Offline System Overview

Adam Lyon

Muon g-2 Computing Readiness Review

November 7, 2016

Outline

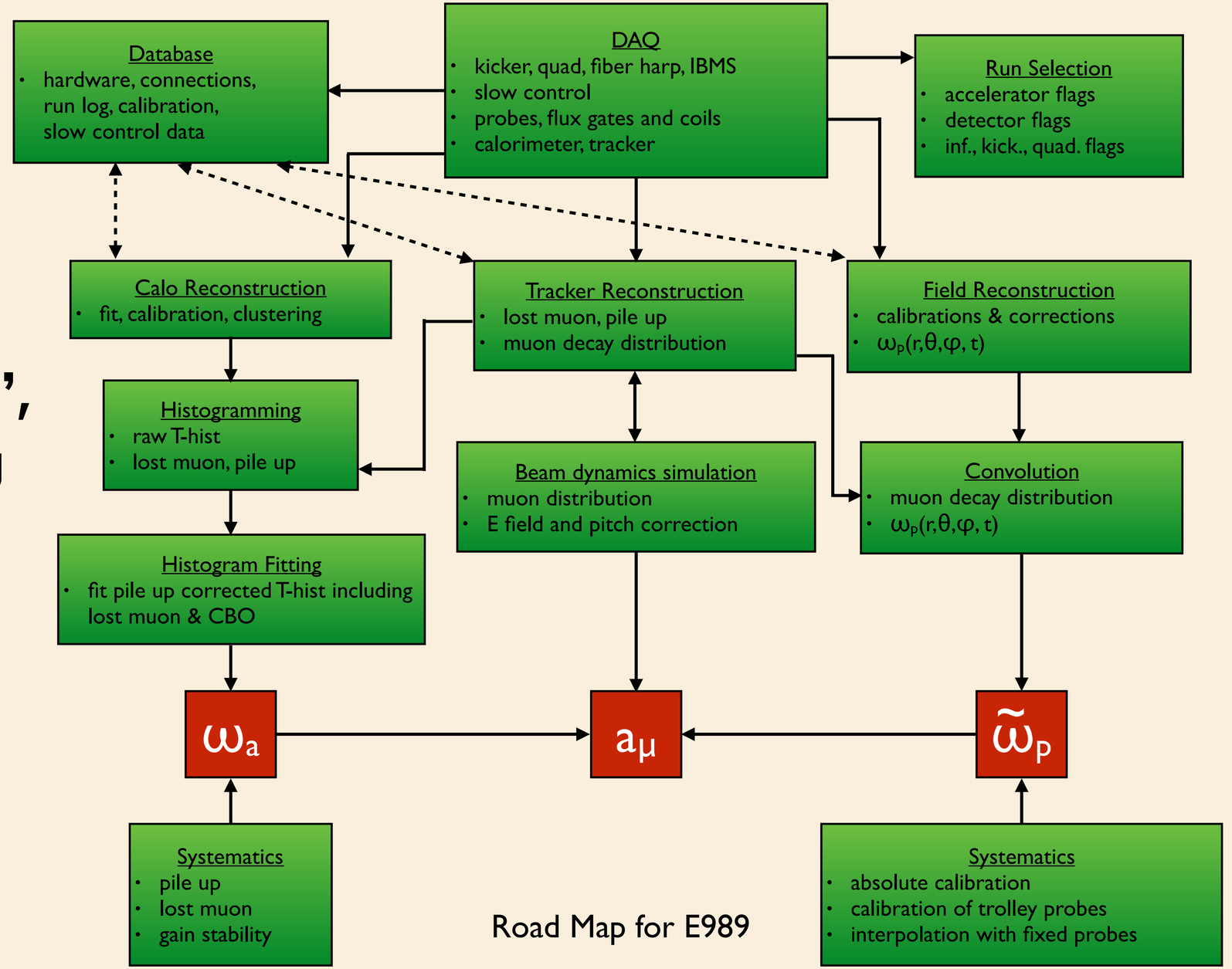
- *art* g-2 framework and offline code structure - our ecosystem
- Data management and production workflow
- Nearline production

- Requirements
- Implementation
- Current status
- Future plans
- Leaving many cool pictures and plots to others

- Is this system ready for data taking?

What we're doing

While we get one "answer", it has many values feeding into it



Road Map for E989

Fermilab

Overall Requirements and Principles

We want to work together!

We need a software system that makes working together easy while maintaining or sanity

What does this mean?

- **Following best coding practices?**
- **Using standard libraries and APIs?**
- **Creating your own libraries for others to use?**
- **Share your code in a repository?**
- **Documenting your code?**
- **Find infrastructure code from somewhere?**

Yes to all the above

Principals

- **Science demands reproducibility.**
  We must have control over our software

- **We want to work together.**
  Share ideas through code

- **We want to do physics, not computing.**
  We wanna make plots! Somehow, that should be easy and sane

- **We want to leverage existing expertise**
  Don’t reinvent the wheel

Principals

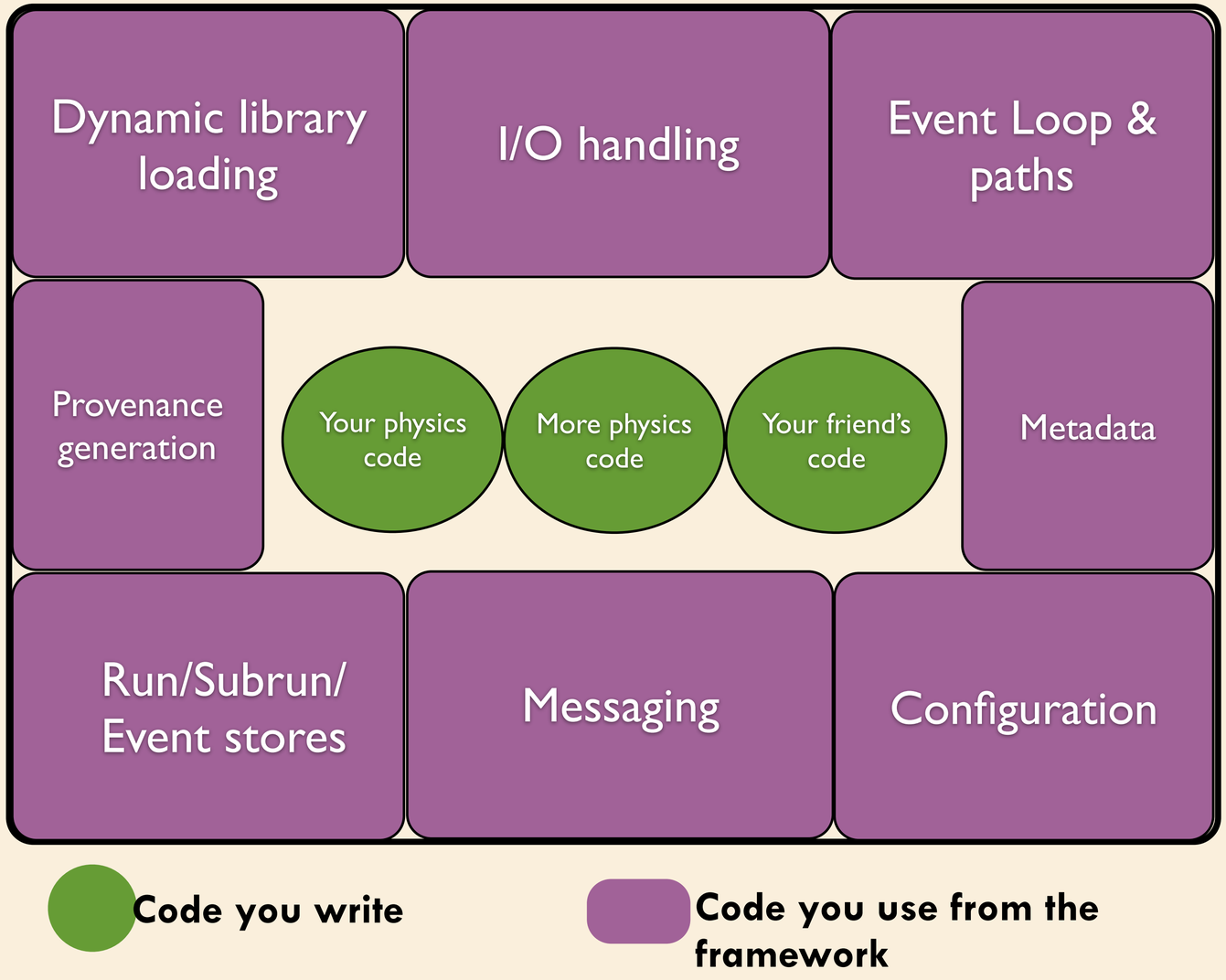
- **Science demands reproducibility.**
  Official results come from version controlled software

- **We want to work together.**
  Code repositories; modular frameworks

- **We want to do physics, not computing.**
  Infrastructure in a framework + an easy build system

- **We want to leverage existing expertise**
  Join a “software community”

The *art* framework



The framework handles the parts of processing that you don't care about and just want to work

Let's you concentrate on the parts you do care about – the physics

Importantly, the framework centers the "ecosystem"

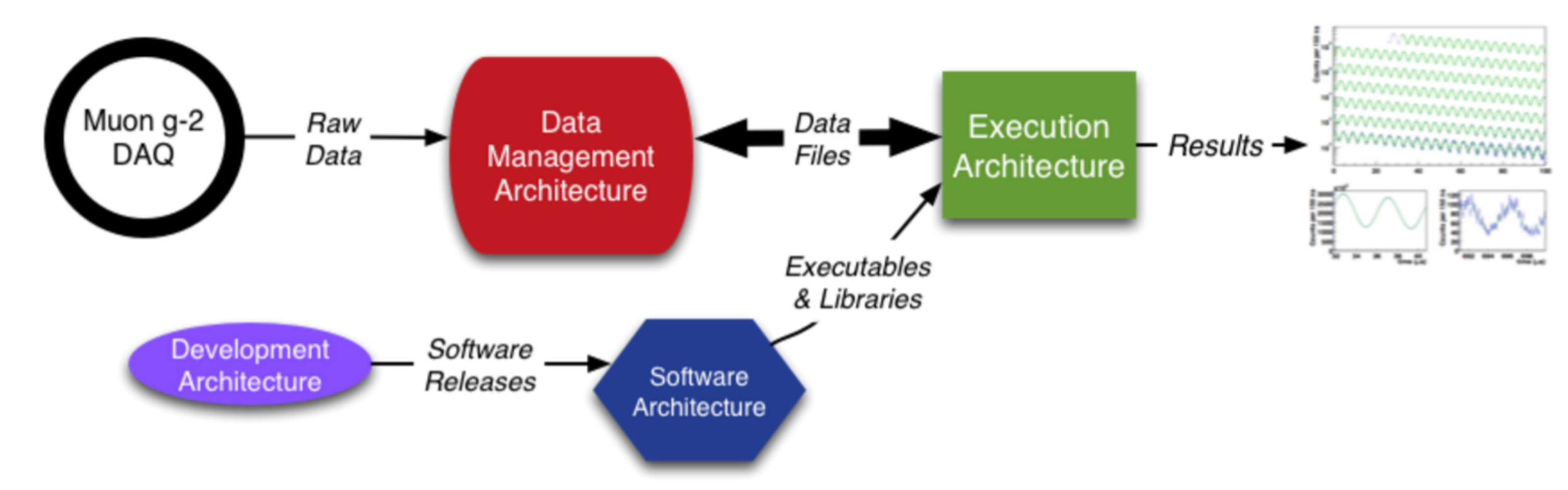
Our Ecosystem

- Source code version control (git)
- Source code repository (redmine, github)
- Build system (cmake/cetbuildtools/mrb, spack)
- Documentation (Redmine Wiki, PDF, github)
- Development Environment (Editors, XCode, CLion)
- Release & dependency system (UPS, spackdev)
- Build infrastructure (Jenkins, CI)
- Distribution system (SciSoft, CVMFS)
- Software framework (*art*)
- Simulation (Geant4, CADMesh, *artg4*)
- Data management (FIFE/SAM, xrootd)
- Job submission (FIFE/jobsub, POMS)
- Databases
- Interactive data analysis (Root, R/Python with *gallery*)
- 3D Visualization (ParaView, *artvtk*, *gm2vtk*)

- Interactive systems (gm2gpvm, Home Institutions, Laptops)
- Guest systems (Virtualbox/Vagrant, Docker)
- Execution sites (FermiGrid, OSG, HEPCloud, HPC)

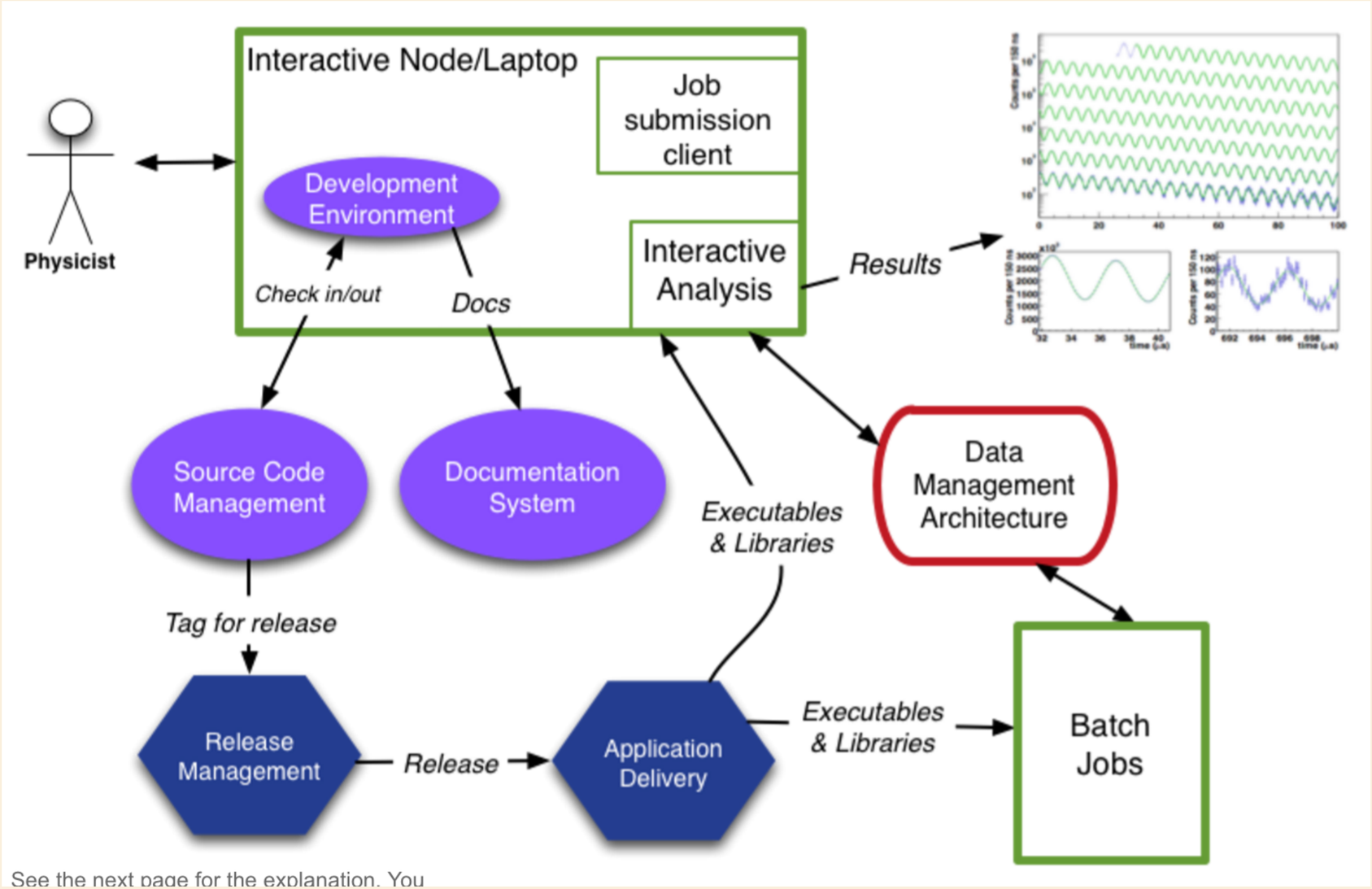
How these fit together - architectures

Architecture: The art of determining the needs of the user and designing to meet those needs as effectively as possible within the constraints of economics and technology



Fermilab

Software/Development Architecture

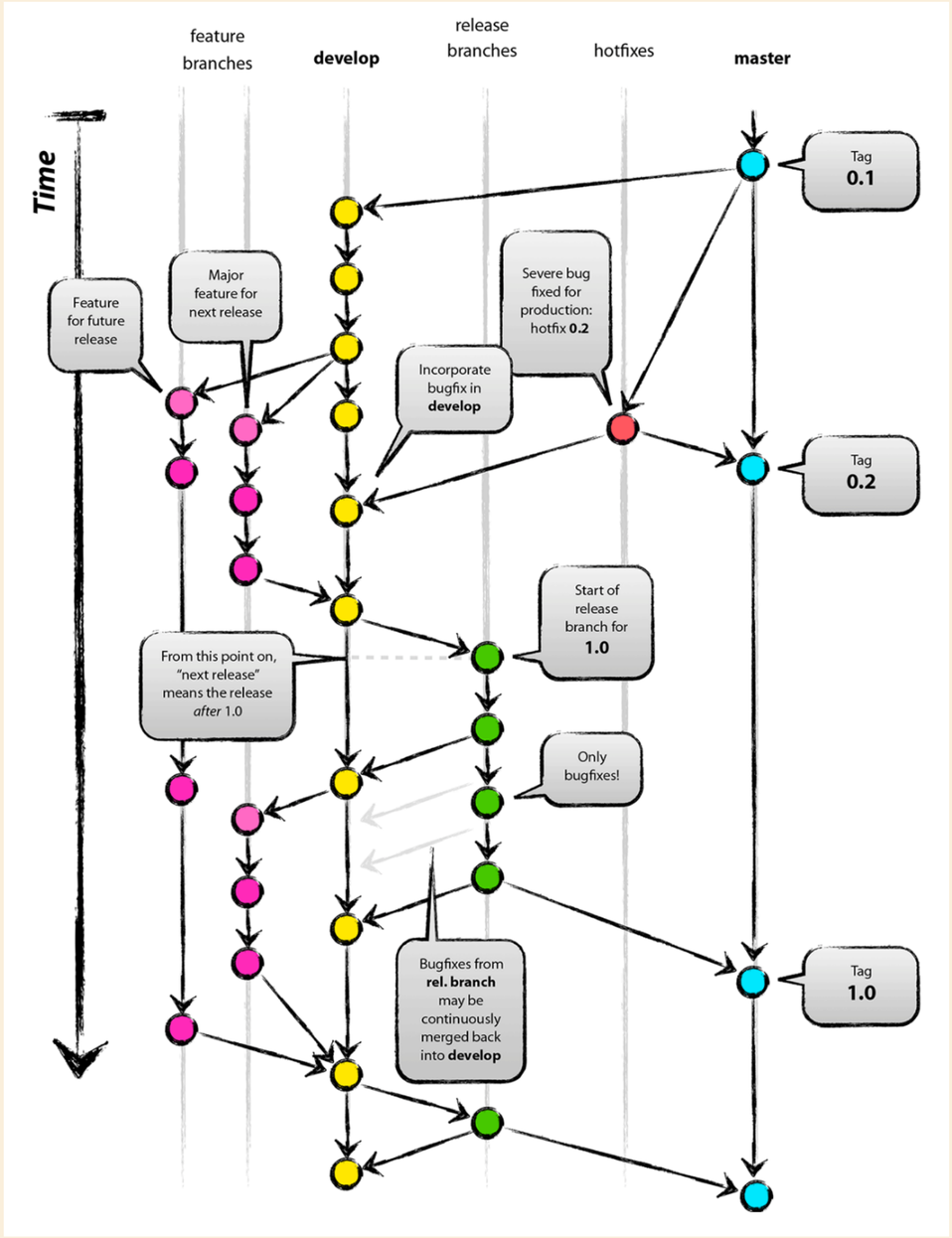


See the next page for the explanation. You

Fermilab

Source code version control

- We were one of the first experiments at Fermilab to use *git* [Because the *art* team used it]
- We use the *gitflow* paradigm and tool
- Hosted by Redmine
- Status: Extremely successful



Fermilab

Source code version control

- Challenges:
  - Disposing of old feature branches
  - Dealing with many multiple repositories

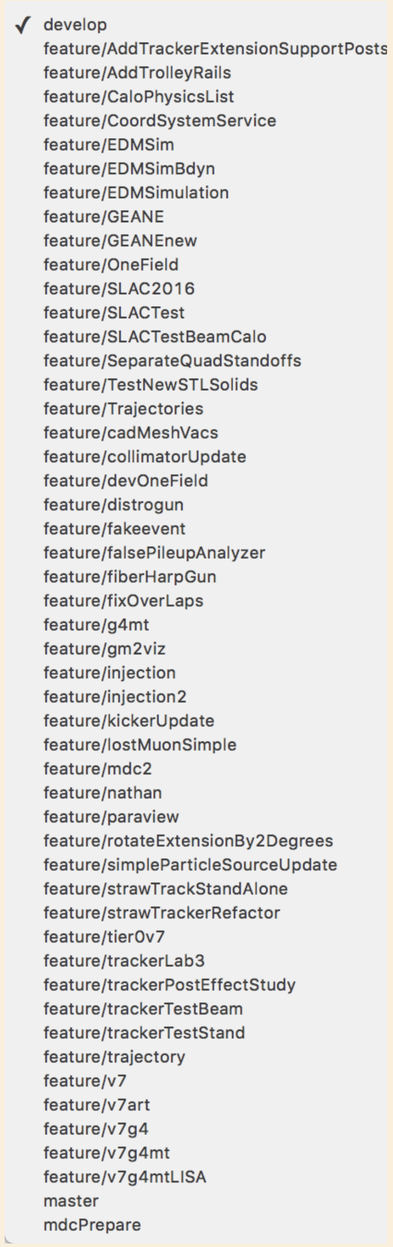
Main offline repositories:

gm2geom, gm2dataproducts, gm2ringsim, gm2midastoart, gm2calo, gm2tracker, gm2unpackers, gm2util, gm2analyses

Hard to know what code is where

Hard to search

Hard to version (in the end it's the version of the whole that matters)



Fermilab

Proposed improvement

- Move to single repository a la CMSSW on github (Redmine as backup)

- Advantages:
  - Fixes searching, versioning, what code is where problems
  - Leverage work from CMS (took them two years to complete their system)
  - Move to github workflow (pull requests, approvals)
  - Embedded documentation possible (huge improvement over Redmine)

- Difficulties:
  - Have to implement git sparse checkout
  - Have to implement dependency checking
  - Migration

- Need time and people to do this (there's a lot of interest)

Build/Release system

- Early adopters of cmake/cetbuildtools (developed and used by *art* team)
- Externals and all of our packages are managed by relocatable UPS
- Externals (gcc, Geant4, Root, …) come from *art* team except for a few that I manage (paraviewvtk, cadmesh — in SciSoft)

- mrb "multi-repository-build" is a command line layer on top of cetbuildtools [started as "gm2d"; adopted by LArSoft, maintained by Lynn]
- mrb newDev, mrb getgit, mrb setenv, mrb build, mrb test, …

- UPS makes coexisting flavors and versions very easy
- Very successful!

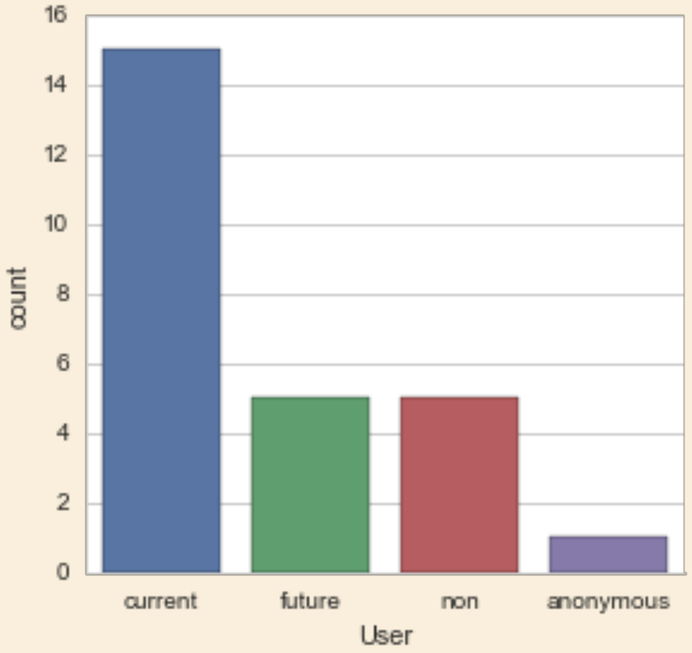
Build/Release system

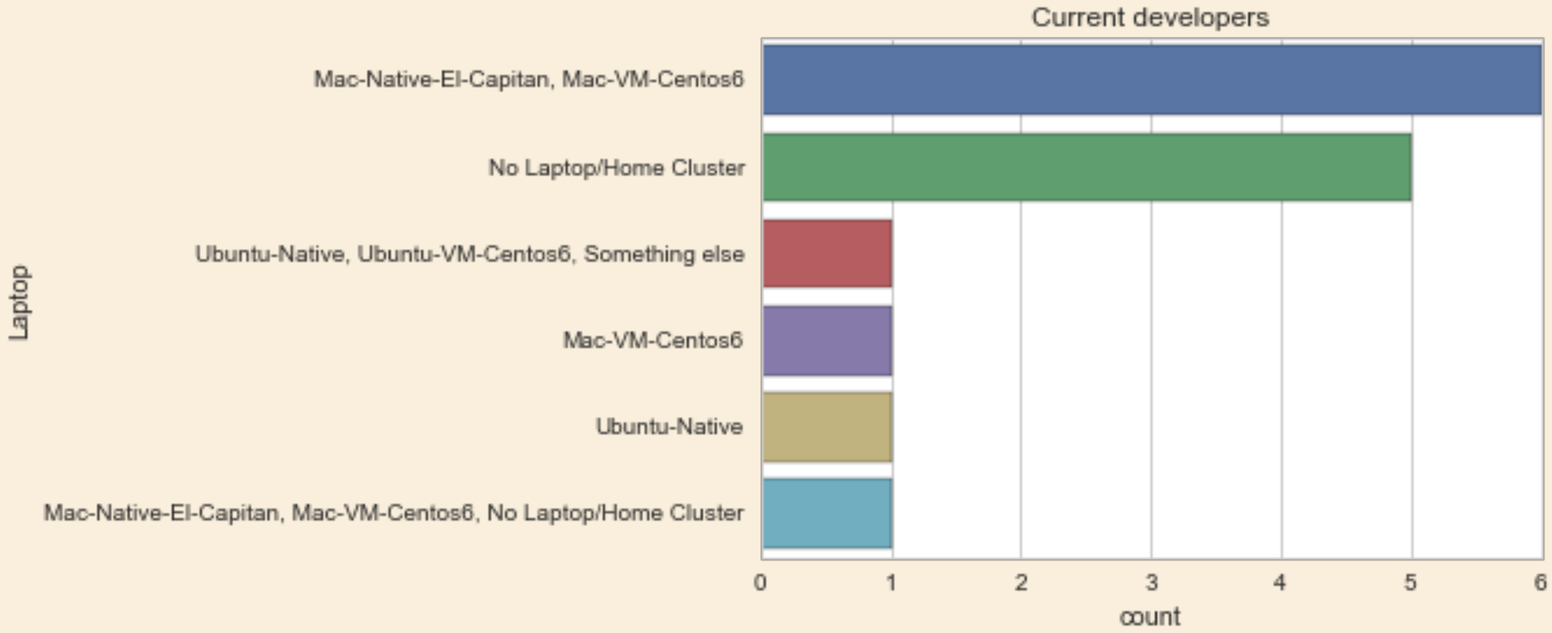
- Challenges:
  - Cetbuildtools is a very heavy-handed system; hard to extend; hard to get around limitations
  - Does not use cmake “in the cmake way” [to be fair, cetbuildtools predates many cmake improvements]
  - Ups-ifying external packages is difficult
  - Small community uses cetbuildtools and ups

- Waiting to see where “Spack” goes
  - Appears to be much more flexible, easier to maintain
  - Larger community
  - Really need “spackdev” too (equivalent to mrb)

Development environment

- Survey taken in early 2016





Are you a g-2 developer?

What kind of laptop do you use?

Fermilab

Substantial use of laptops

- Native Yosemite, Native El-Capitan, SLF6 via Vagrant/Virtualbox, SLF6/7 via Docker

- Vagrant/Virtualbox (makes Windows laptops usable)
  https://github.com/lyon-fnal/centos-gm2-dev

- Docker
  https://github.com/lyon-fnal/docker-gm2

- Emacs, Xcode, CLion — can integrate with mrb

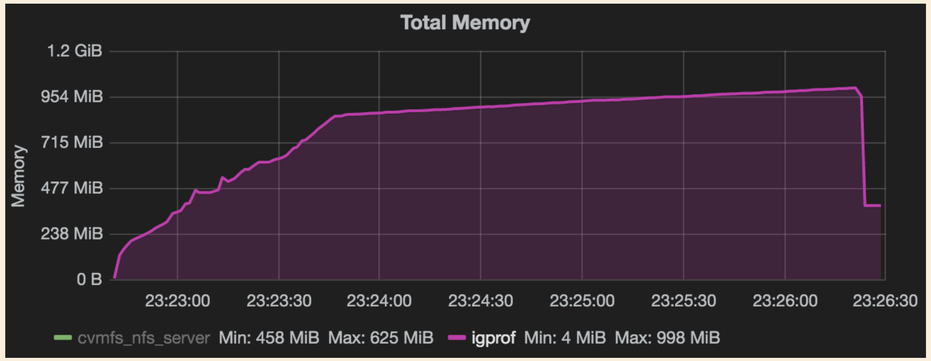
- gm2gpvmXX machines (all SLF6) are useful, but a little slow

Testing

- We have integration tests
  - Our builds run tests to exercise parts of the simulation
- Unit tests are virtually non-existent
  - We don't know how to write these, especially in a Geant4 environment
  - We don't know how to get this expertise
  - Had an IMSA student try to write some tests — not very successful
  - We should think about this for algorithm writing — but no expertise
- We rely on validation tests
  - This seems to be sufficient for now
  - [See Verification Package Breakout / Renee]

Optimization

- We use *art* time/memory services and igprof
- Docker container monitoring helps



```
  // make lookup hit
  gm2ringsim::LookupHit* lh = new gm2ringsim::LookupHit(lookupTable, aSecondaryTime, depth, local_cosTheta, copyNum, rand1, rand2);

  // only save lookup hit if this photon is detected
  if ( lh->detected ) gm2ringsim::LookupHitStorer::getInstance().addHitToVector(lh);
}
```

```
  // make lookup hit
  auto lh = std::make_unique<gm2ringsim::LookupHit>(lookupTable, aSecondaryTime, depth, local_cosTheta, copyNum, rand1, rand2);

  // only save lookup hit if this photon is detected
  if ( lh->detected ) gm2ringsim::LookupHitStorer::getInstance().addHitToVector(lh.release());

}
```


Fermilab

Build Infrastructure

- Jenkins is used to make released code
  - SLF6, SLF7, Mac Yosemite, Mac El Capitan
  - gm2 “superbuild”
  - May just release SLF6 for the future

- Jenkins is awesome, except…
  - The mac mini machines are severely underpowered
  - CVMFS on the macs has severe problems, I think due to load; I gave up on it
  - The whole system is poorly monitored - build nodes go down without notice

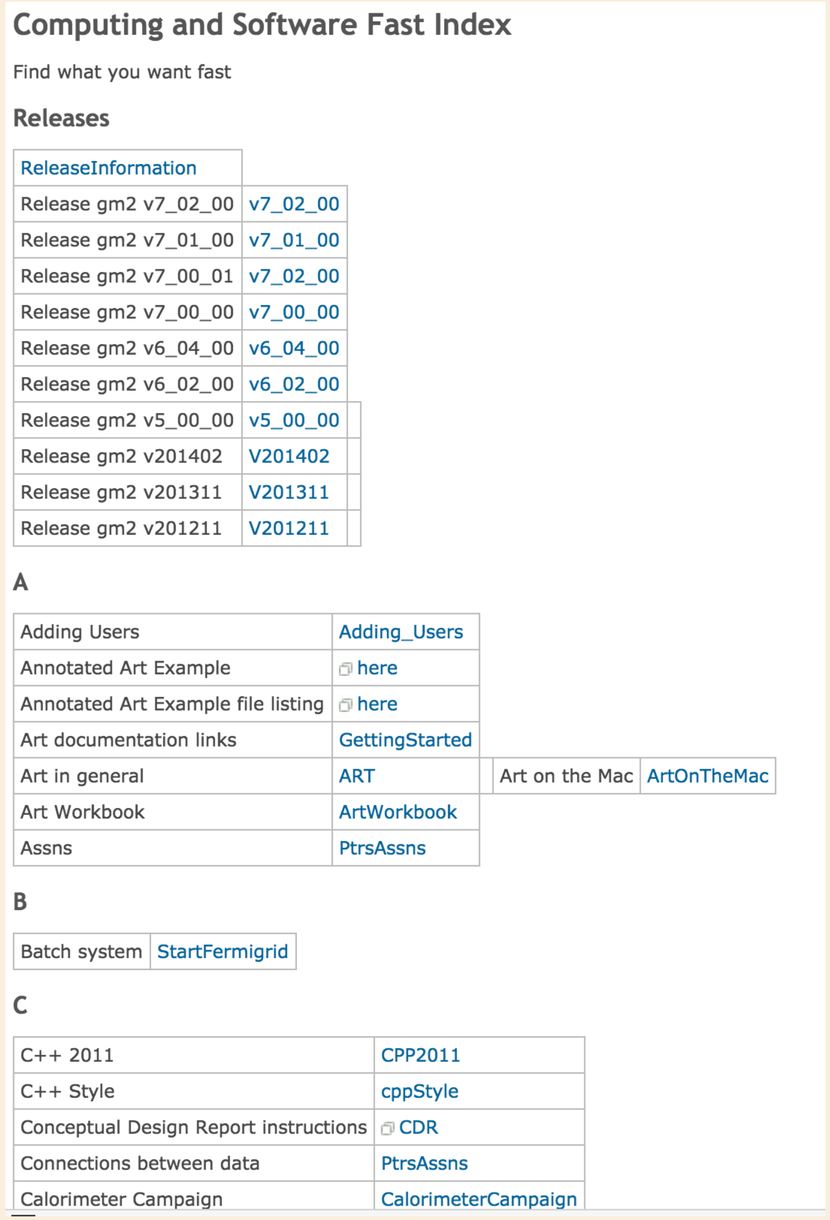
- We need to investigate the Continuous Integration system

Documentation

- We are still looking for the right documentation solution

- Redmine Wiki
  - Easy to write, Hard to search, a pain to maintain, Can't link to code versions, Can't read offline, Hard to index, very far from code, ugly

- PDF Manual
  - See link
  - Easy to write [mixture of LaTeX, Markdown, and generated], easy to search, can link to code versions, can read offline, a pain to maintain, Easy to index, closer to code (in git), less ugly



Computing and Software Fast Index

Find what you want fast

Releases

ReleaseInformation	
Release gm2 v7_02_00	v7_02_00
Release gm2 v7_01_00	v7_01_00
Release gm2 v7_00_01	v7_02_00
Release gm2 v7_00_00	v7_00_00
Release gm2 v6_04_00	v6_04_00
Release gm2 v6_02_00	v6_02_00
Release gm2 v5_00_00	v5_00_00
Release gm2 v201402	V201402
Release gm2 v201311	V201311
Release gm2 v201211	V201211

A

Adding Users	Adding_Users		
Annotated Art Example	here		
Annotated Art Example file listing	here		
Art documentation links	GettingStarted		
Art in general	ART	Art on the Mac	ArtOnTheMac
Art Workbook	ArtWorkbook		
Assns	PtrsAssns		

B

Batch system	StartFermigrid

C

C++ 2011	CPP2011
C++ Style	cppStyle
Conceptual Design Report instructions	CDR
Connections between data	PtrsAssns
Calorimeter Campaign	CalorimeterCampaign

Documentation

- Two possible directions for improvements:

- Doxygen
  - Easy to write, but…
  - Hard to write comprehensive documentation in Doxygen
  - Often just looking at the code is clearer

- MiniBooNE style README files (Chris Polly)
  - Every directory in repository has a comprehensive README describing contents
  - Release not allowed until documentation is in place
  - Easy to version, easy to search [git grep or with github], easy to link to versions, easier to maintain, documentation very near code, can read offline for checked out code, can be pretty with Markdown
  - Using Github and automatic Markdown rendering would make this really nice
  - But would still need a place to document non-code (e.g. submitting jobs, etc)

Fermilab

Training

- How to get non-experts up to speed?
- Periodic workshops, usually at collaboration meetings
- Often refer people to the *art* class materials from summer 2015 and workshop from summer 2016
- … and the *art* workbook
- The Offline software Manual is helpful

Official Releases

- Philosophy:
  - A “base release” vX_00_00 is only art and externals. No g-2 code. With this release, you can build all of the g-2 code yourself
  - A “point release” vX_YY_ZZ is layered over a base release with specific versions of g-2 packages. v7_02_00

- A release “system” is evolving
  - Pieces are automated with Jenkins (suffers when Jenkins has problems)
  - But it is time-consuming — we need a release manager

- In Practice:
  - We don’t make releases as often as we should
  - We often go a year without upgrading art (though this hasn’t been a huge problem)
  - We need a release manager — I do this now and can’t always give it full attention

Release distribution

- We manage with relocatable UPS
  - For release management, relocatable UPS works extremely well [but has other problems]
  - All flavors and versions can live together in the same “area” - very convenient
  - Spack could eventually replace UPS

- CVMFS is awesome!
  - This is our standard (only) way to distribute executable code
  - gm2.opensciencegrid.org
  - Used on laptops and home institutions
  - Hosts g-2 code and externals

- SciSoft
  - Used for the source of externals — works well

- No plans to change — Spack may replace UPS, but we’ll still use CVMFS for distribution

Framework — *art*

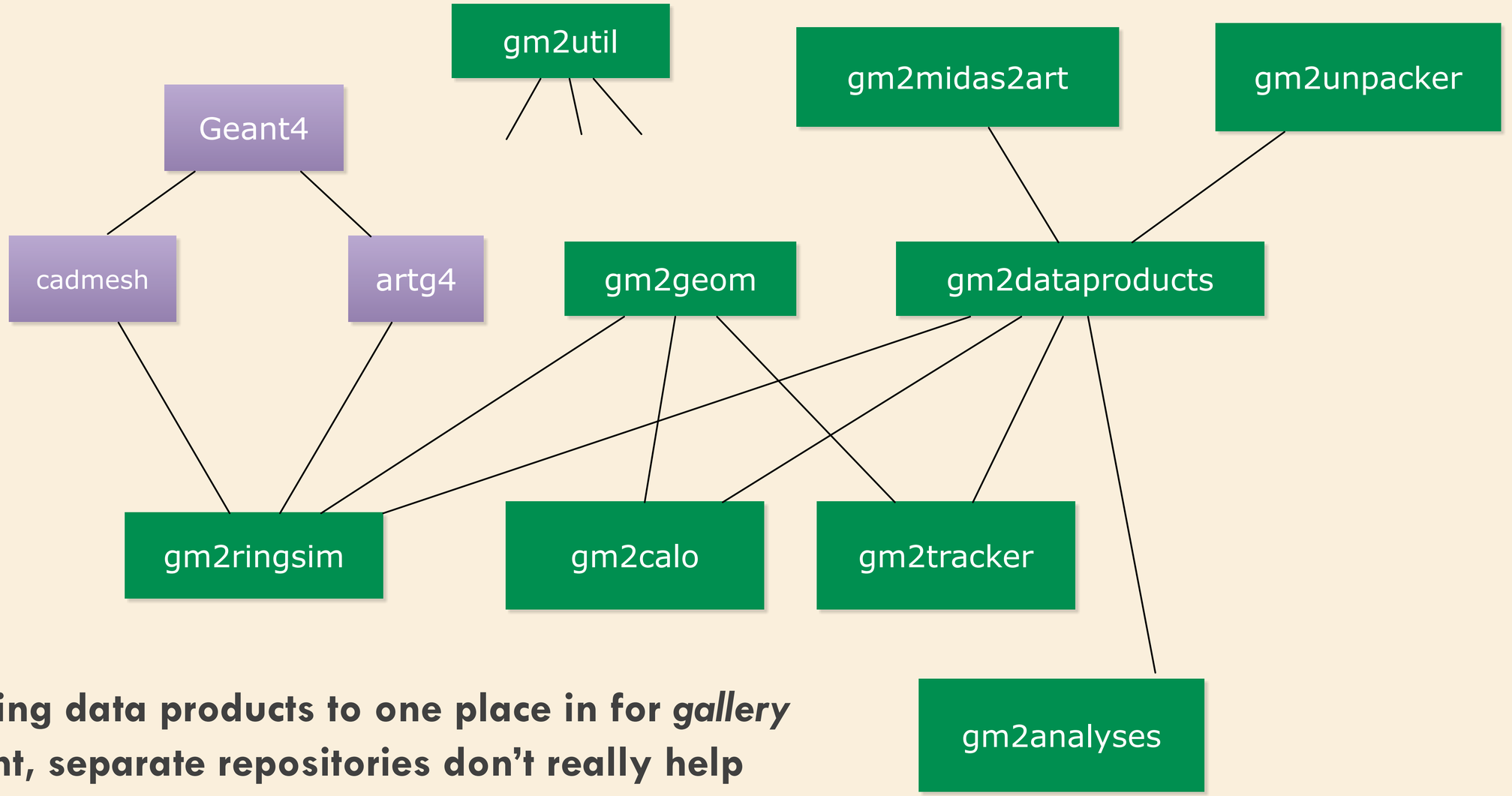
- I think I can write that we are all enthusiastic believers in and users of *art*

- Our simulation system (artg4) has unique flexibility because of *art*
- We are beginning to use a 3D visualization system with unique flexibility because of *art*

- We are pushing some envelopes…
  - Internal module multithreading with TBB (e.g. process a calorimeter per thread)
  - Real time network i/o with Zero-MQ (e.g. for data quality monitoring)

- We are successful in leveraging expertise…
  - Redefining events with input source (learned from 35ton)
  - Reading raw data format into art (learned from NOvA)
  - Making code more efficient (learned from MicroBooNE)
  - We consult with *art* experts often and appreciate their help

Software dependency layout



We're moving data products to one place in for *gallery*
At this point, separate repositories don't really help

Fermilab

Data Tiers (K-S Shaw 2015-11-12)

- Raw “hits” from DAQ or Simulation

RawIslands
(t, V)

- Calibrated and reconstructed <u>detector level</u> data

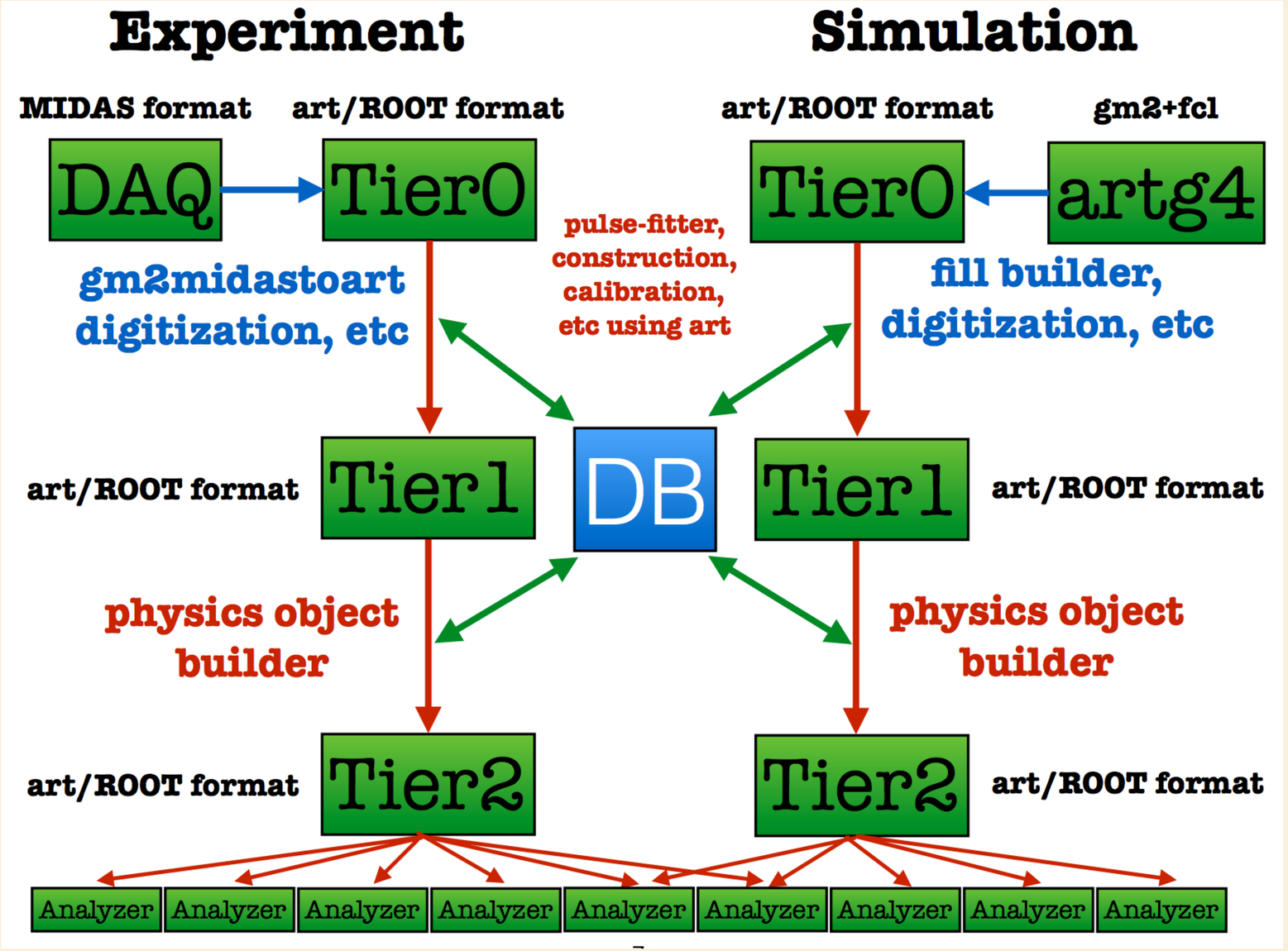
CrystalHits
(t, E, χ^2)

- Reconstructed <u>physics</u> objects ready for analysis

CaloClusters
(t_{clus}, E_{clus}, w_{clus})

Fermilab

Data tiers



K-S Khaw 2015-11-21

Fermilab

Databases

- We need databases for…
  - Hardware production information
  - Run parameters (MIDAS ODB)
  - Slow control readings and settings
  - Calibration data
  - Perhaps geometry / alignment information
  - Beam information [IFBeam]

- Mantra: We should use existing applications and systems

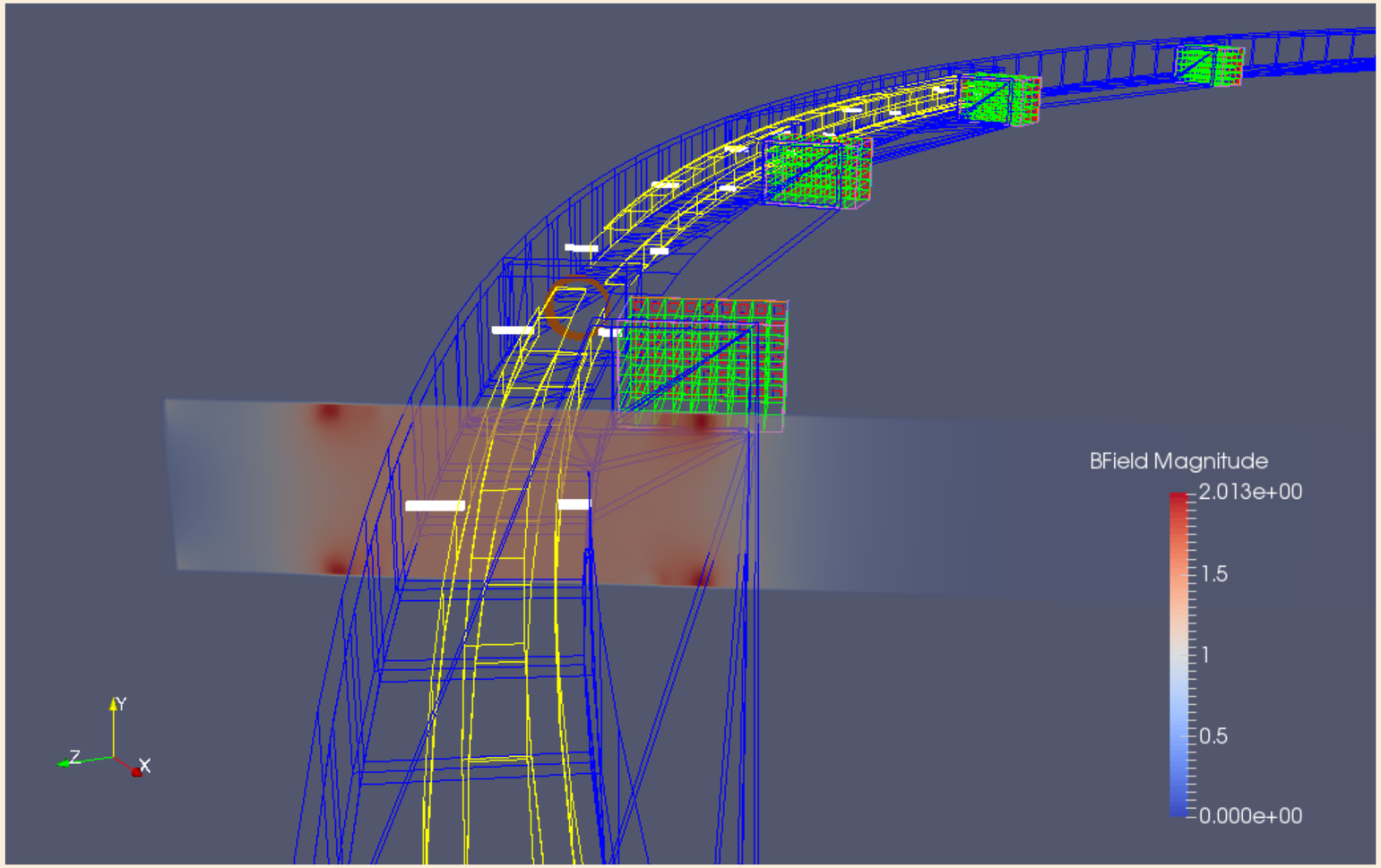
- The Shanghai group owns this
- We have detailed requirements from g-2 systems

- Consulting with Steve W and Igor M for best path forward

- [See Databases Breakout talk / Dikai]

3D Visualization with ParaView

- Crucial for simulation verification — see talks at breakout



Fermilab

Interactive Analysis

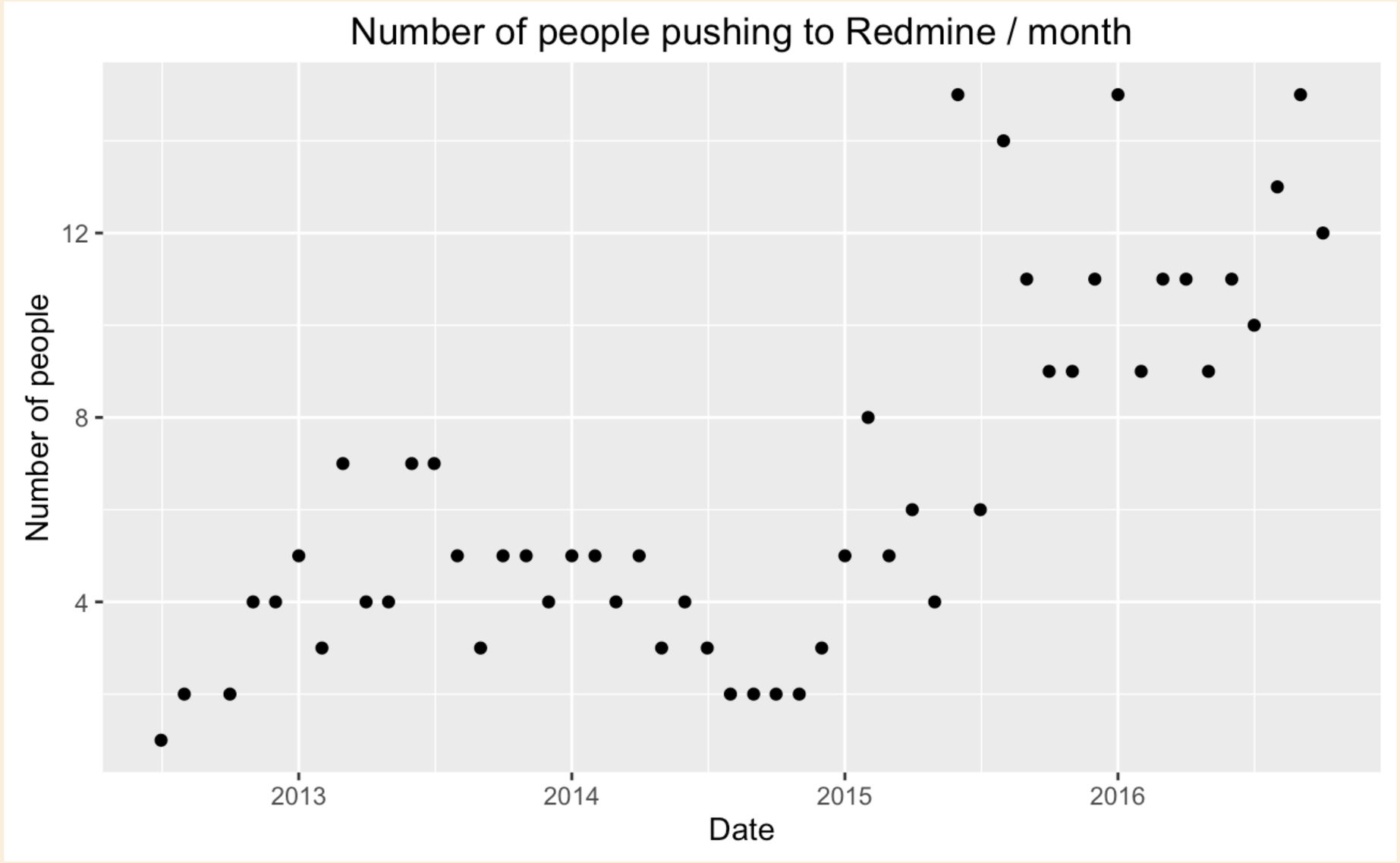
- Root - of course
- Want to try gallery and open up other analysis tools like R and Python and ?

- How does the “spokesperson” look at the data?
  - For SLAC testbeam, converted art files into flat TTree’s
  - Easy to make a plot with Root
  - But the risk here is the start of another ecosystem
  - We’d like to avoid multiple ecosystems on g-2

- A “makestudy” art product will be useful
  - Set up a directory and set up environment with one command
  - Make a ready-to-go skeleton file

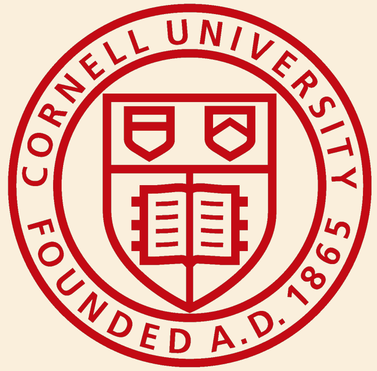
- Gallery is non-trivial and unlike Root - would like to see usability improvements

Developers

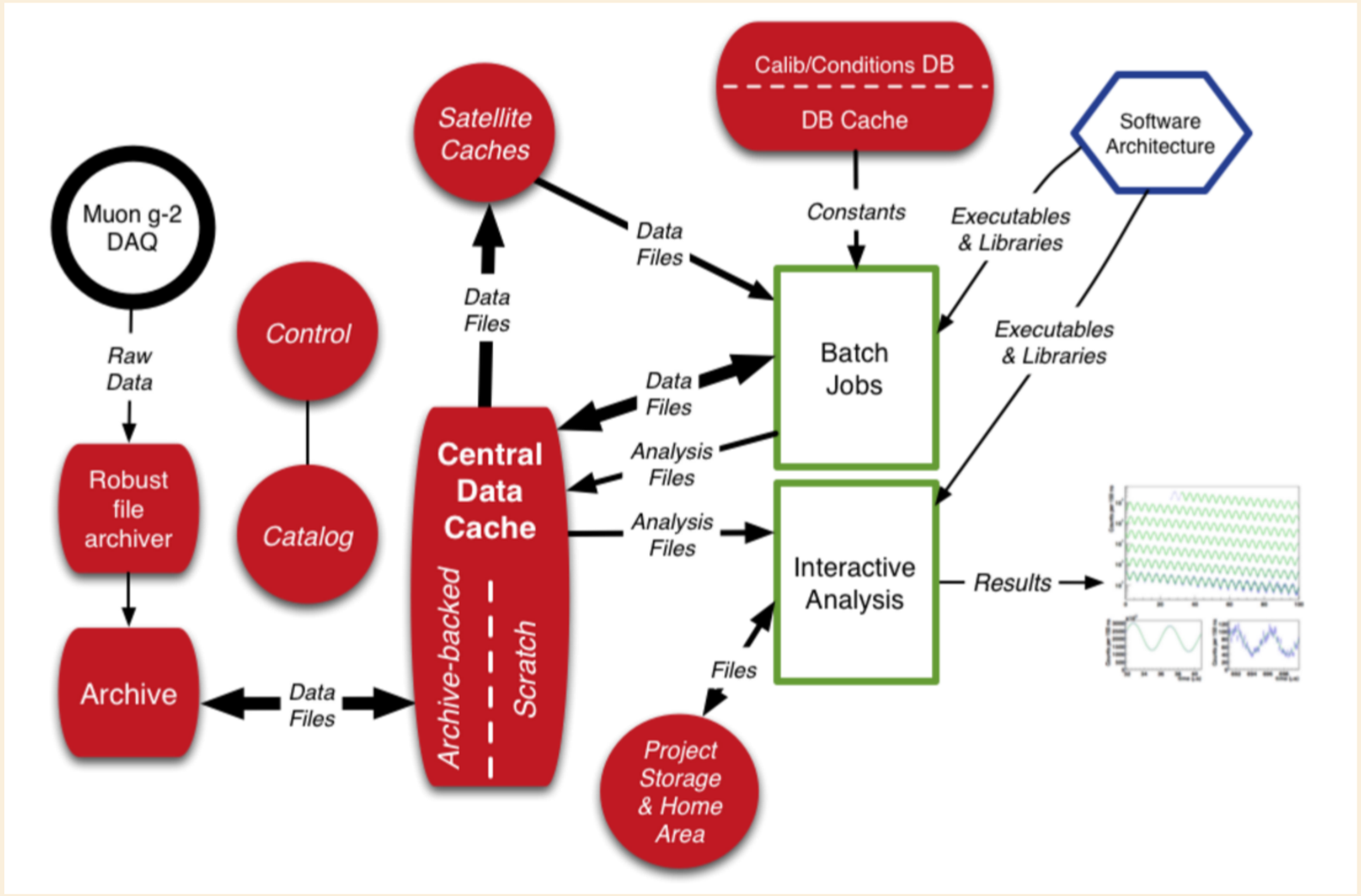


Fermilab

Participation in Offline work



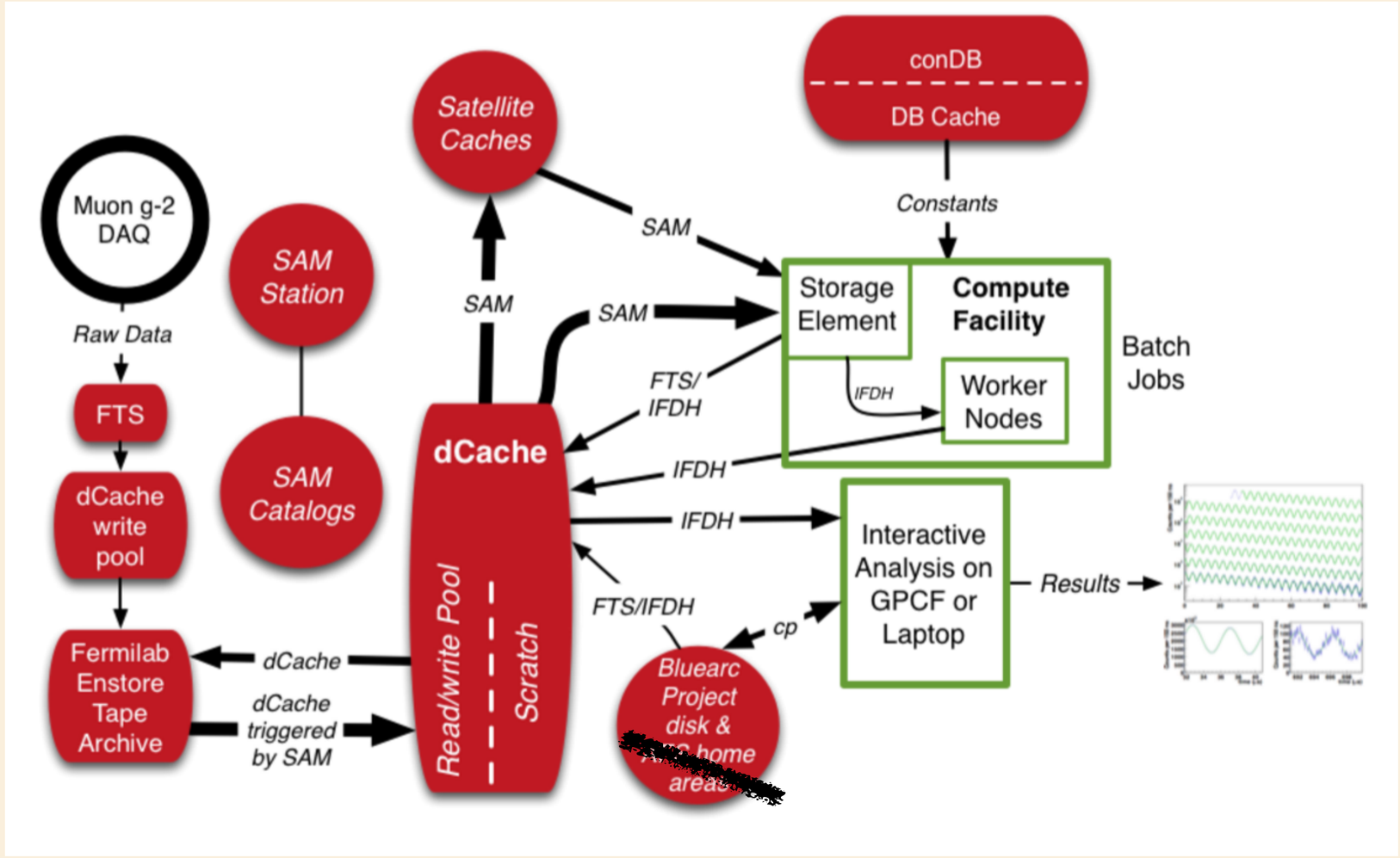
Data Management Architecture



Fermilab

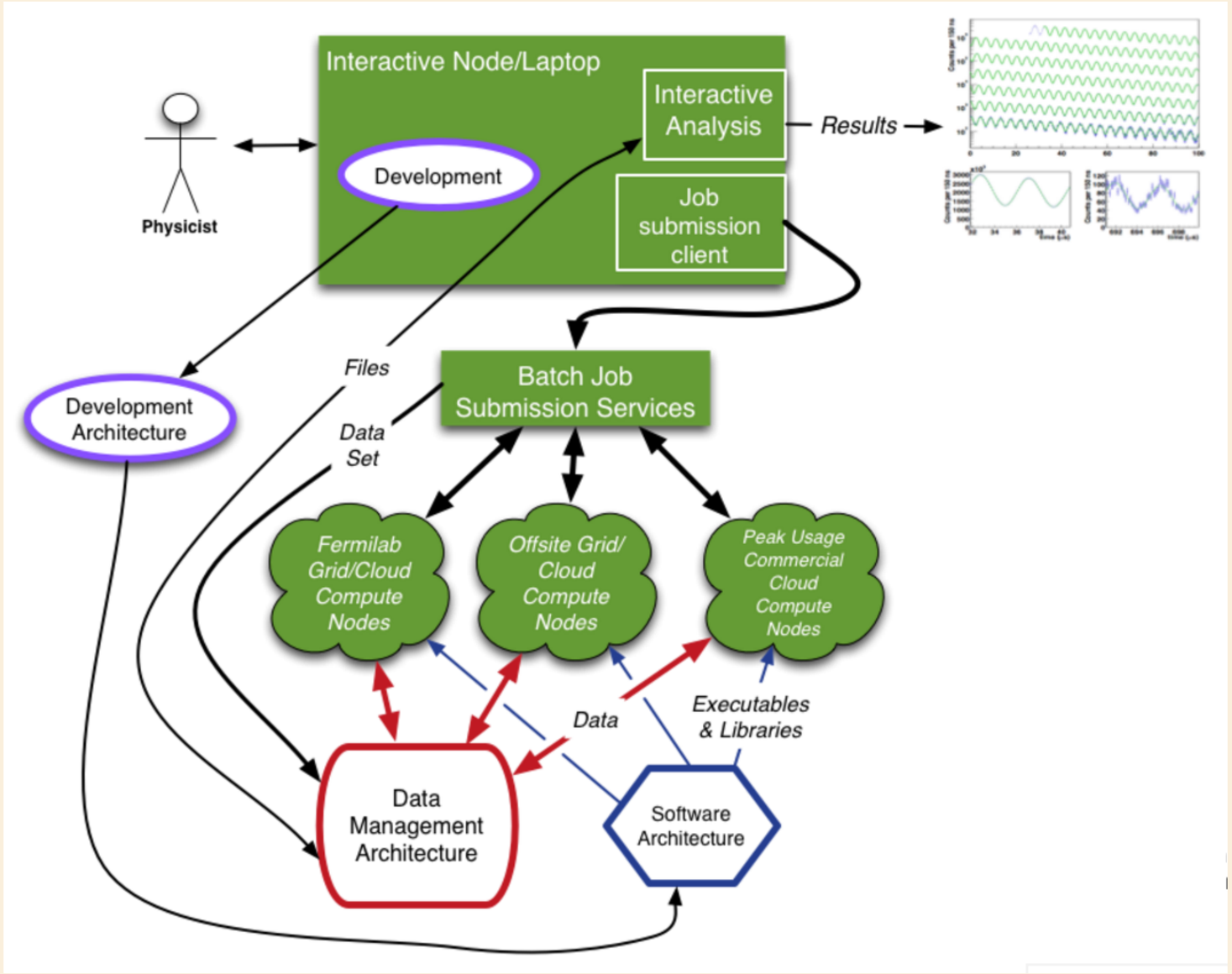
Data Management Implementation

- **Our Mantra:**
  <u>We won't reinvent the wheel</u>
- **We use the standard tools the standard ways**
- **FTS will be used for the DAQ (already in place)**
- **We're using the dCache pools including scratch and experiment specific**



Fermilab

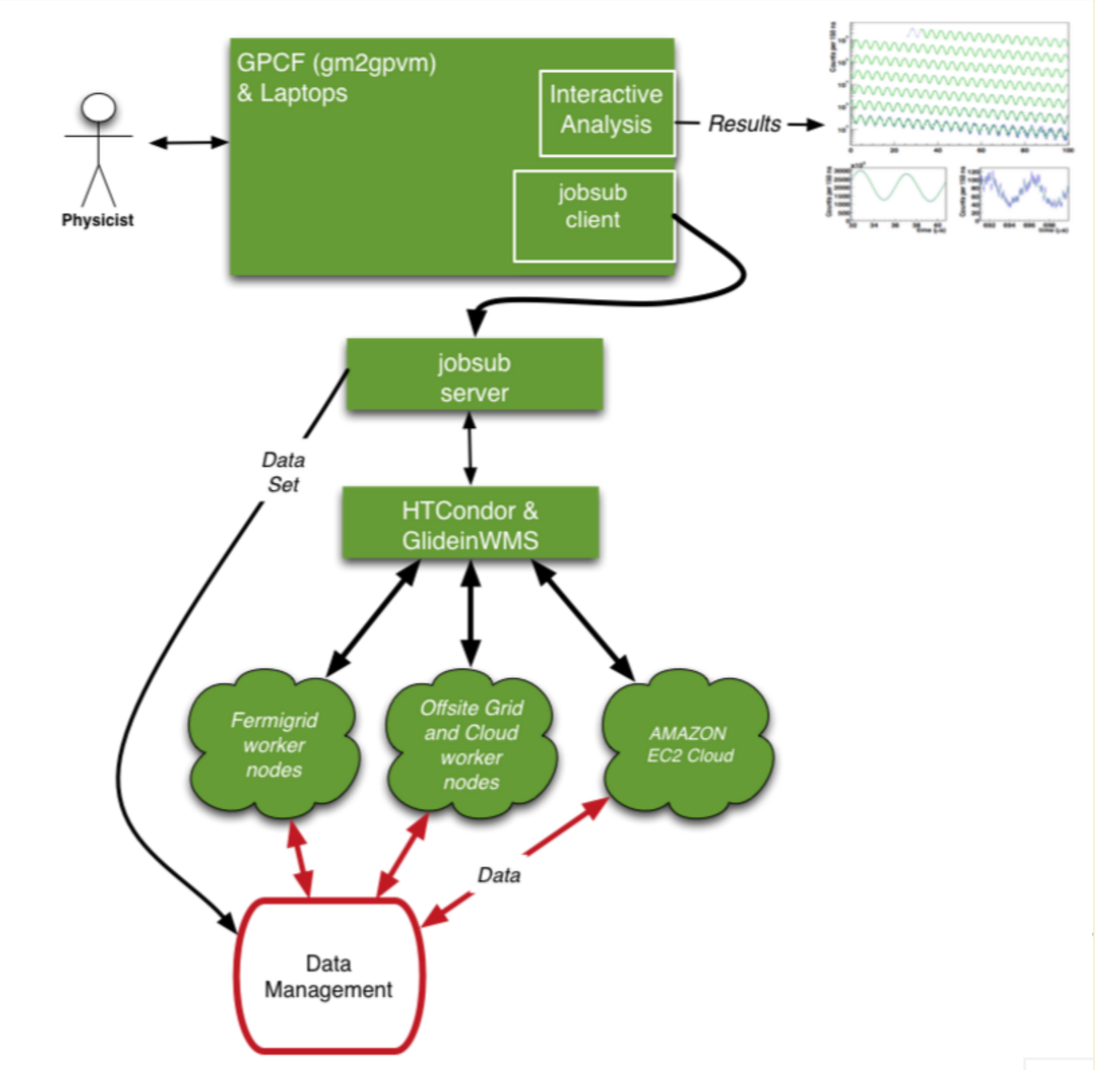
Execution Architecture



Fermilab

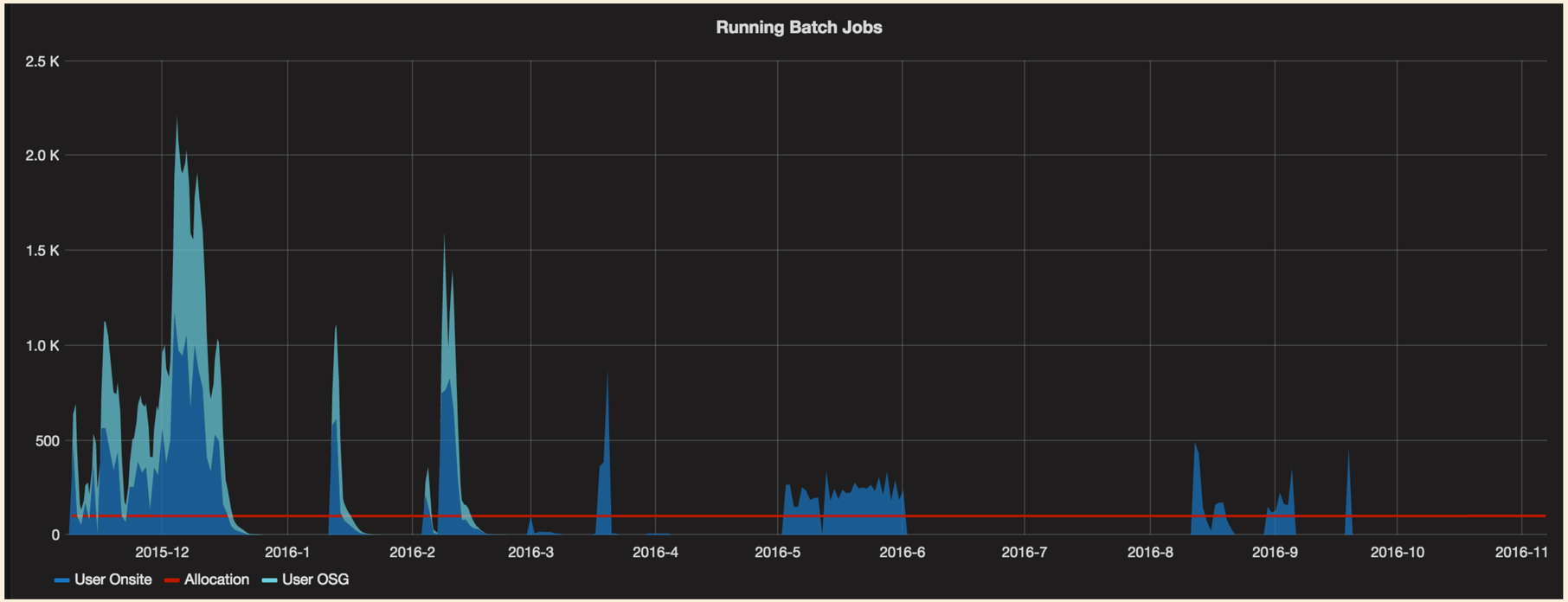
Execution Implementation

- We use jobsub the standard way
- We have python scripts based on NOvA and vetted by (former) production group
- Need to investigate POMS
- We have not yet needed resources on OSG
  Would like to leverage Mu2e expertise there
- We are a test case for HPC use
  (artg4 makes parallelization straightforward)
- [See Job Submission & Workflow Breakout Talk / Tammy]



Fermilab

Data & Execution Summary



- Storage:
  - 4 TB g-2 persistent
  - 10 TB scratch
  - 60 TB RW (Tape)

Fermilab

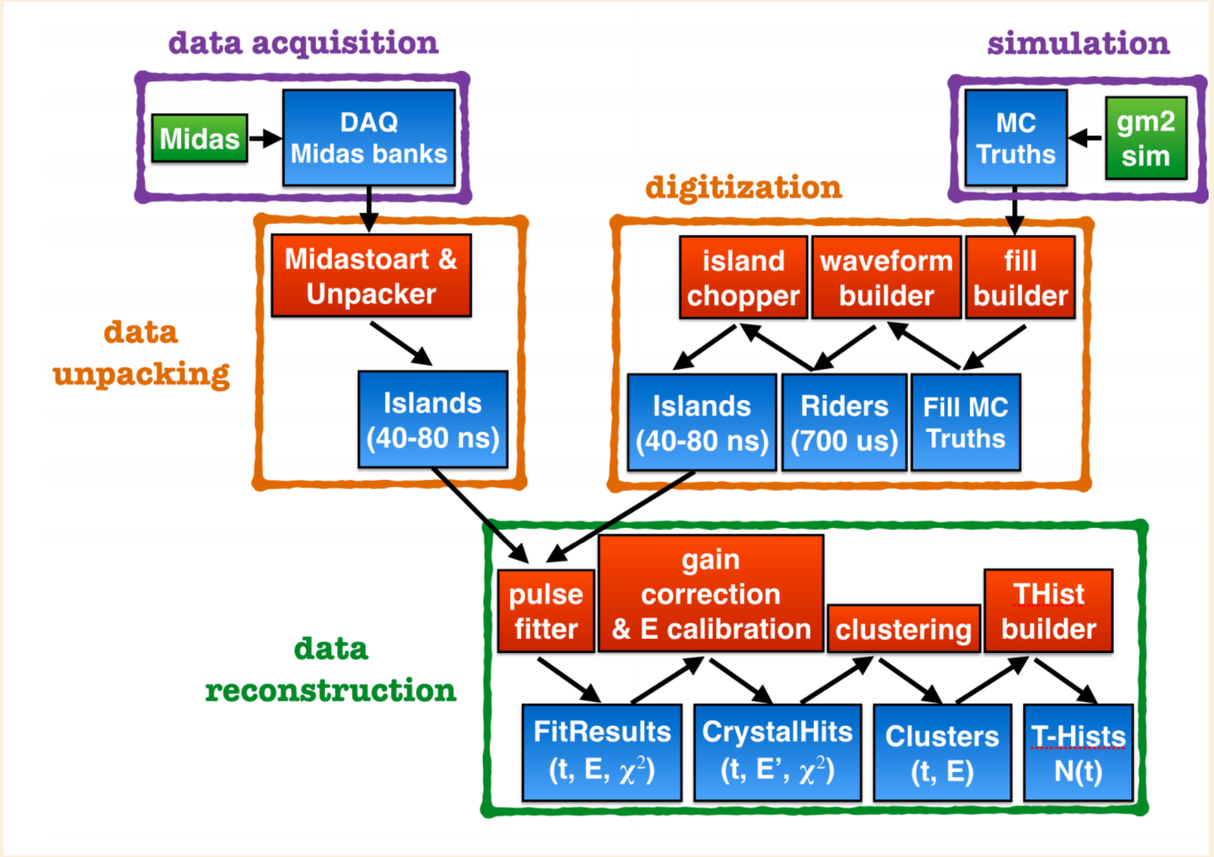
Production tasks

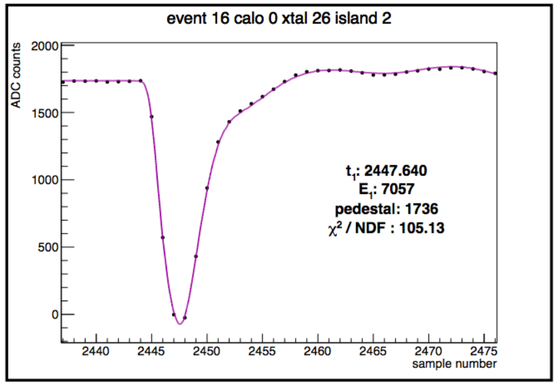
- Simulation campaigns
  - Generate samples for lost muon study; practice samples
  - [See Simulation Overview talk / James]

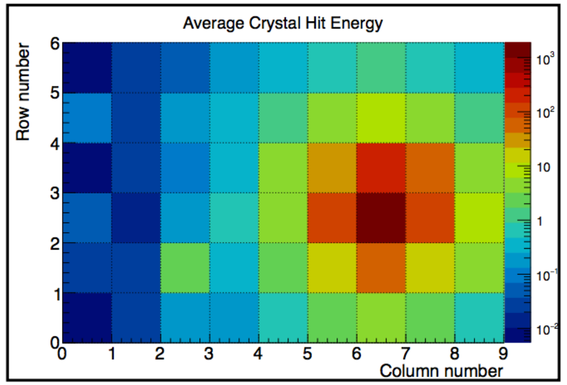
- Reconstruction
  - T method — convert digitized waveform islands into positron energies and time
    - Calorimeter clustering, fitting [See Calorimeter Algorithm talk / Aaron]
  - Q method
    - Similar to T method, but no energy threshold — histogram all the data
  - Tracker
    - Track finding and fitting (hard due to varying magnetic field)
    - Pitch correction and EDM limit
    - [See Tracking Algorithm talk / Tammy]
  - Auxiliary Detectors

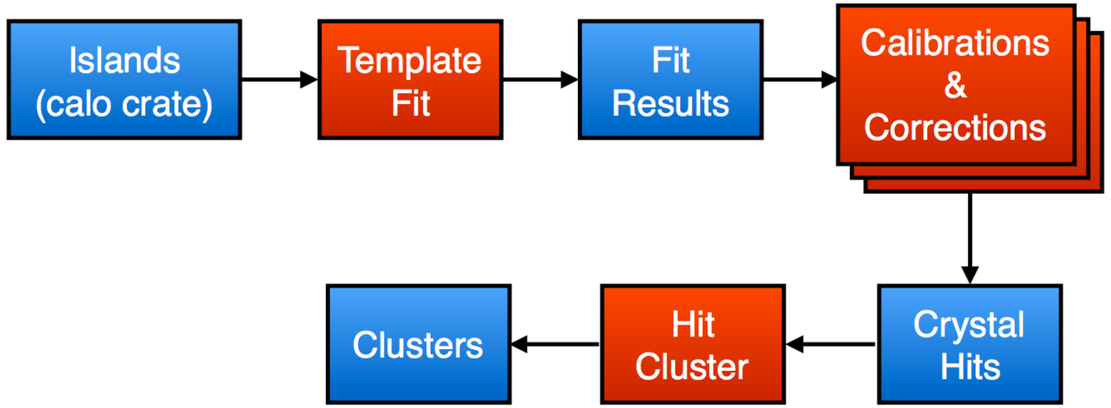
Fermilab

Calorimeter processing









K-S Khaw 2016-07-26

Fermilab

Processing statistics

- See MicroBooNE style spreadsheet
- The experiment will take ~300M fills
- Target is to keep up with data taking (12 fills/s)
- So our time budget is 80 CPU-ms/fill or with 1000 CPUs 80 s/fill
- Calorimetry reconstruction is 1 s/fill
- Tracking reconstruction is ~ 100 s/fill (rough order of magnitude)

- "Faster" Simulation is ~ 1s/muon, 16,000 muons per fill
  - Generate 4.5 fills/hour/CPU; 3M fills/month with 1000 CPUs

- Lots of optimization opportunities
- Learning from other experiments

Introduction to Nearline Processing

- DQM is real time monitoring of apparatus (plots update immediately)
  Statistics on order of a few fills is fine (e.g. 1 sec worth = 12 fills)
  Very simple reconstruction
  art + ZeroMQ

- Nearline is fast turn-around reconstruction requiring more statistics
  (30 minutes ~ 22K fills)

- Physics requirements of Nearline:
  Monitor apparatus/physics parameters that require statistics in order to discover problems on time scale < 1 hr — e.g. beam corrections
  Was found to be crucial for SLAC testbeam - will use for e.g. pedestals, proto-wiggle plots

- [See Nearline Breakout talk / Kim] - uses TBB for parallel tasks

Magnetic Field Measurement

- Another analysis asynchronous to omega_a
- Processing needs are very small compared to omega_a
- Still early days for the field analysis group
- [See Magnetic Field Analysis breakout talk / Ran]

We've already tested the system

- We've been using this ecosystem for years
- End-to-end Tracker Test beam [Tip-to-tail FNAL Talk / Joe]
- End-to-end Calorimeter Test Beam [Tip-to-tail SLAC / Kim]

- Exercised DAQ systems, *art* infrastructure, data management, algorithms
- Of course experiment is at a larger scale, but so far so good

- We already have experience before real data comes

- Would like to gain more experience with simulated data

Summary

- **Our experiment might be weird, but our offline computing is not**
- **We haven't reinvented the wheel and have leveraged infrastructure, tools and expertise to create an ecosystem I think is uniquely capable**
- **We have a comprehensive software ecosystem that works and improvements are desired: Documentation, github, Spack, POMS, CI, Release team**
- **Not complete: Databases, Completed simulation & reconstruction — we know this and are working hard**
- **Need to exercise system at scale, do more optimization studies**
- **We have an enthusiastic (and I think super-smart), but small developer base**
- **I'm confident that we're on track for data taking, but a perturbation could make for delays**
- **See many subsequent talks**