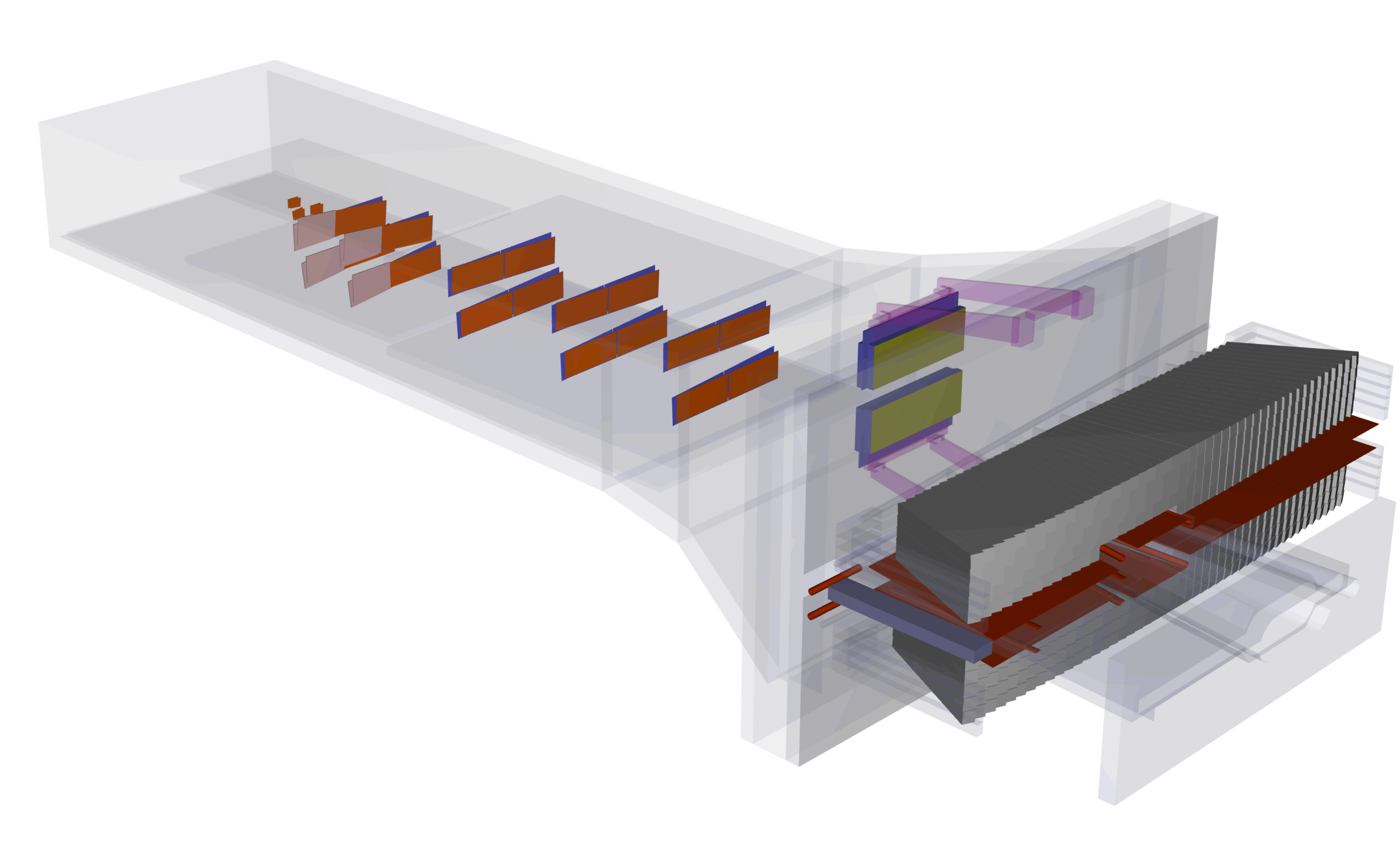


HPS Software

Presentation for DOE Review at SLAC

January 18, 2019

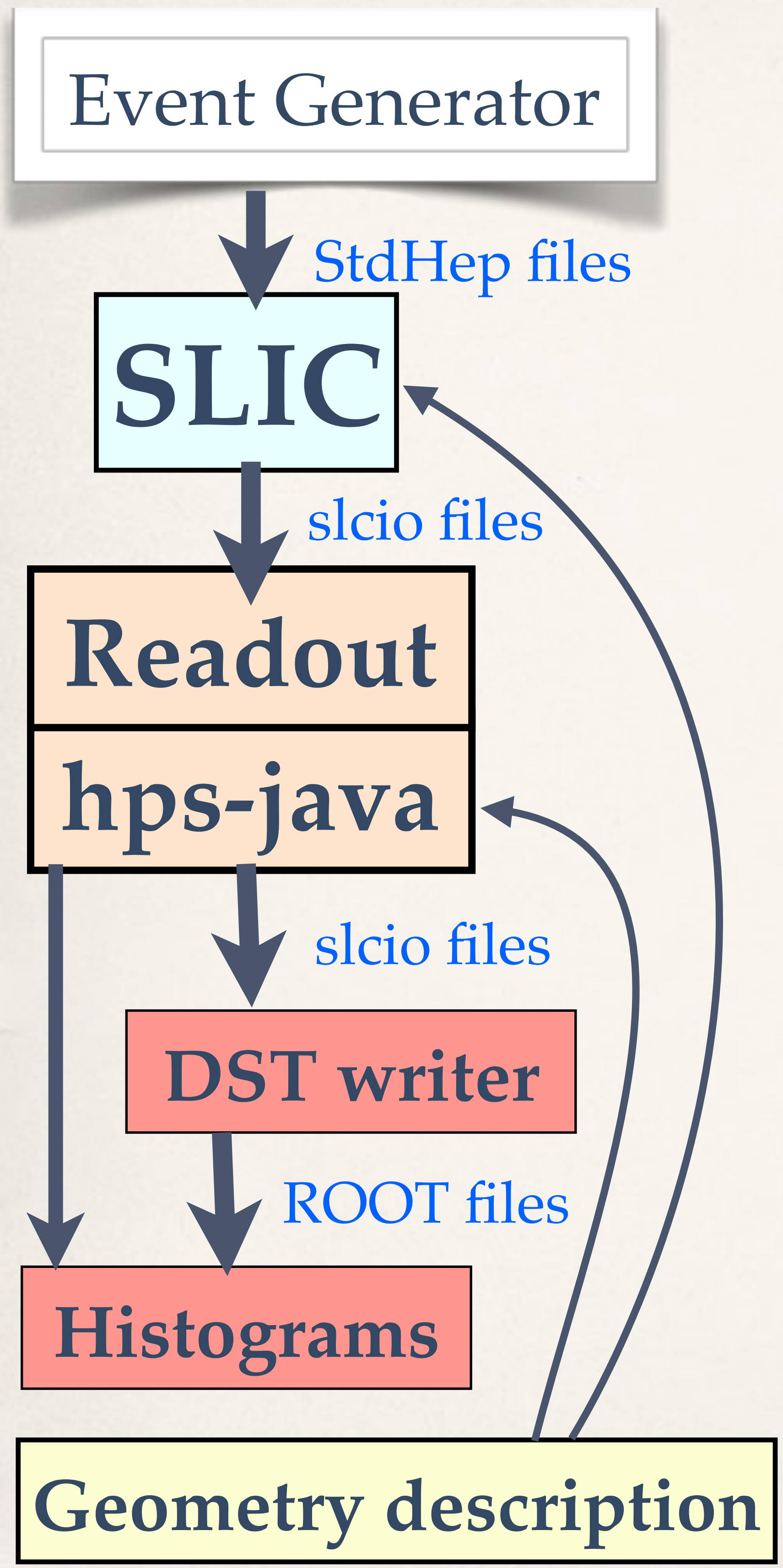
Overview

- Introduction
  - History
  - System Overview
  - Software Organization Overview
    - Software Cycle and Releases
  - Software Group
- Codes:
  - Generators, MC code
  - Reconstruction, Tracking,
  - Calibration, Monitoring.
- System resource utilization
- Outstanding Task List
- Historic and projected manpower
- Conclusions

Introduction - history

- Early decision by collaboration to leverage the existing expertise in the SLAC group with the Linear Collider Simulation, LCSim software framework.
  - JLab (CLAS12) software was too immature, and would not suffice for expected 6-GeV era run.
  - Not enough time and manpower to start from scratch.
- Result:
  - Development of "hps-java" code, which utilizes the "lcsim" framework.
    - +/- Main code development is in Java.
    - + Robust framework to develop on.
    - + Existing tracking component: seed tracker.
    - - No overlap with JLab code.
  - Main data storage model: LCIO.
    - + Read/write capabilities from Java and C++.
    - - Less flexibility in contents.

Introduction - System Overview: MC



A' events,
Background events

SLIC or hps-sim : Main GEANT4 based detector simulation.

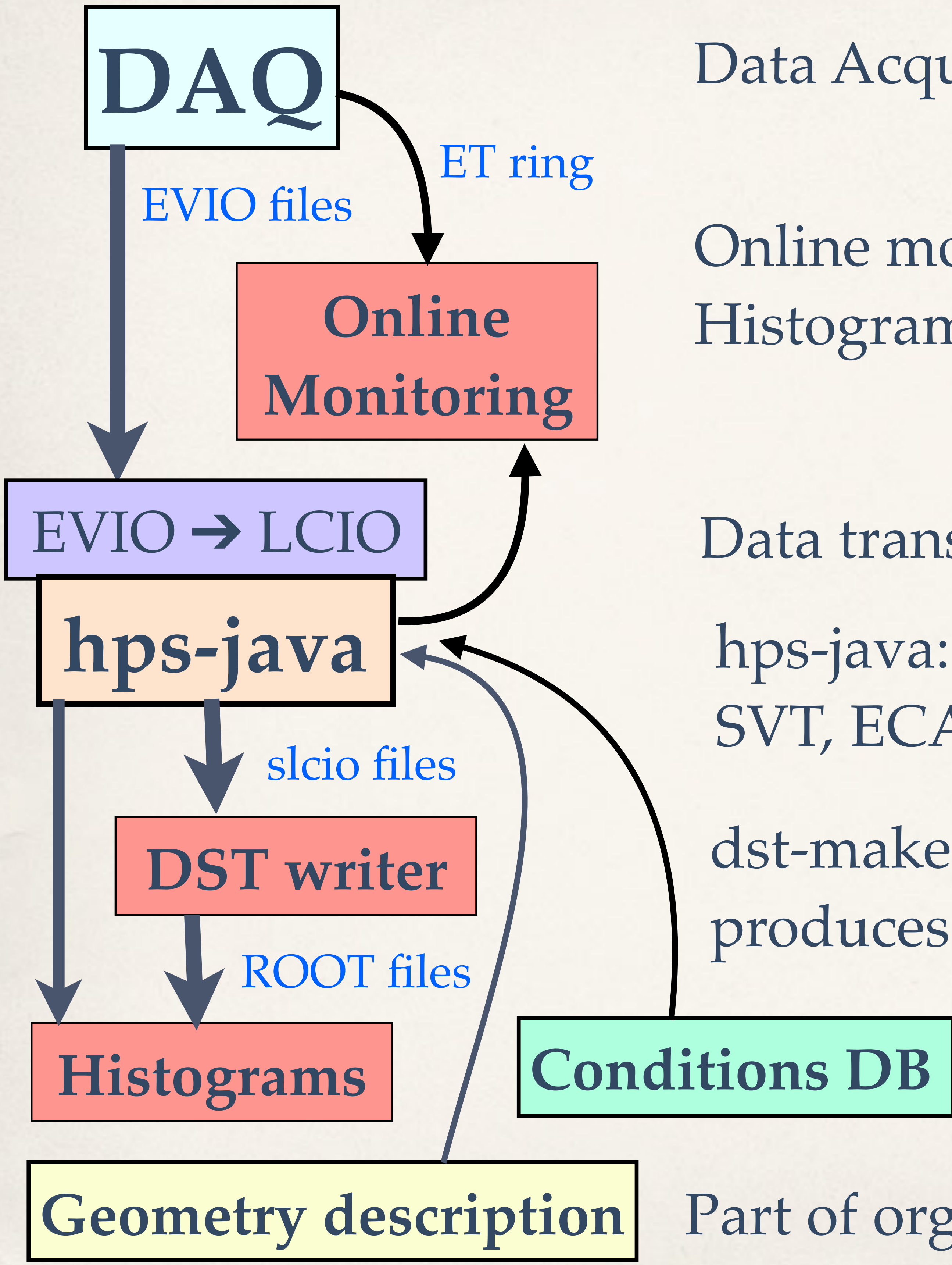
Readout: Simulates electronics and trigger.

hps-java: Analysis framework:
SVT, ECAL, Tracking, Hodoscope

dst-maker or hpstr - reads slcio files and produces data summary files.

Part of org.lcsim that computes geometries.

Introduction - System Overview: Data



Data Acquisition System

Online monitoring system
Histograms and Event display.

Data translation layer.

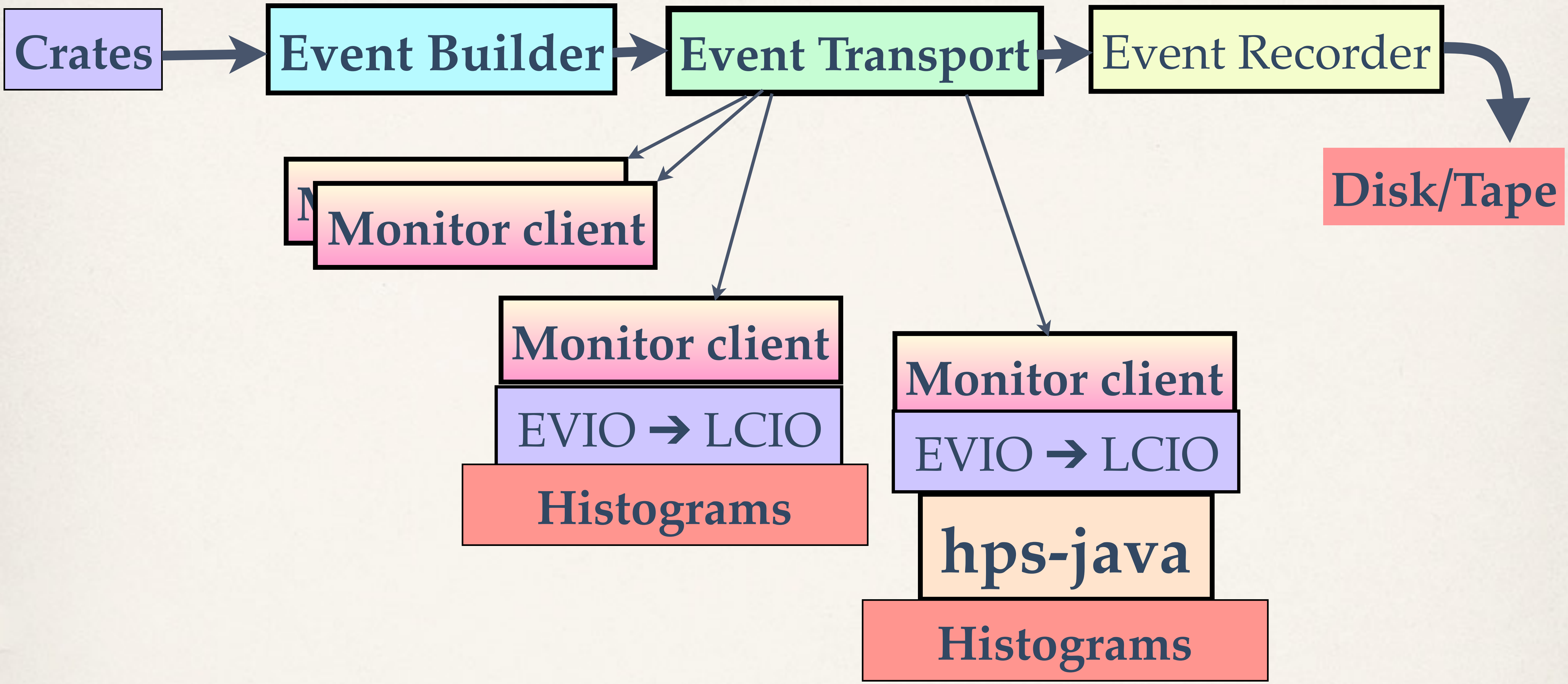
hps-java: Analysis framework:
SVT, ECAL, Tracking, Hodoscope

dst-maker or hpstr - reads slcio files and produces data summary files.

Stores run by run information.

Part of org.lcsim that computes geometries.

Introduction - System Overview: Online



DAQ uses the EVIO format internally and for data storage of raw data.
Event transport distributes and transports events.
Monitoring clients use EVIO or the EVIO → LCIO translation layer.

Introduction - System Overview

- **Calibrations codes:**
  - SVT online calibration code - timing in, pedestals, gains.
    - Existing code that runs during commissioning to time in and check SVT.
  - ECal calibration - Cosmic ray calibration, Full Energy Electron calibration.
    - Existing code to calibrate ECal pedestals, gains and timing.
  - Hodoscope calibration.
    - Code needs to be written, but can borrow from ECal code.
  - Detector Alignment - Millipede II
    - Complicated procedure for getting a good alignment.
    - High on Tracking Group priority list to simplify and improve this procedure.
- **Physics Analysis Code:**
  - Runs after data reconstruction.
  - Was in the domain of individual analyzers, but is now becoming more centralized.
  - See presentation by Nathan Baltzell.

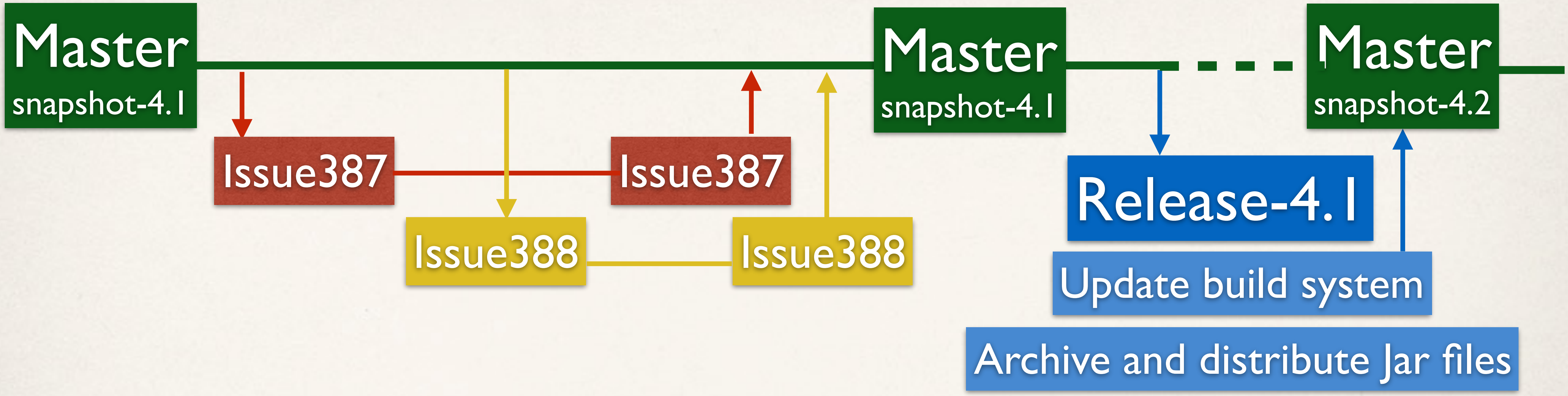
Introduction - Software Organization

- **Code repository - GitHub**
  - Tracks code, allows development on branches
  - Merging only through "pull requests", which must be approved.
- **Issue Tracking - GitHub**
  - Couples code issues with branches.
- **Code Documentation - Confluence Wiki + Java Doc**
- **Build System - Maven**
- **Testing - Maven integration tests.**
- **Continuous integration testing - Jenkins / Hudson**
- **Code profiles - JProfiler**
  - Output to web pages at: http://nuclear.unh.edu/HPS/Profiles
- **Releases: - Github + Maven + Nexus.**
  - Release is tagged on GitHub.
  - Resulting JAR file is available for download from Nexus.

Software Cycle and Releases

- We make extensive use of the “git” code management system, with GitHub as the repository site.
  - Jefferson lab is an established organization on GitHub.
- Development cycle:
  - “Clone” master branch of repository.
    - This gives the latest official version of the code. A “snapshot” release.
  - Create an issue on GitHub and a corresponding branch of the master code.
  - Develop on your branch. The master branch is not affected.
  - When ready, run integration tests to make sure nothing is now broken.
  - Create “Pull request”, with some documentation.
  - Pull request is reviewed. If approved, branch is merged with master.

Software Cycle and Releases



- Either after a milestone in the development, or before a set of data is processed, we create a new formal release of the software.
  - Current state of the software is tagged in git with a new release number.
  - The build system is updated to increase snapshot number.
  - The resulting jar file for the release is put on Nexus for distribution.
- Development continues. The master branch now produces the next snapshot version of the code.

Introduction - Software Group

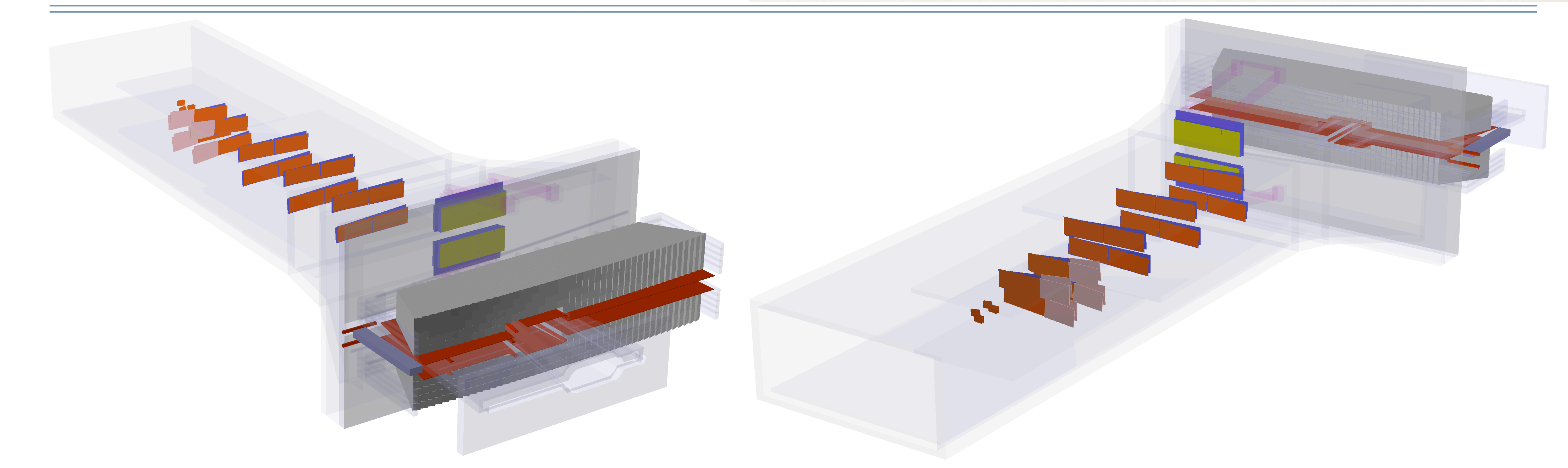
- Bi- Weekly meetings with online presentations.
- Software group mailing list
- SLACK for more immediate communication.

- **People:**
  - Lead: Maurik Holtrop
  - Tracking lead: Norman Graf
  - MC Generators: Takashi Maruyama
    - MC data production: Bradley Yale
  - Trigger: Valeri Kubarovsky
    - Trigger code: Kyle McCarty
  - Data Processing: Rafayel Paremuzyan
  - Analysis software: Matt Graham, Nathan Baltzell
  - Specific codes:
    - DST code: Omar Moreno
    - MC Simulation code, conditions system: Jeremy McCormick

Monte Carlo Generators

- The MC physics generators simulate the beam interaction with the target. HPS Expert: Takashi Maruyama.
  - HPS is sensitive to the tails of some distributions which are not fully represented in the GEANT4 simulation, so other tools are required:
    - EGS5 - Electro-Magnetic (EM) interactions.
    - GEANT4 - EM, hadronic and neutron production.
    - MadGraph/MadEvent - Trident (background) production and A' (signal) production, Wide Angle Bremsstrahlung (WAB) production.
  - The output of these various generators are combined according to cross section.
    - This creates a time realistic pulse train of "2 ns events", which represent a small period of real-time. These "2 ns events" are then run through the detector simulation.
    - Many of these "2 ns events" are empty!
  - Detector readout computes triggers, similar to the hardware trigger.
  - Events for which a trigger is found are further analyzed.
  - Generated events can be biased so that the probability of finding a trigger is much larger than a random actual beam time period.

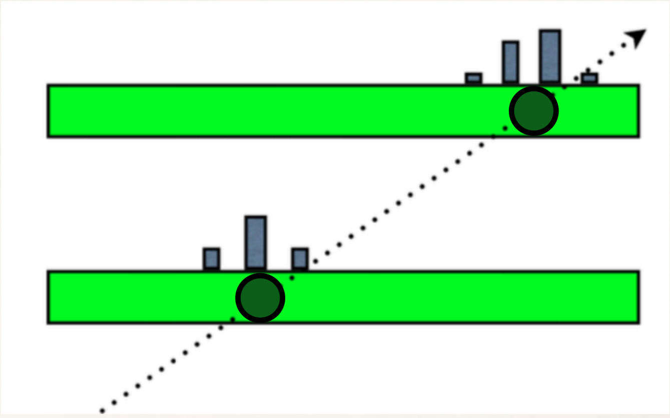
Monte Carlo Detector Model

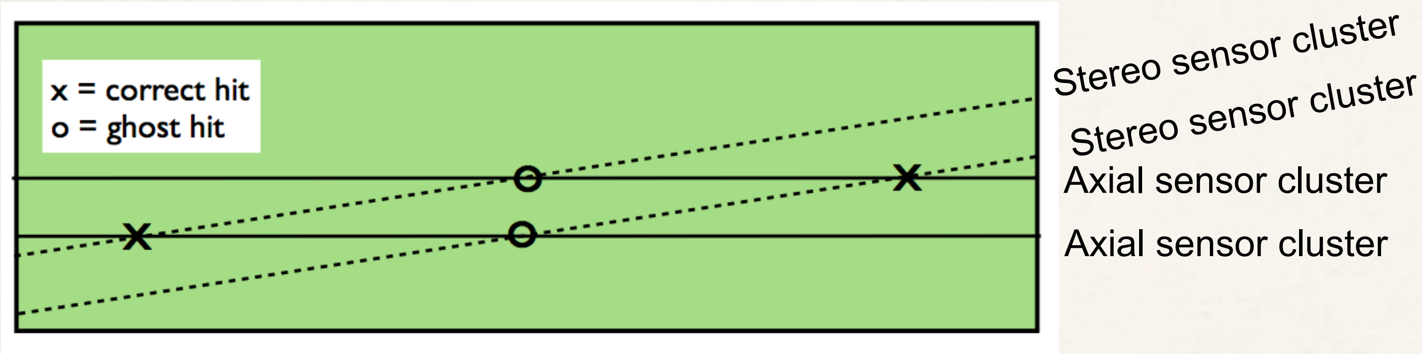


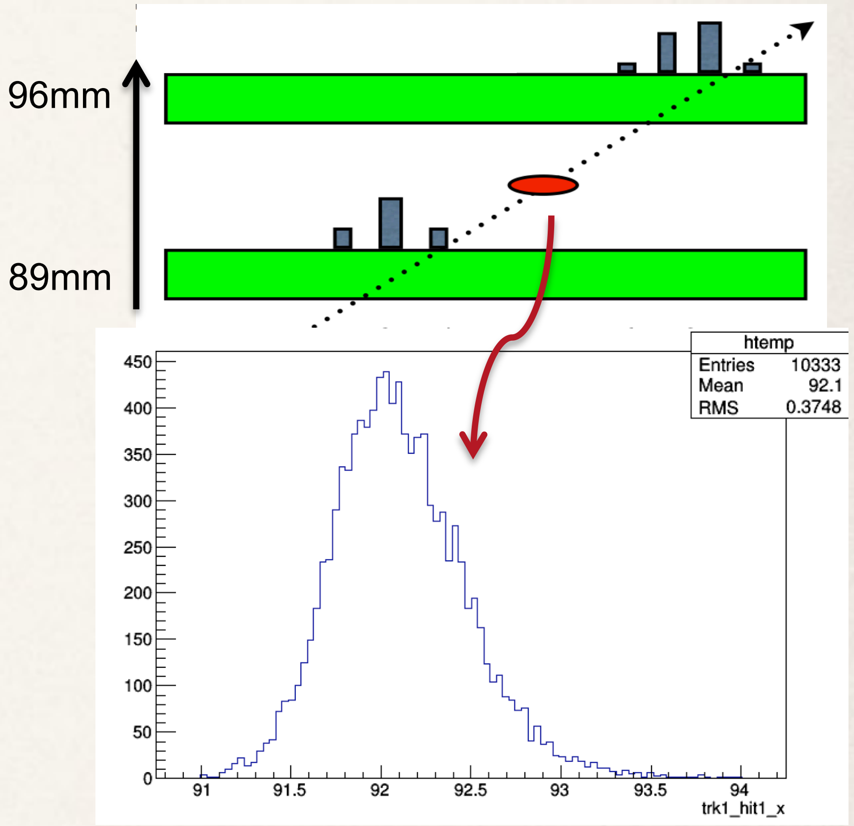
- The detector is accurately simulated using the GEANT4 framework.
  - Viewable at: http://nuclear.unh.edu/HPS/Detector/
- All active components are accurately rendered.
- Most of the inactive components that could interact with particles are accurately rendered.
- Two version of the code:
  - SLIC - current production version. Inherited from Linear Collider software.
  - hps-sim - rewrite. Uses same geometry, but has modernized features to allow more control over the events.

Tracking: Hit building

- First steps:
  - Sensor Hit Reconstruction
    - Build a 1-D hit from the signals on the strips.
  - Stereo Hit Reconstruction.
    - Combine two adjacent 1-D hits into a 3-D hit.

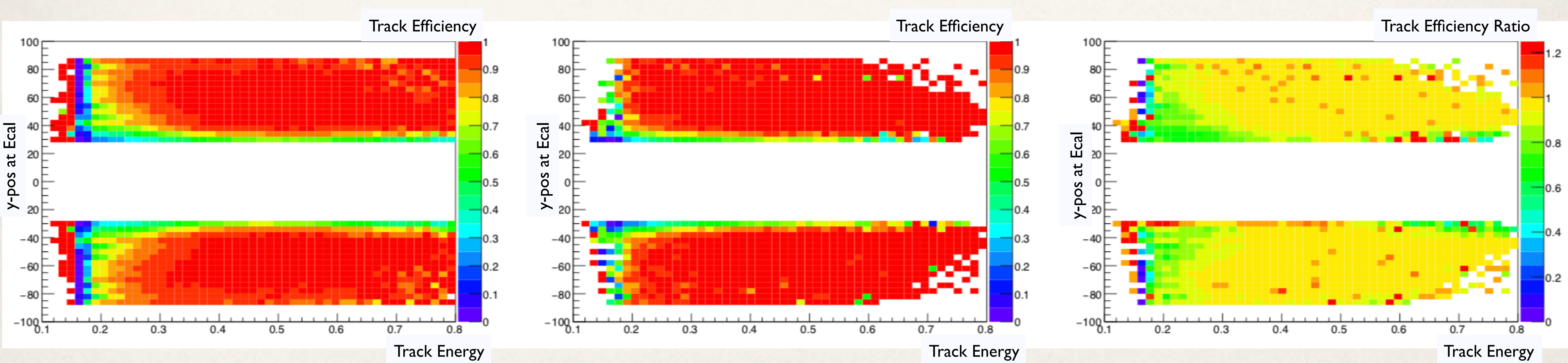






Tracking: Seed Tracker

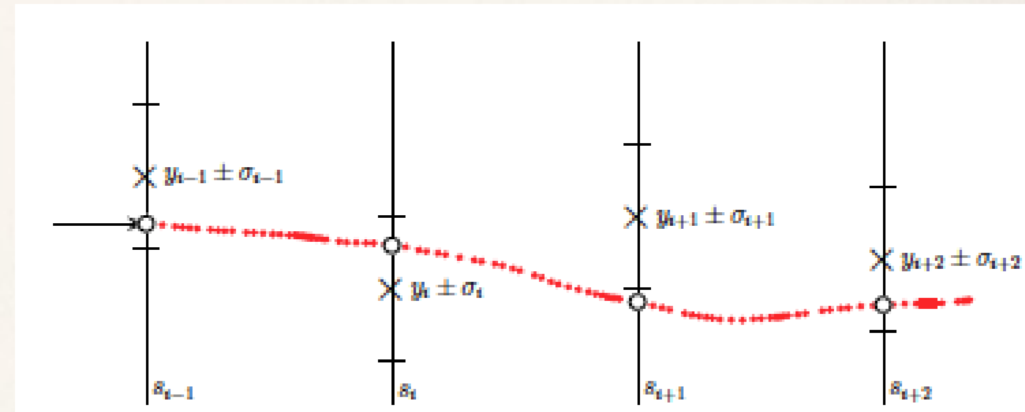
- Second Step: Seed Tracker
  - Largely inherited from Linear Collider.
  - Start with 3-hit track seed.
  - Add hit from confirm layer.
  - Add hit(s) from extend layer.
  - Algorithm allows for different layer combinations to create a seed track.
  - We use 4 combinations. (Seed345Conf2Ext16, Seed456Conf3Extd21, Seed123Conf4Extd56, Seed123Conf5Extd46) .
    - Algorithms removes tracks that are found more than once.
  - Resulting tracking efficiency is high.

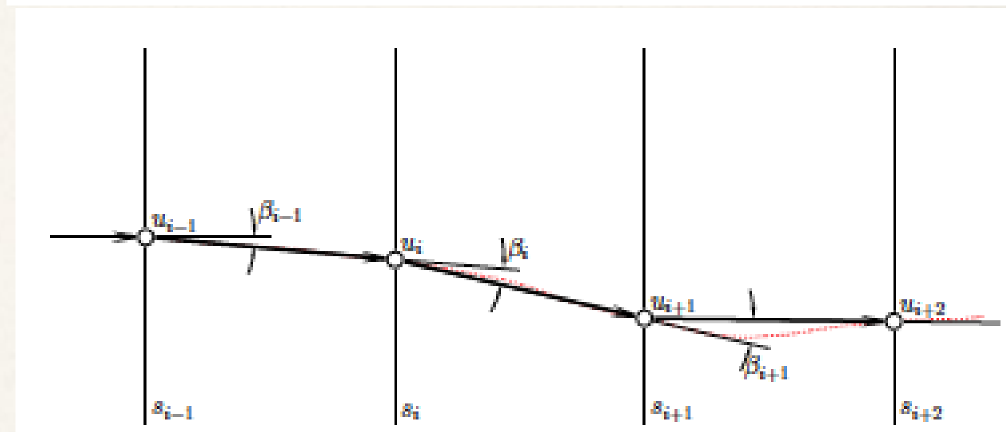


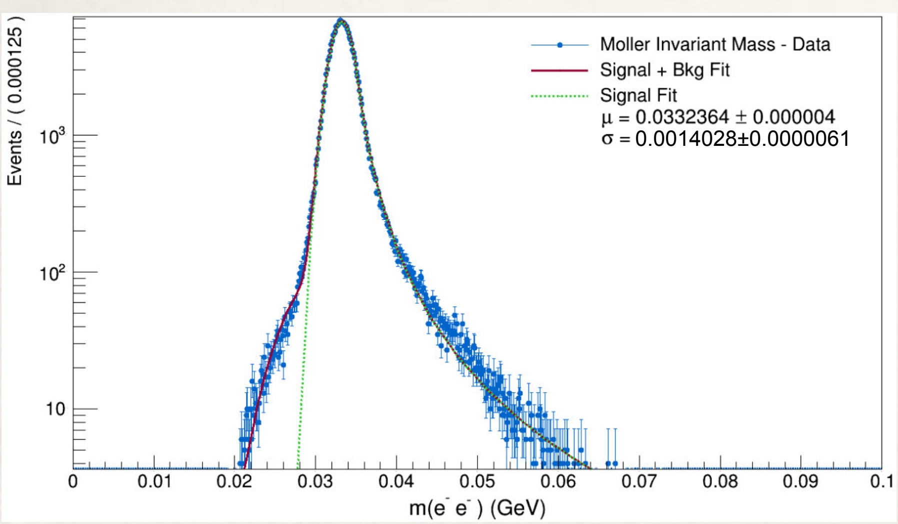
Tracking efficiency for electrons. Left data, center MC, right ratio of data/MC.
from HPS Note: trackEfficiency.pdf by Matt Graham

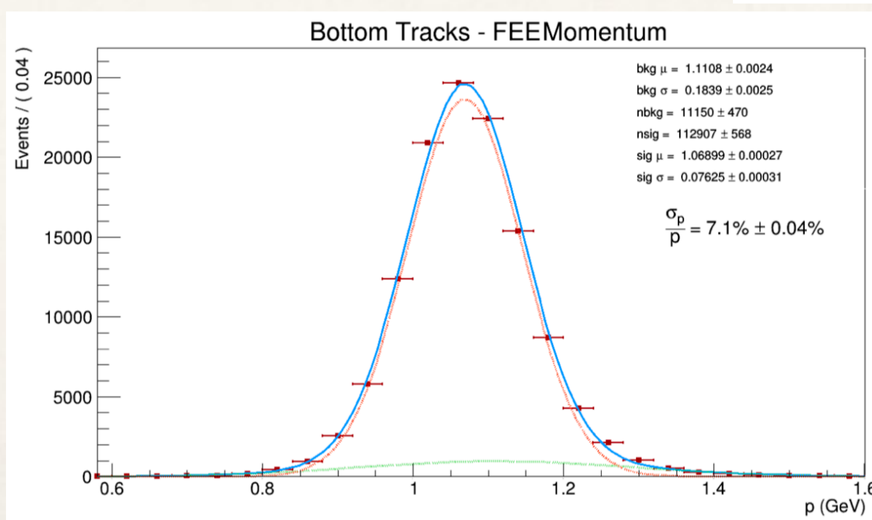
Tracking: GBL fit

- Final Step: Generalized Broken Line fit.
  - Seed Tracker tracks are refit with the GBL to improve resolution.
  - Resolution improves 20-30%
  - Fit also provides full covariant matrix of all track parameters.
- Resolution: $\frac{\sigma_p}{p} = 7.1\% \pm 0.04\%$
- Inv. Mass resolution for Møller events is 1.4 MeV at 33.2 MeV, for the 2015 data (1.05 GeV beam energy)









$$S(u) = \sum_{i=1}^{n} \frac{(y_i - u_i)^2}{\sigma_i^2} + \sum_{i=2}^{n-1} \frac{\beta_i^2}{\sigma_{\beta,i}^2}$$

Sources:
↑ Pelle Adrian
← Omar Moreno

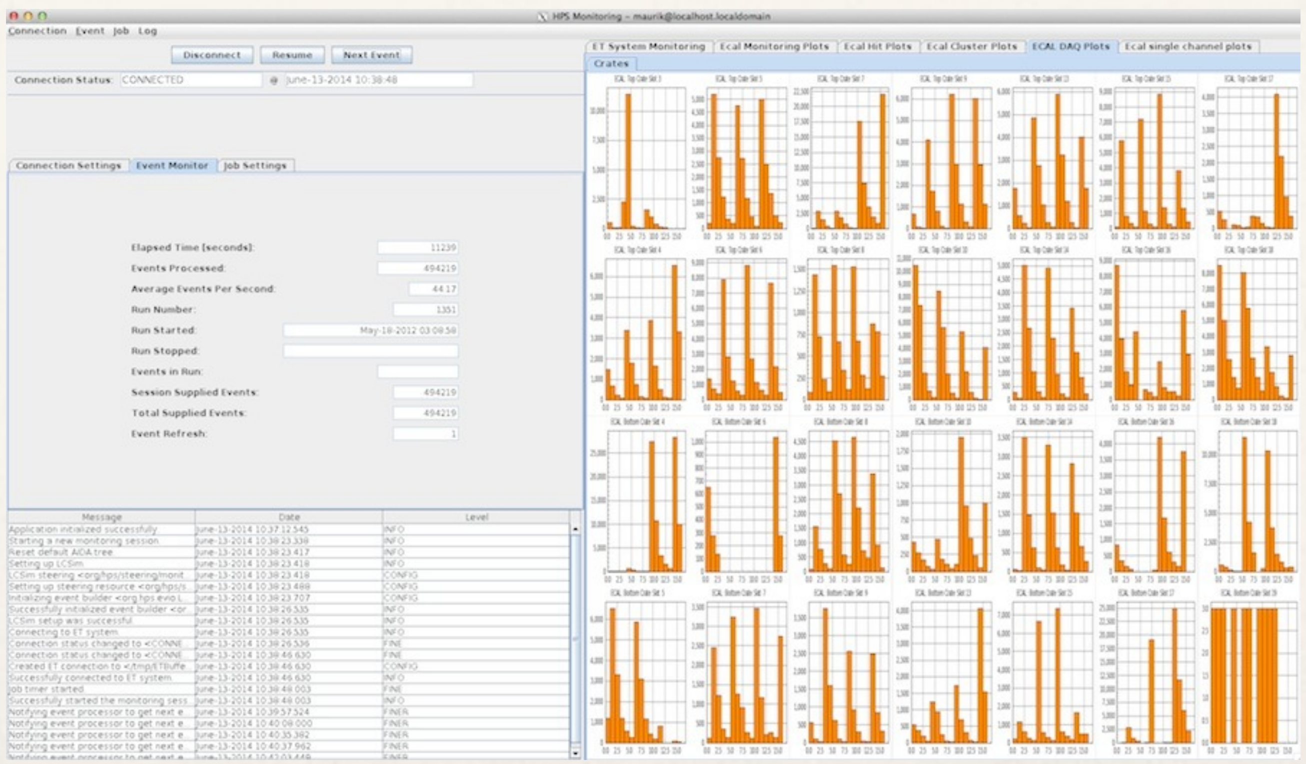
Alignment

- Initial alignment of detector is from surveys.
- Next, alignment is improved using track based alignment.
- We used Millipede-II, which works well with the GBL package.
  - Performs least square fit of local (track) and global (alignment) parameters on a track by track basis.
  - Large minimization problem with a lot of parameters.
  - Difficult, because not all the alignment parameters are well constrained, our data does not cover all tracking layers with tracks.
  - Experts: Alessandra Filippi and Norman Graf.

- This has been a fairly time consuming process.
  - Better understanding of the problem and improved procedures have made the alignment a lot better and faster.
  - Some further improvements could make it easier to perform.

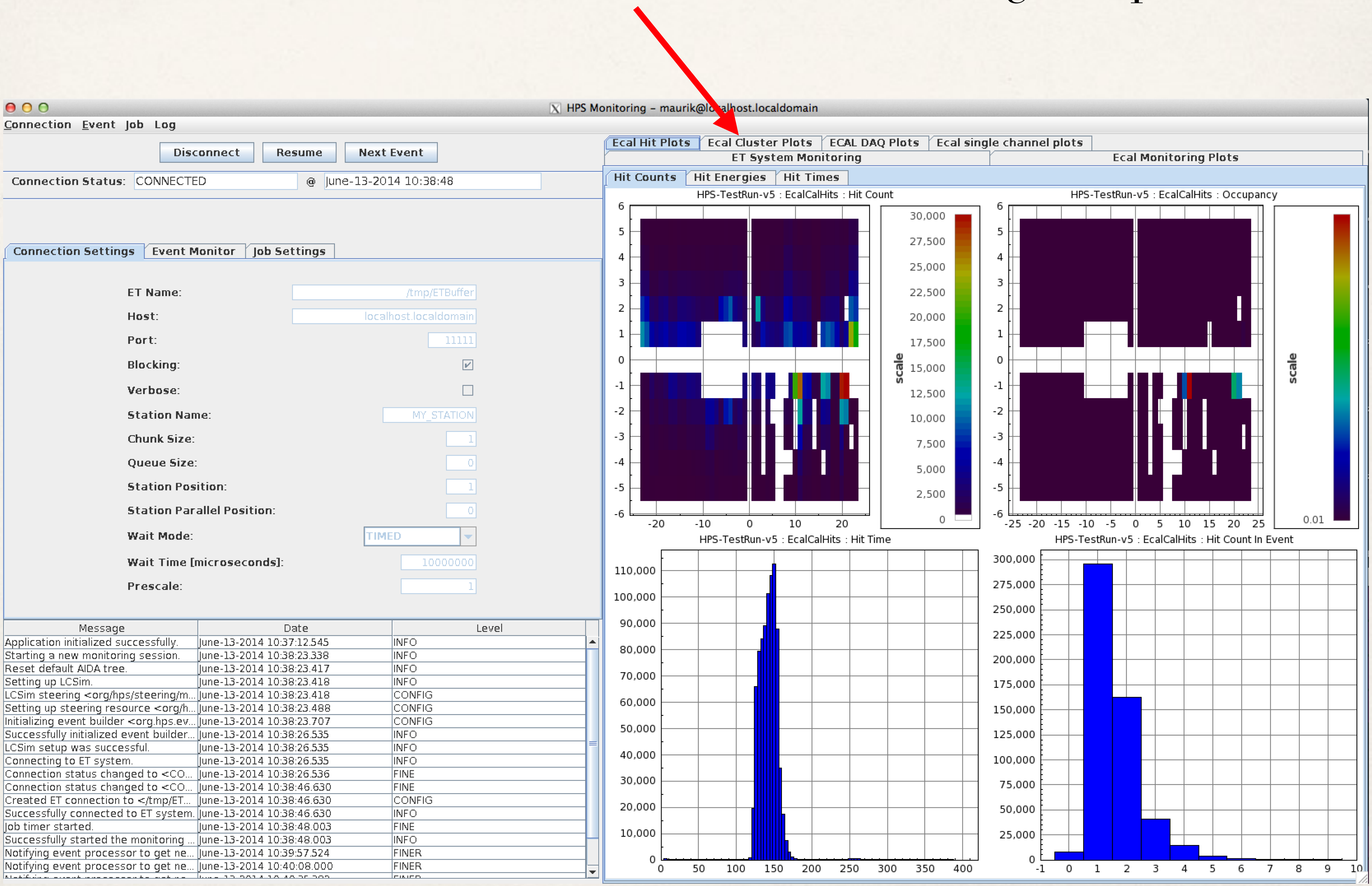
Monitoring Application

- Monitoring application written in JAVA
- Provides flexible platform to display live updating histograms
- Reads data from ET in EVIO format, converts internally to LCIO (data analysis framework file format)
- Can run multiple copies, i.e. separate ones for different sub-systems.
- Histograms can be saved at end of run.
- Reasonable data rate: ~ 100 Hz (for complicated ECAL monitor)
- Full reconstruction framework available to app to make high level plots.



Monitoring Application

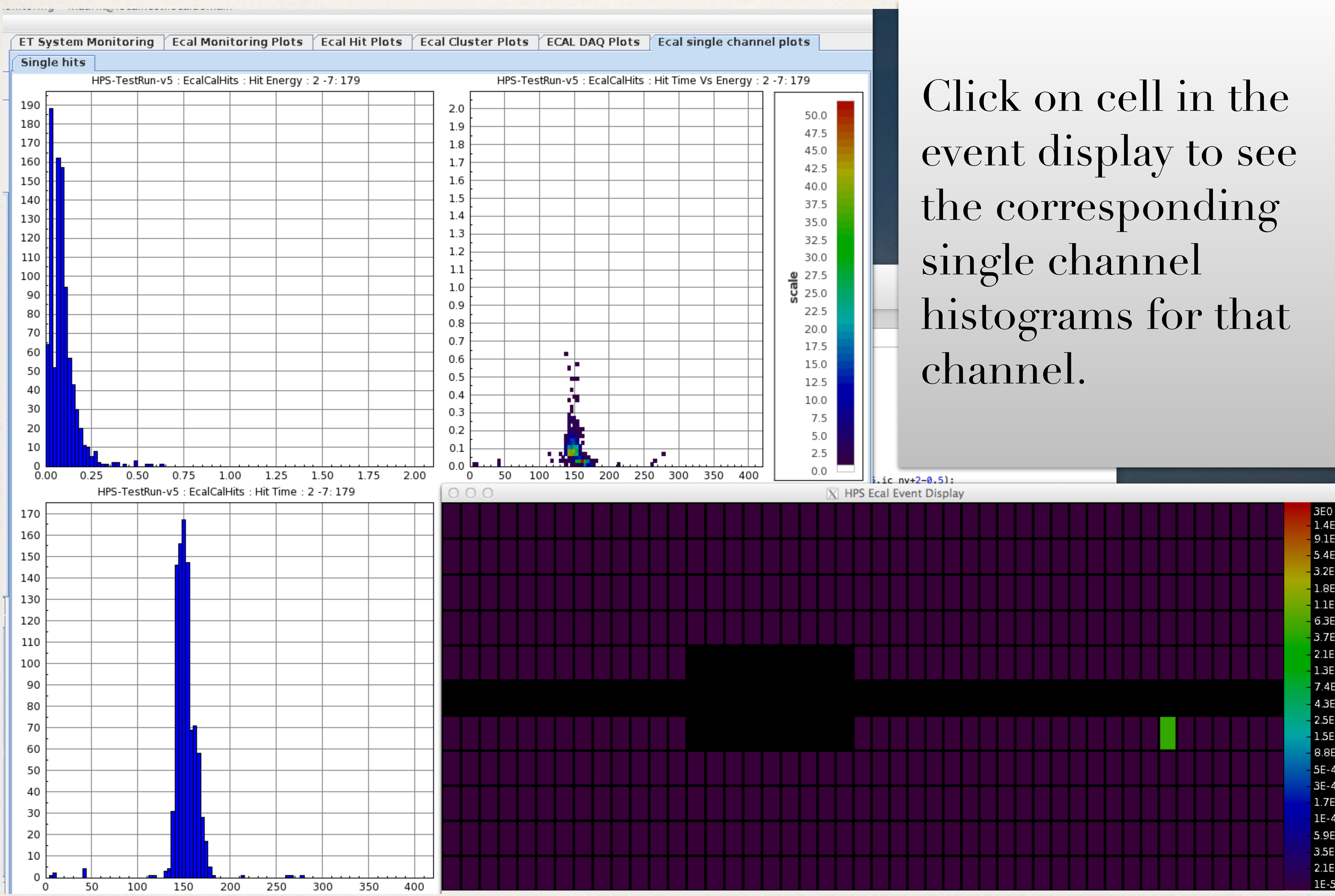
Tabs allow shift taker to select various histogram panels.



Monitoring: ECAL Event Display

- Shows the hits in the ECAL, color coding indicates energy of hit.
- Red box (not shown) indicates a found cluster.
- Text box below shows values of cell that was hovered over.
- Similar display can be added for hodoscope.

Single channel histograms.



Click on cell in the event display to see the corresponding single channel histograms for that channel.

Resource use, CPU

- CPU requirements for main data processing step.
  - 2017 - May - 165 ms/event/core
  - 2018 - Nov - 70 ms/event/core ⇒
    - 500 simultaneous job slots ≈ 7 KHz data analysis (2 - 3x slower than data taking)

      ⇒ **About 13 weeks to process 2019 data.**

      Actual time will depend on number of job slots.
  - Profile: http://nuclear.unh.edu/HPS/Profiles/Call Tree doProcess 2018 12 11.xml

```
Tree: Call Tree
▼ calls: 4827, local time:NaN, total time: 1,663,533.008 ms , 50.30 % -- org.hps.recon.tracking.TrackerReconDriver.process
  ▼ calls: 4827, local time:NaN, total time: 1,663,406.703 ms , 50.30 % -- org.lcsim.util.Driver.process
    ▼ calls: 4827, local time:NaN, total time: 1,663,402.222 ms , 50.30 % -- org.lcsim.util.Driver.processChildren
      ▼ calls: 4827, local time:NaN, total time: 1,663,391.957 ms , 50.30 % -- org.lcsim.util.Driver.doProcess
        ▶ calls: 4827, local time:NaN, total time: 1,663,382.259 ms , 50.30 % -- org.hps.recon.tracking.SeedTracker.process
          calls: 9653, local time:NaN, total time: 2.511 ms , .00 % -- java.lang.System.nanoTime
        calls: 9653, local time:NaN, total time: 1.784 ms , .00 % -- java.util.Iterator.hasNext
        calls: 4827, local time:NaN, total time: .789 ms , .00 % -- java.util.List.iterator
        calls: 4827, local time:NaN, total time: .708 ms , .00 % -- java.util.Iterator.next
  ▶ calls: 21254, local time:NaN, total time: 38.275 ms , .00 % -- hep.physics.vec.VecOp.sub
  ▶ calls: 9652, local time:NaN, total time: 10.848 ms , .00 % -- org.lcsim.event.base.BaseLCSimEvent.get
  ▶ calls: 4826, local time:NaN, total time: 10.659 ms , .00 % -- org.hps.recon.tracking.TrackerReconDriver.setTrackType
  ▶ calls: 21254, local time:NaN, total time: 6.257 ms , .00 % -- hep.physics.vec.BasicHep3Vector.magnitude
    calls: 33688, local time:NaN, total time: 3.698 ms , .00 % -- java.util.Iterator.hasNext
    calls: 25058, local time:NaN, total time: 2.749 ms , .00 % -- java.util.Iterator.next
    calls: 21254, local time:NaN, total time: 2.389 ms , .00 % -- hep.physics.vec.BasicHep3Vector.<init>
    calls: 21254, local time:NaN, total time: 1.971 ms , .00 % -- org.lcsim.fit.helicaltrack.HelicalTrackHit.getCorrectedPosition
    calls: 21254, local time:NaN, total time: 1.921 ms , .00 % -- org.lcsim.fit.helicaltrack.HelicalTrackHit.getPosition
    calls: 21254, local time:NaN, total time: 1.891 ms , .00 % -- org.lcsim.fit.helicaltrack.HelicalTrackHit.chisq
    calls: 8630, local time:NaN, total time: 1.333 ms , .00 % -- java.util.List.iterator
    calls: 4826, local time:NaN, total time: .887 ms , .00 % -- java.util.List.size
    calls: 3804, local time:NaN, total time: .432 ms , .00 % -- org.lcsim.event.base.BaseTrack.getTrackerHits
▶ calls: 1207, local time:NaN, total time: 692,632.061 ms , 21.00 % -- org.hps.recon.ecal.EcalRawConverter2Driver.process
▶ calls: 1207, local time:NaN, total time: 483,325.585 ms , 14.60 % -- org.hps.recon.tracking.RawTrackerHitFitterDriver.process
▶ calls: 1206, local time:NaN, total time: 259,630.054 ms , 7.90 % -- org.hps.recon.tracking.gbl.GBLRefitterDriver.process
▶ calls: 1207, local time:NaN, total time: 87,474.758 ms , 2.60 % -- org.hps.recon.tracking.DataTrackerHitDriver.process
▶ calls: 1207, local time:NaN, total time: 36,481.321 ms , 1.10 % -- org.hps.recon.tracking.HelicalTrackHitDriver.process
```

Resource use, CPU - Monte Carlo

- Throughput for MC production is much harder to assess due to the many different steps involved.
- The A' "signal" MC events are quick to produce.
- Most expensive is full background simulation:
  Wide Angle Bremsstrahlung + Trident + Beam background.
  - A useful input event sample, 100M events, requires:
    - 100M Trident events from MadGraph ("tritrig") ≈ 10k core-hours
    - Proportional amount of WAB, MadGraph ≈ 15k core-hours
    - Beam background generation ≈ 2k core-hours
  - Detector simulation ≈ 7 ms/event, 1 hour per file, 500k mixed events.
    10k files ≈ 10k core-hours for full run.
  - Reconstruction of simulated data 10 to 1, so 1000 jobs, 3h each ≈ 3k core-hours
  - Total CPU for 100M event run is ≈ 40 k core-hours ⇒ ≈ **500 jobs for 4 days.**
- Will ultimately want to run 10-20 times more events.
- Monte Carlo jobs will also run on off-site farms: SLAC & UNH.
- Investigating the use of Open Science Grid.

Resource use, disk

- Estimated disk space usage:
  - For 2016 engineering run raw data were 2 GB
    - File contains ≈ 407k events, takes 20s to 30s to write.
    - Processed reconstruction file has 396k events.
    - Space for reconstructed event file + all DSTs = 7 GB
- Estimated Space for 2019 run
  - 9 Weeks, at 50% efficiency = 756h = 2.7 M sec.
  - At 20 kHz, we expect ≈ **54B events**.
  - Raw data storage expected: 260 TB.
  - Processed data storage expected: 910 TB
    - The **full DST** only would take **65 TB**.
    - The **V0 DST, pre-selected trident events, takes 3.3 TB** — fits on single hard disk!

One run file:

	Size [Mb]	# of events
Raw data	2048	407500
recon	6100	395930
dst	521	395930
v0_dst	26	10852
pulser_dst	6.8	14036
Moeller_ds	14	6396
nt_tri	26	10962
nt_Moeller	9.1	4350
v0	219	10852
pulser	159	14036
Moeller	124	6396
Total	7205	

- MC, 100M events simulation output ≈ 8 TB - most of this is intermediate files.
  - Reconstruction of simulated output is only about 1% if input, because of acceptance and background rejection ≈ 81 GB, but we probably want to simulate 10 to 20x more.

Software Task List

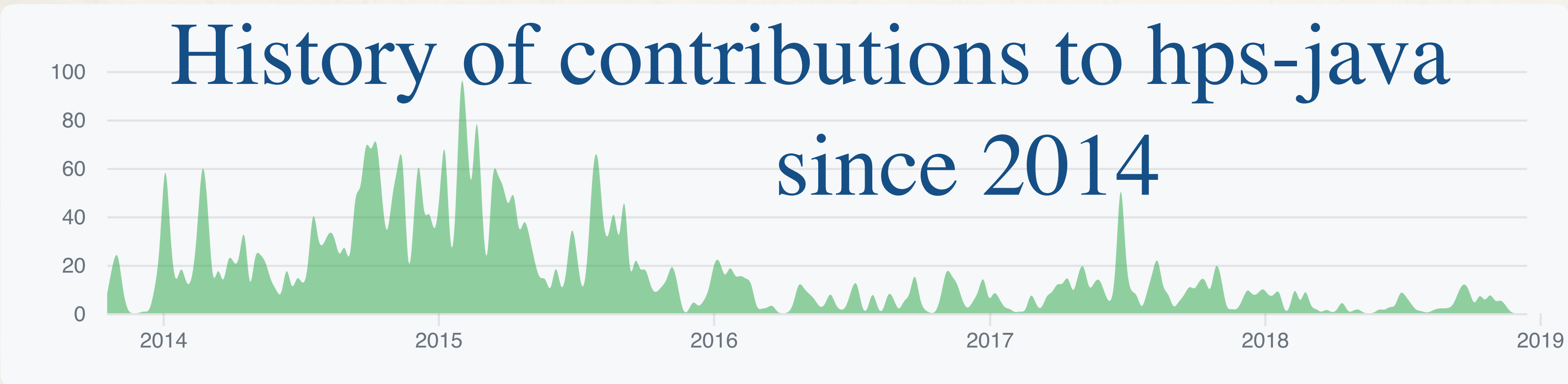
Mostly, our software is in reasonably good shape, but many improvement are desirable: directly related to 2019 running, smoothing operations, speeding up processing.

- **Very Important (critical) Tasks for 2019 run:**
  - Complete Hodoscope simulation and new trigger optimization analysis.
    - Extensive task which is already well underway. See Rafayel Paremuzyan's talk.
  - Add FADC bit-packed data decoder to hps-java.
    - Already exists for CLAS12, so not expected to be too complicated or time consuming.
  - Update monitoring histograms.
    - Needs hodoscope and L0 histograms added.
    - Cleaning up, revisit, existing histograms.
  - Improve/update data quality monitoring.
    - Update for hodoscope and L0.

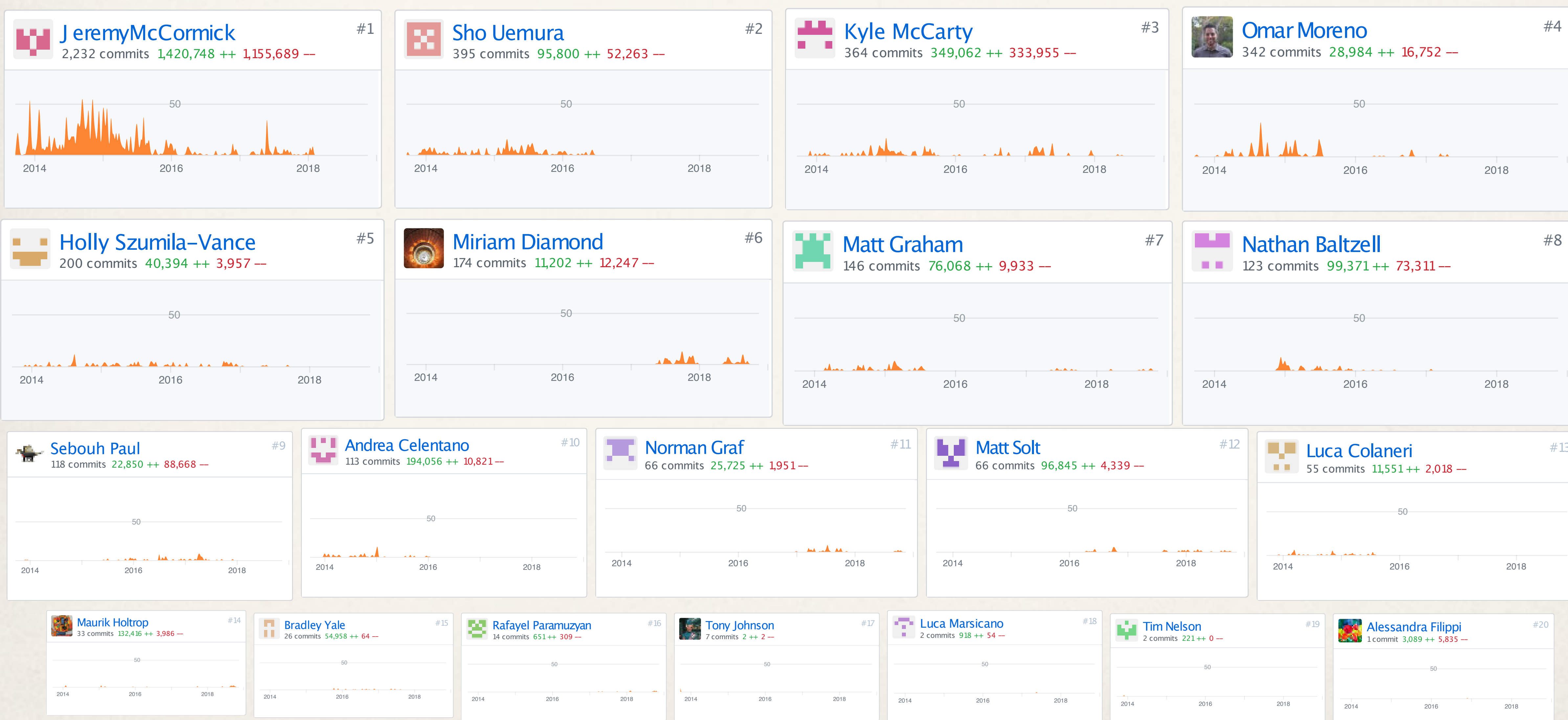
Software Task List

- **Important Tasks, highly desirable:**
  - Improve the alignment procedures.
    - We need the detector alignment to be easier so results can be obtained more quickly.
    - Procedures have already improved tremendously, now to make it easier to use.
- **Other Important Tasks:**
  - Revisit all other calibrations and see where updates are needed.
    - It has been a little while since we last needed a full calibration.
  - Improve processing speed of the code. Not essential, but makes us good citizens of JLab farm.
    - Further improve the speed of the tracking code.
    - Complete the investigation of alternate tracking: Kalman filter and different seed finder.
    - Possibly: preprocess the FADC and SVT pulse fits.
  - Learn to use the Open Science Grid for simulation.
  - Biassing MC events in the hps-sim simulation to speed up MC.
    - Specifically WAB production can be sped up a lot.
- Lots of minor issues, code maintenance, and code improvements.

People: Software contributions

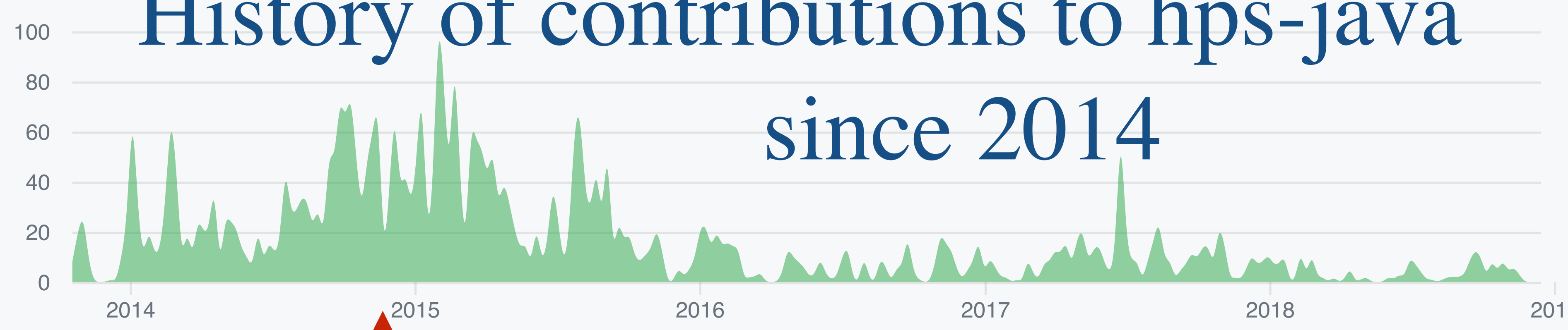


Caveat:
Number of lines or number of commits does not linearly correspond to software productivity

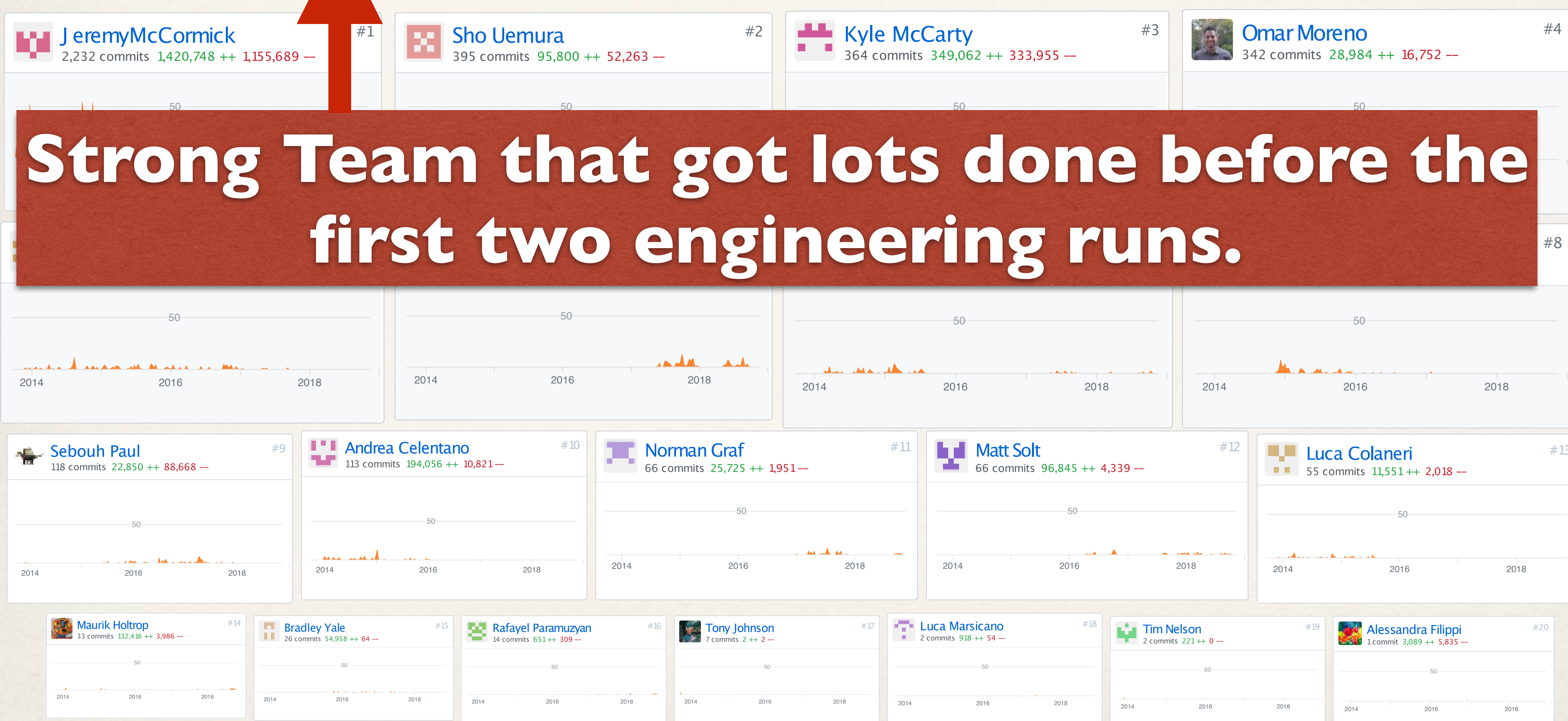


People: Software contributions

History of contributions to hps-java since 2014

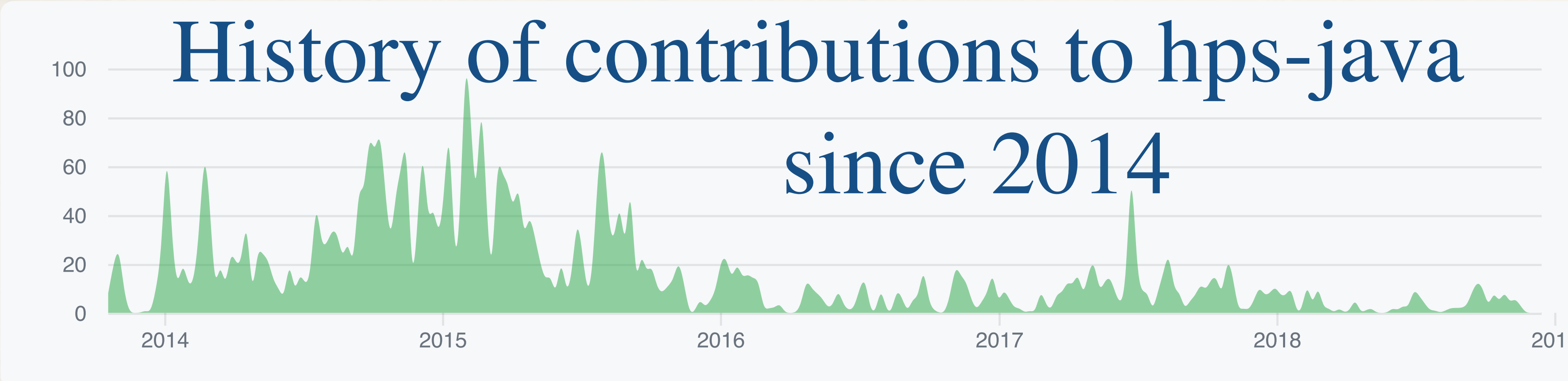


Caveat:
Number of lines
or number of commits
does not linearly
correspond to software
productivity

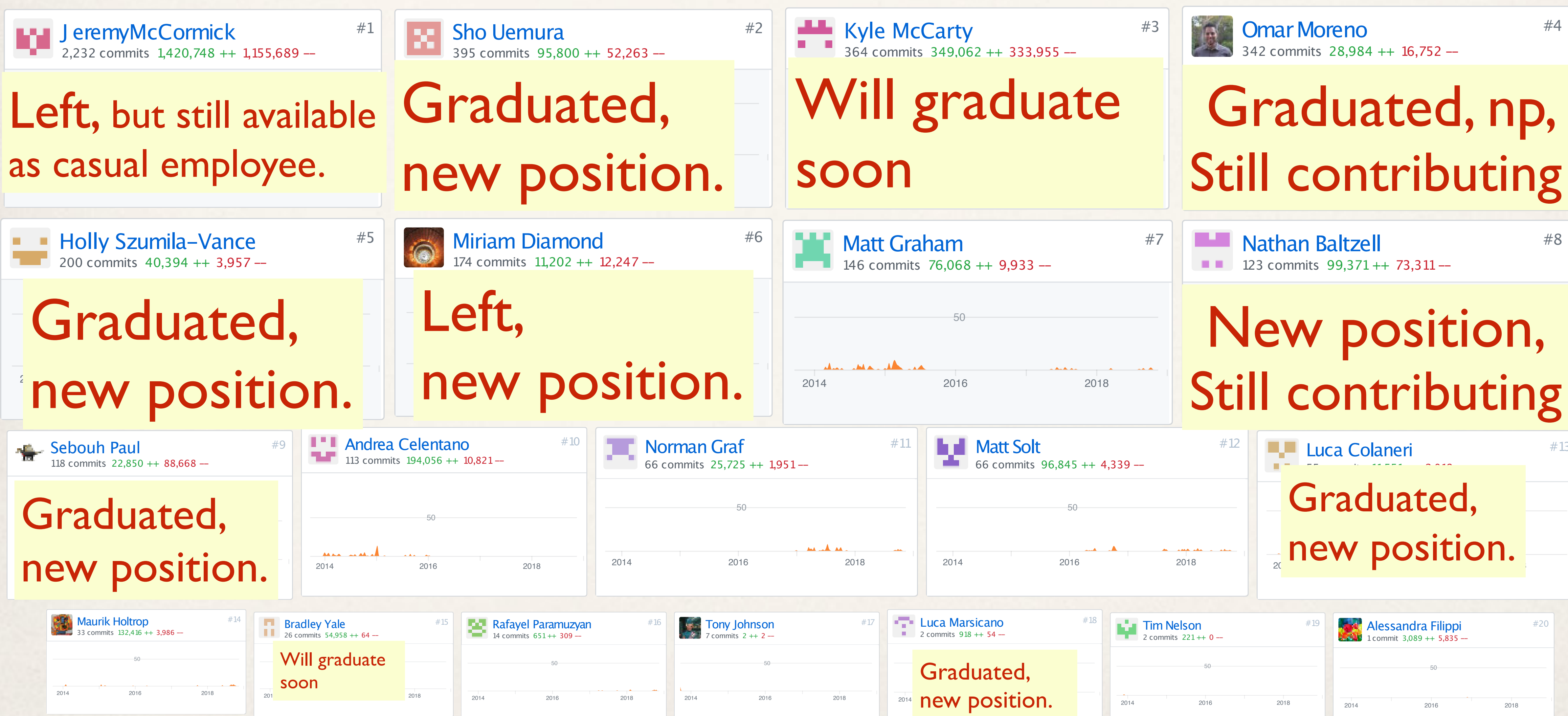


Strong Team that got lots done before the first two engineering runs.

People: Software contributions



Not too surprising, there is a fair bit of turnover in the contributors to the software.
This is mostly success.
Students graduate,
Postdocs find permanent jobs.



Software contributions

- **New people joining software team:**
  - Cameron Bravo (SLAC), once SVT L0 work is finished.
  - New Postdoc (UNH), advertisement is out.
  - New Postdoc (SLAC)
  - New students

- Total number of software tasks, and amount of effort required, is now lower than 4 years ago, just before the 2015 engineering run.
- Fewer tasks are critical.
- **With the new people, there is enough manpower to continue a strong software group.**

Conclusions

- **The HPS software is fully functioning.**
  - Data analysis of 2015 engineering run is complete and published.
  - Data analysis of 2016 engineering run is well under way.
  - A lot was learned by the collaboration to get there.
- Some updates are needed for the new detector.
  - No show stoppers.
  - Amount of work to be performed is manageable with new people coming on board.
- Procedures are becoming standardized and more streamlined.
  - Data processing will be faster, requiring far fewer iterations.
  - Analysis path is now clear and becoming standardized. (See Nathan's talk)
- **HPS is (nearly) ready to process the 2019 data.**