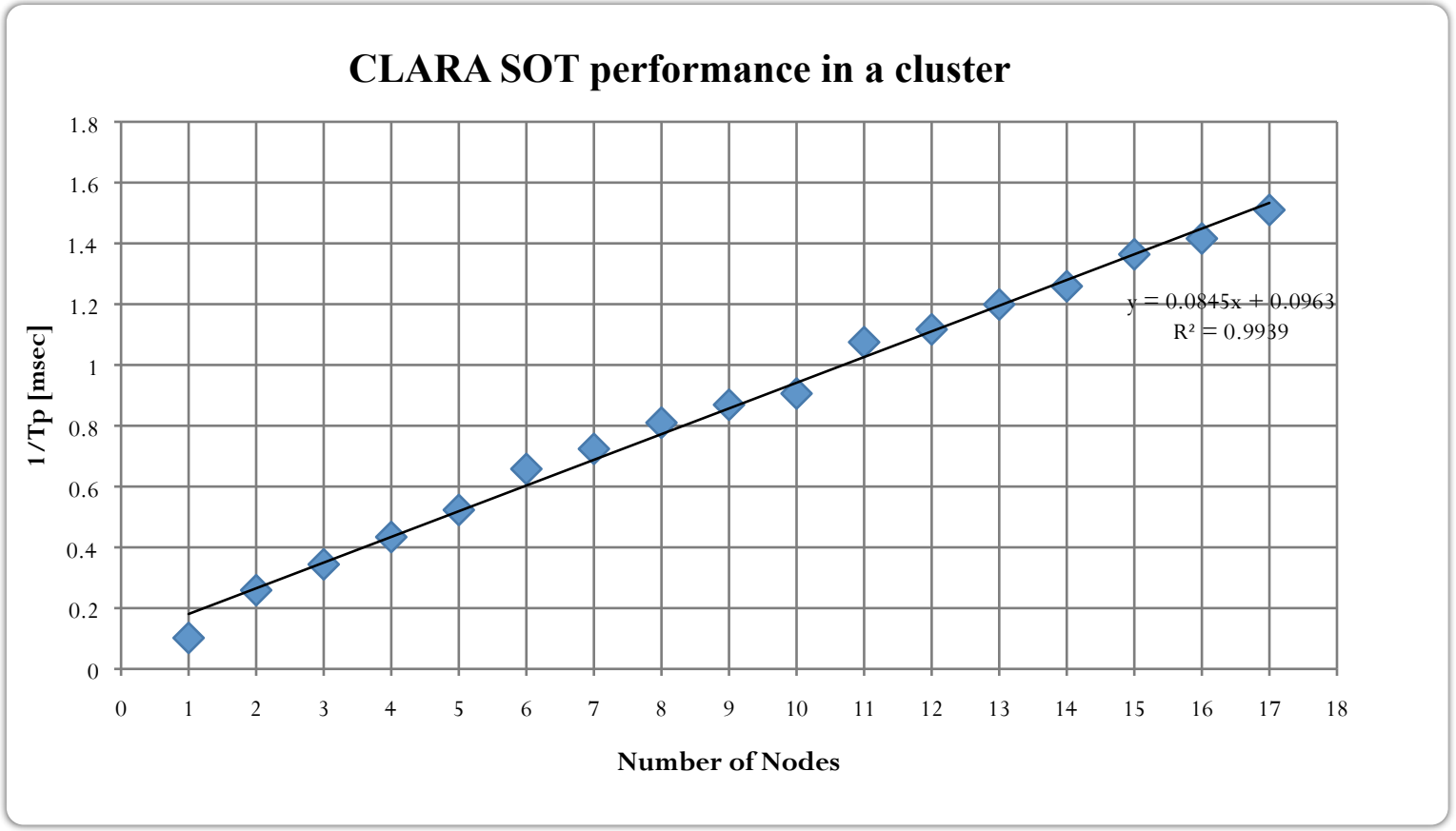
MultiThreading with one 1 Task Service (SOT) running in a single node



Place Holder for Current Performance Plots

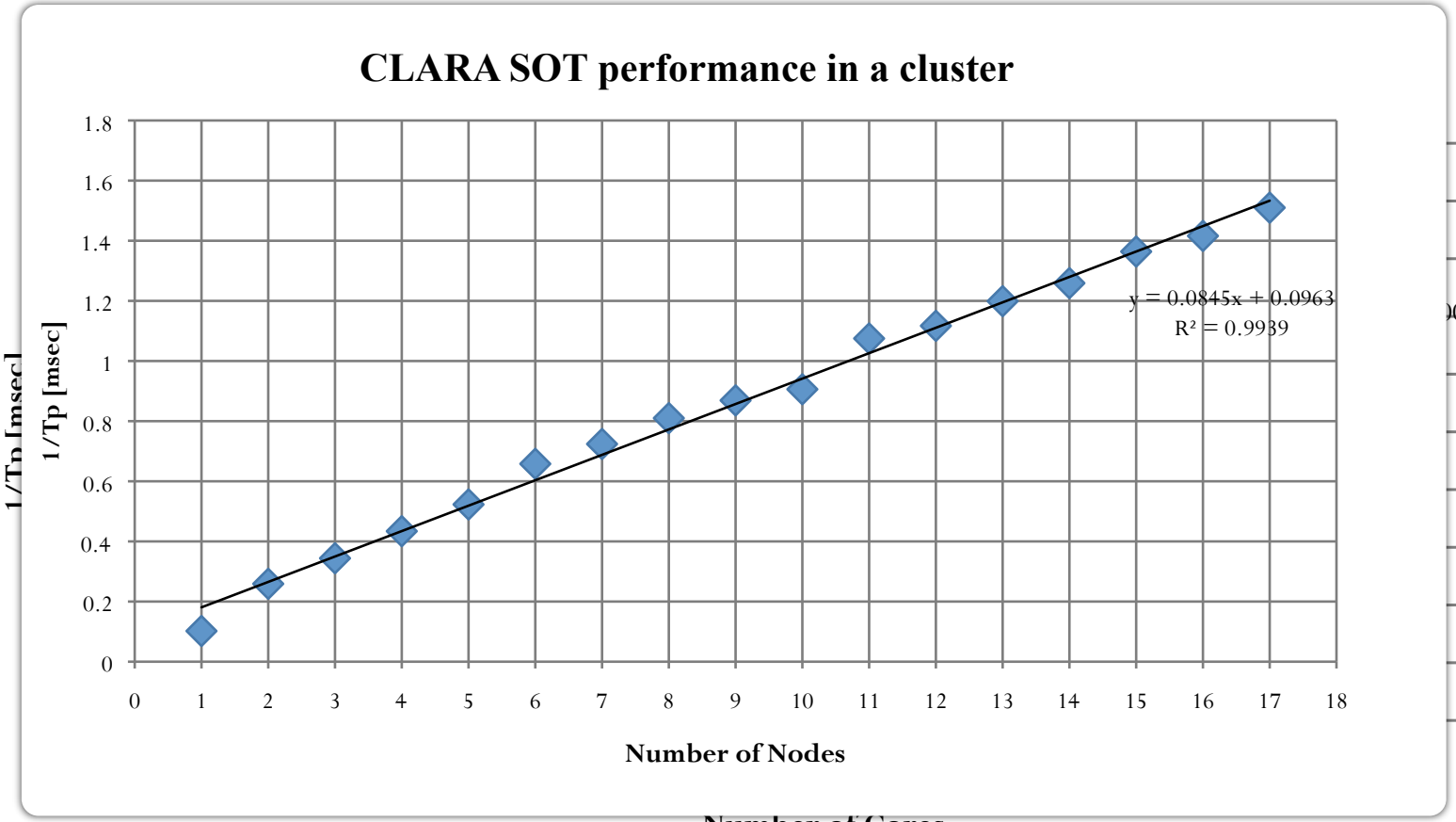
Parallel distributed processing

17 node Xeon 2x6 Westmere CPU



MultiThreading with one 1 Task Service (SOT) running in a single node

Parallel distributed processing 17 node Xeon 2x6 Westmere CPU



0035

13

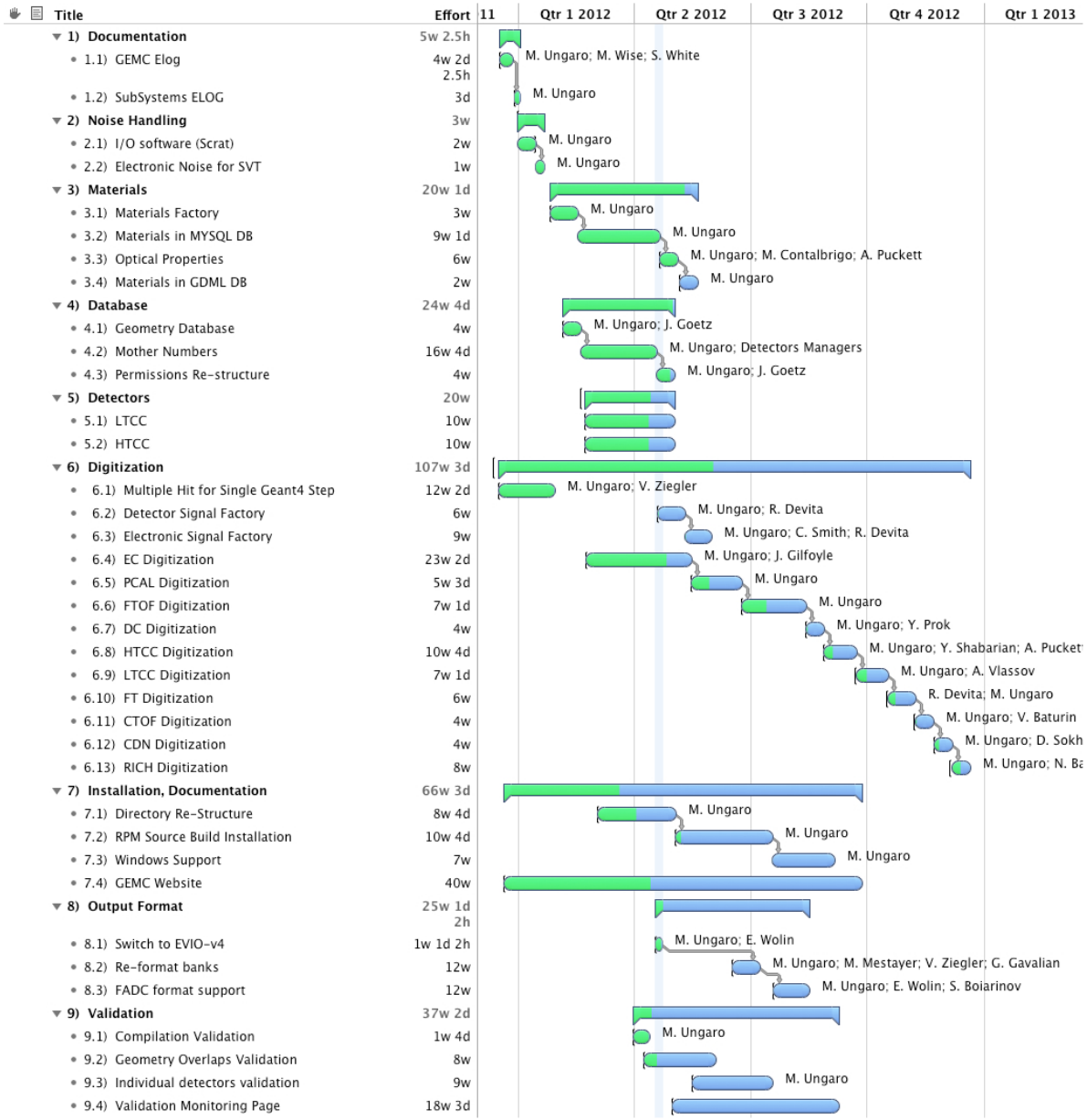
Number of Cores

9 8

Challenges

- Communication latency
  - careful analysis of the criticality of a service, introduce built-in tolerance for variations in network service response times
- Increases author and user pools.
  - Management and administration. Strict service canonization rules
- Workloads of different clients may introduce “pileups” on a single service.
  - Solve by introducing service access policies
- Network security
  - Client authentication and message encryption

GEMC timeline through 2012



Calibration and Commissioning Software