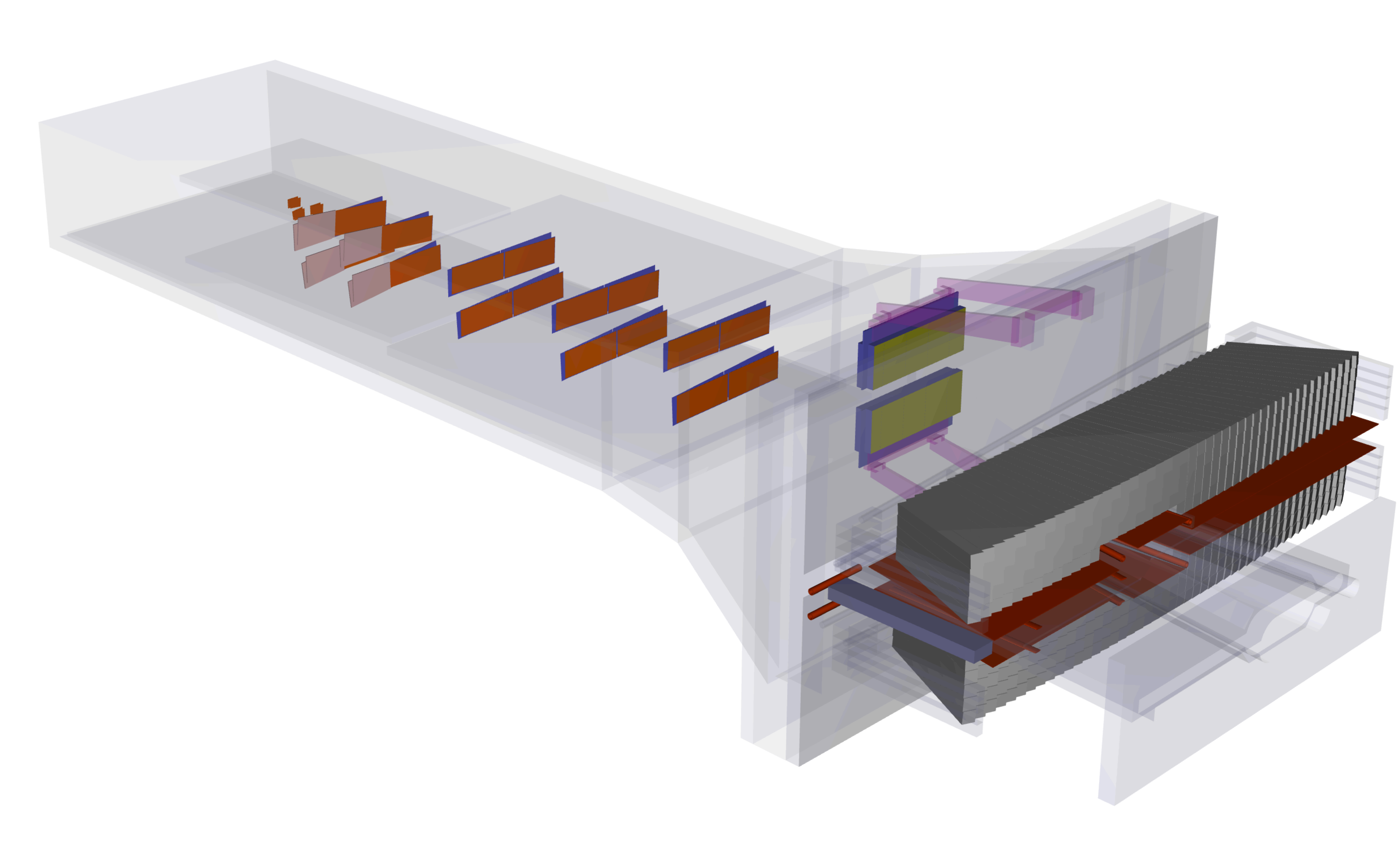


HPS Software

Presentation for DOE Review at SLAC

January 18, 2019

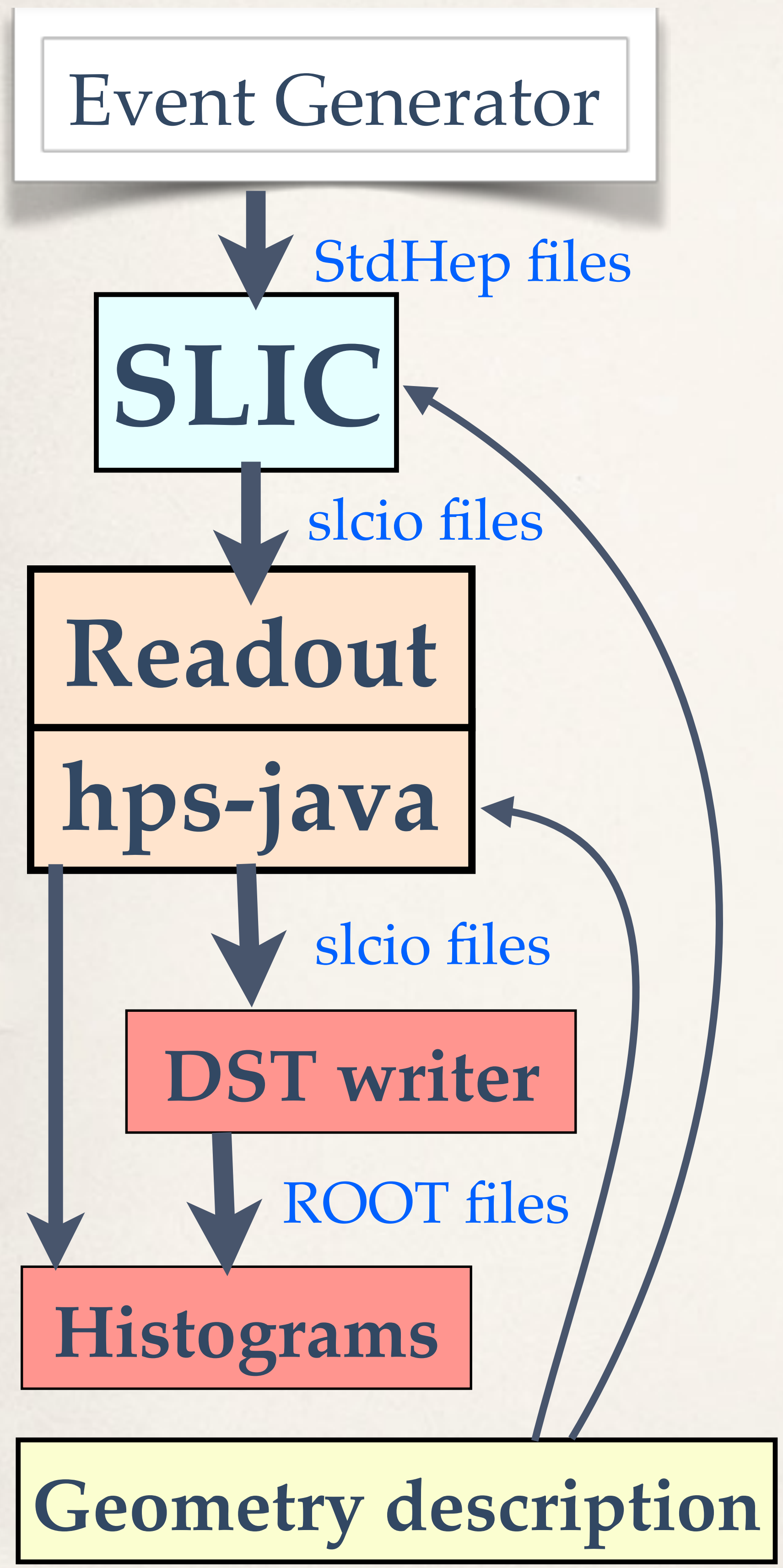
Overview

- Introduction
  - History
  - System Overview
  - Software Organization Overview
  - Software Group
- Outstanding Task List
- Historic and projected manpower
- System resource utilization
- 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 simulation.

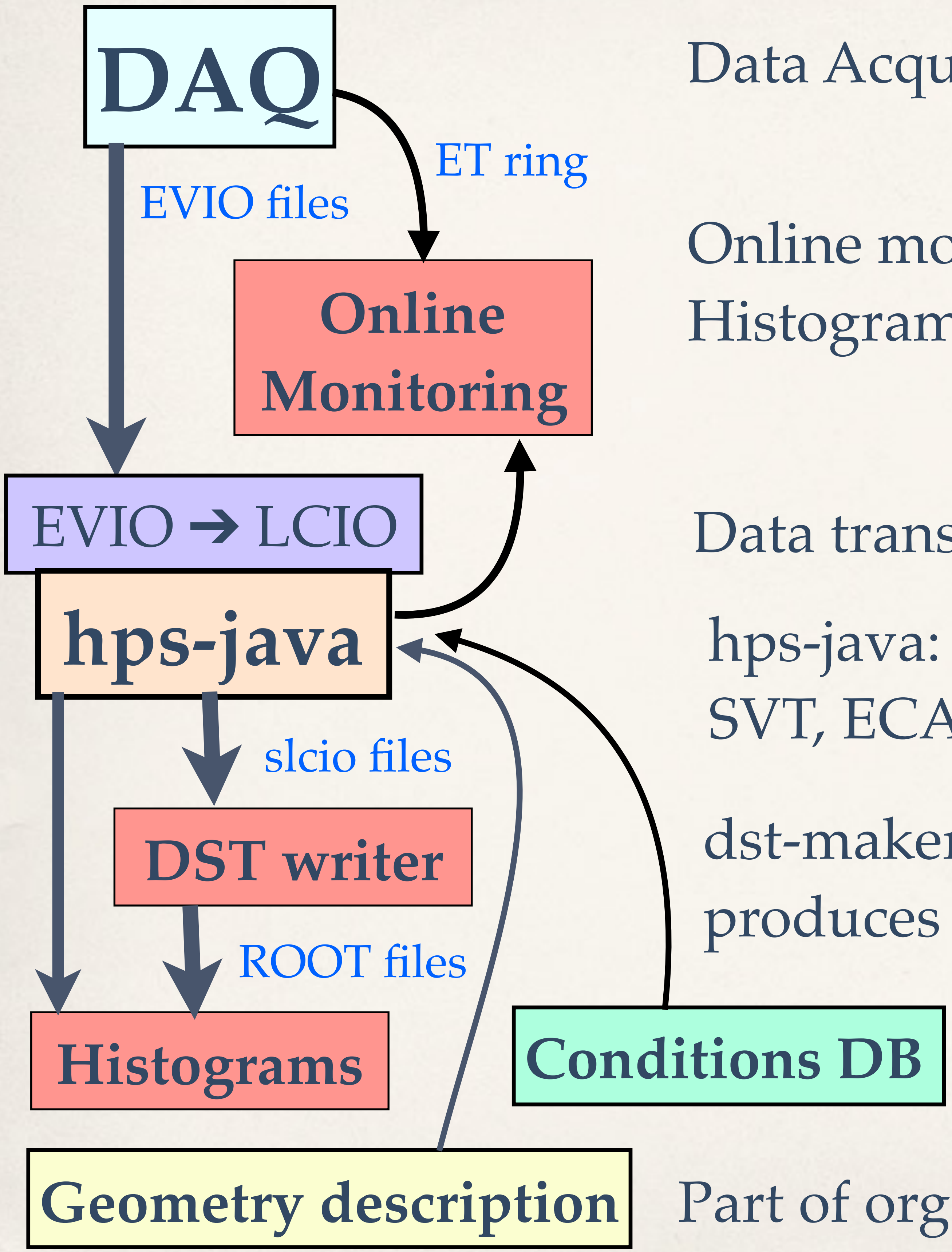
Readout: Simulates electronics and trigger.

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.

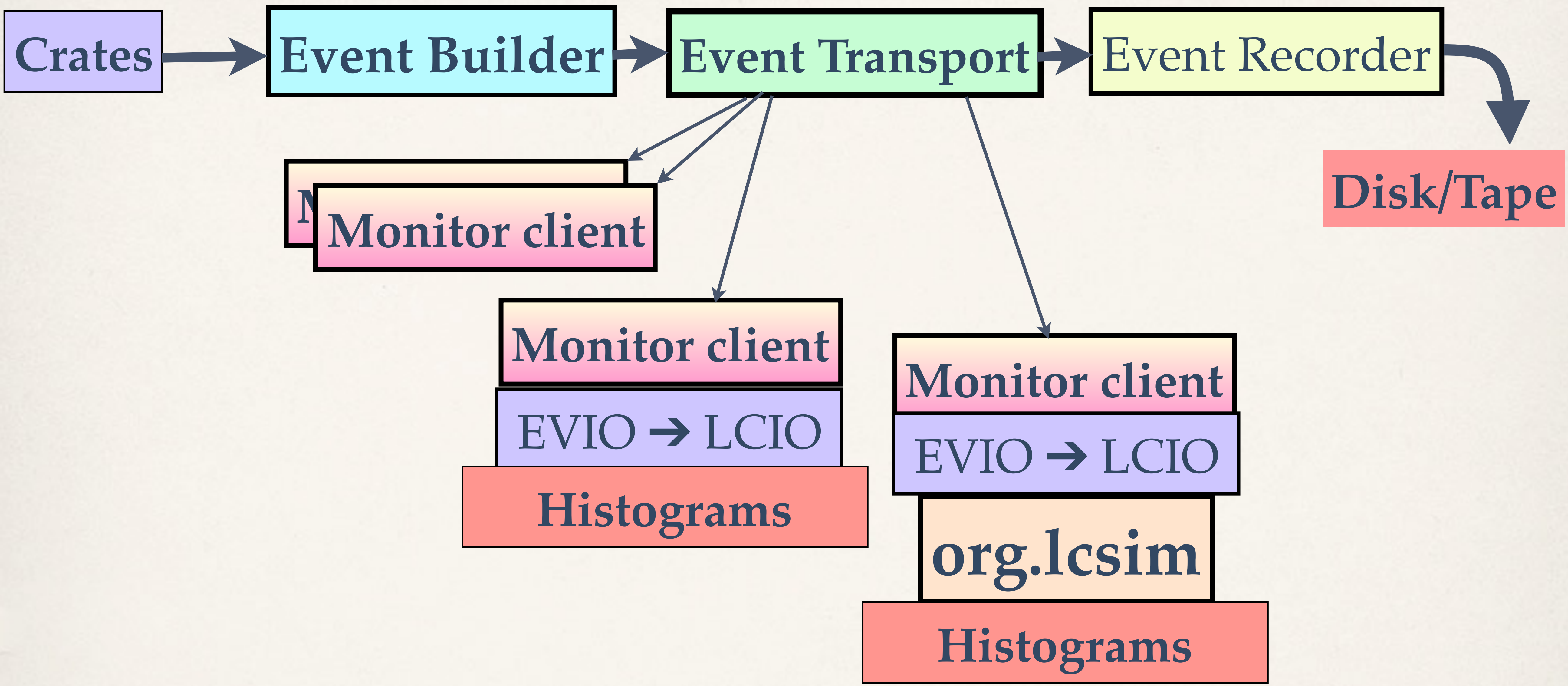
hps-java: Analysis framework:
SVT, ECAL, Tracking, Hodoscope

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:
  - 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 and gains.
  - 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
- Releases: - Github + Maven + Nexus.
  - Release is tagged on GitHub.
  - Resulting JAR file is available for download from Nexus.

Introduction - Software Group

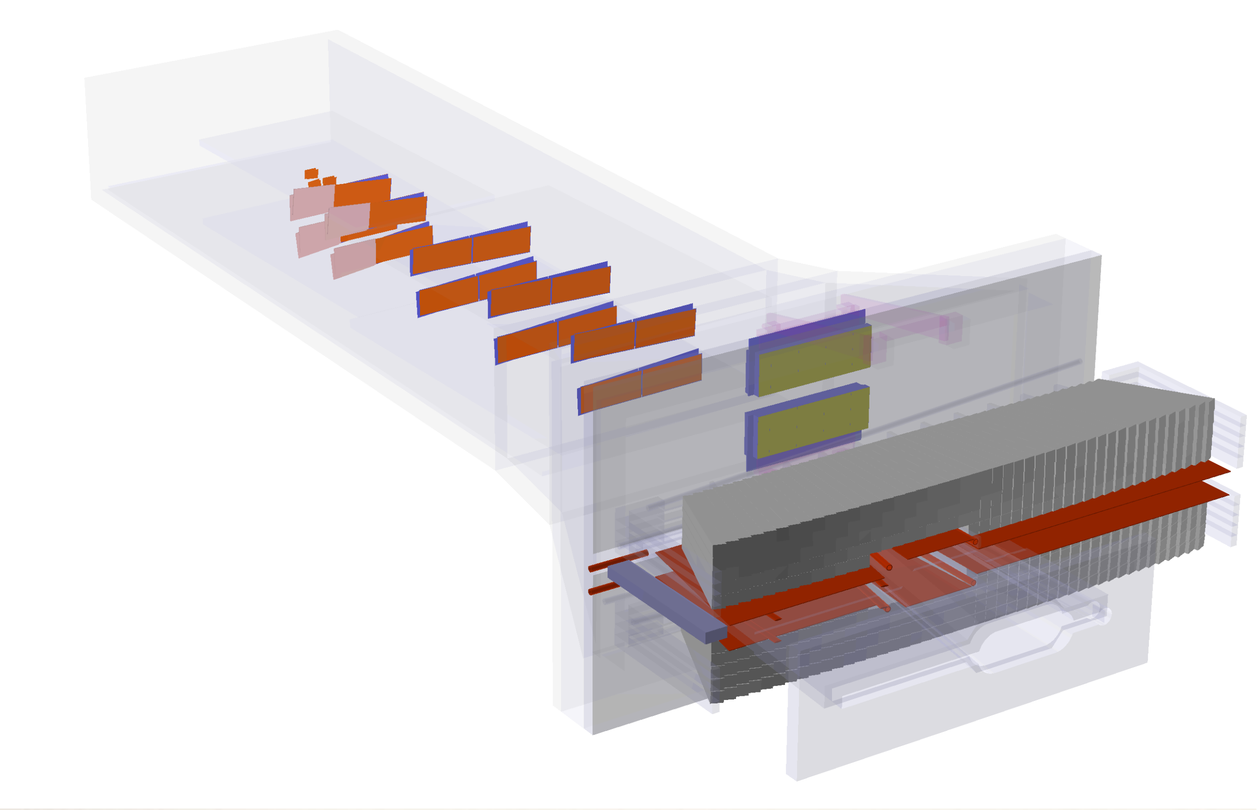
- Bi- Weekly meetings with online presentations.
- Software group mailing list
- SLACK for more immediate communication.

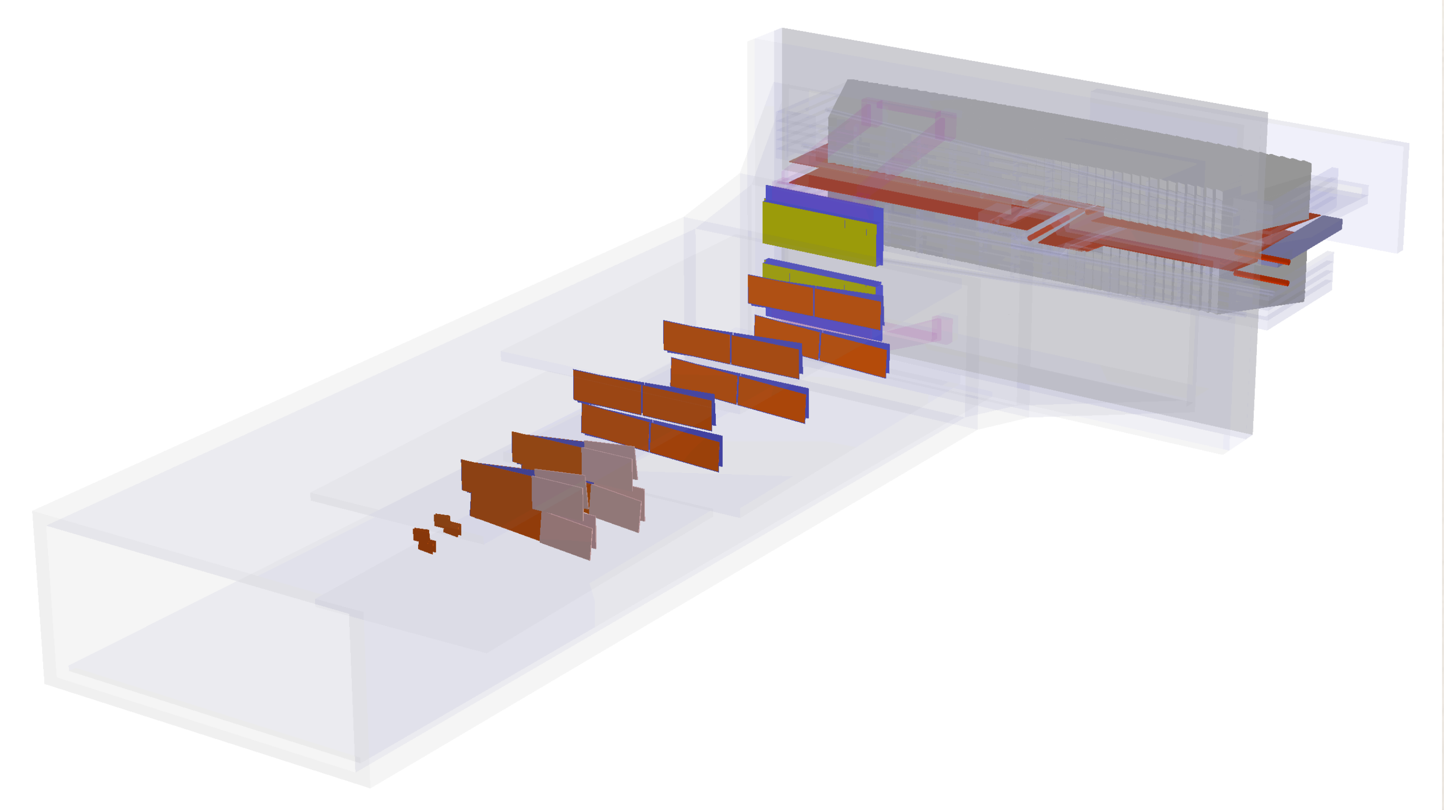
- Lead: Maurik Holtrop
  - Tracking lead: Norman Graf
  - MC Generators: Takashi Maruyama
    - MC data production: Bradley Yale
  - Trigger: Valeri Kubarovsky
    - Trigger code: Kyle McCarty
  - Data Processing: Rafayel Paremuzyan
  - Analysis software: Matt Graham, Nathan Baltzell
  - Specific codes:
    - 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 into a 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!
  - After the detector simulation, the "2 ns events" are combined, and in the readout step, a trigger is searched for similar to the hardware trigger.
  - Events for which a trigger is found are further analyzed.
  - Generated events are 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.
- All active components are accurately rendered.
- Most of the inactive components that could interact with particles are accurately rendered.

Tracking

Alignment

Monitoring

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 ≈ 10k core-hours
    - Proportional amount of WAB, MadGraph ≈ 10k core-hours
    - Beam background generation ≈ 10k core-hours
  - Detector simulation ≈ 7 ms/event, 1 hour per file,
    10k files ≈ 10k core-hours for full run.
  - Reconstruction of simulated data ≈
  - Total CPU for 100M event run is ≈ 45 k core-hours ⇒ 500 jobs for 4 days.

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
- 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 DST only would take 65 TB.

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.
  - Reconstruction of simulated output is only about 1% if input, because of acceptance and background rejection.

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.

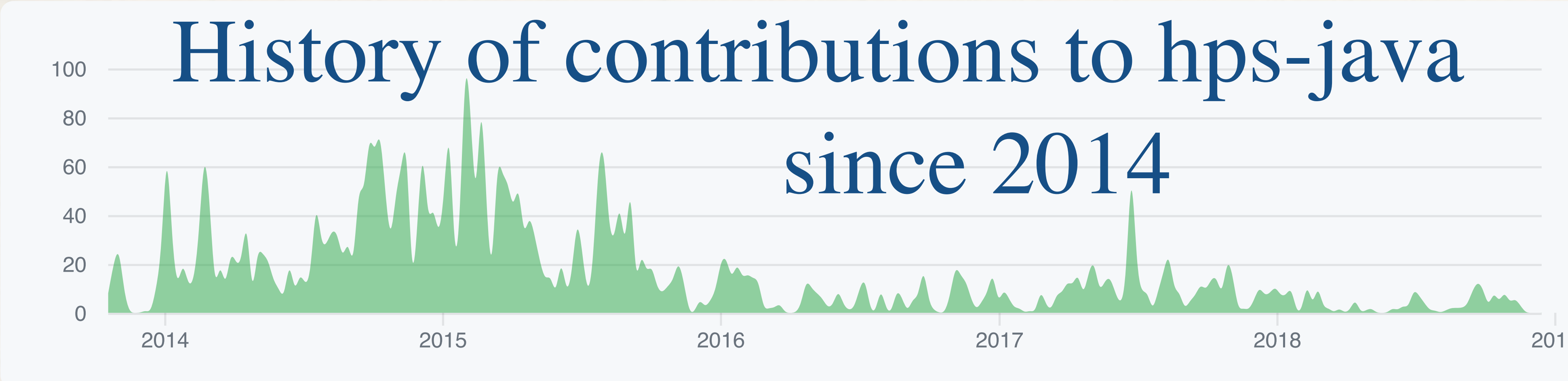
- **Very Important (critical) Tasks for 2019 run:**
  - Complete Hodoscope simulation and new trigger optimization analysis.
    - Extensive task which is already well underway. See Rafayel Paremuzyan's talk.
  - Add FADC bit-packed data decoder to hps-java.
    - Already exists for CLAS12, so not expected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

Software Task List

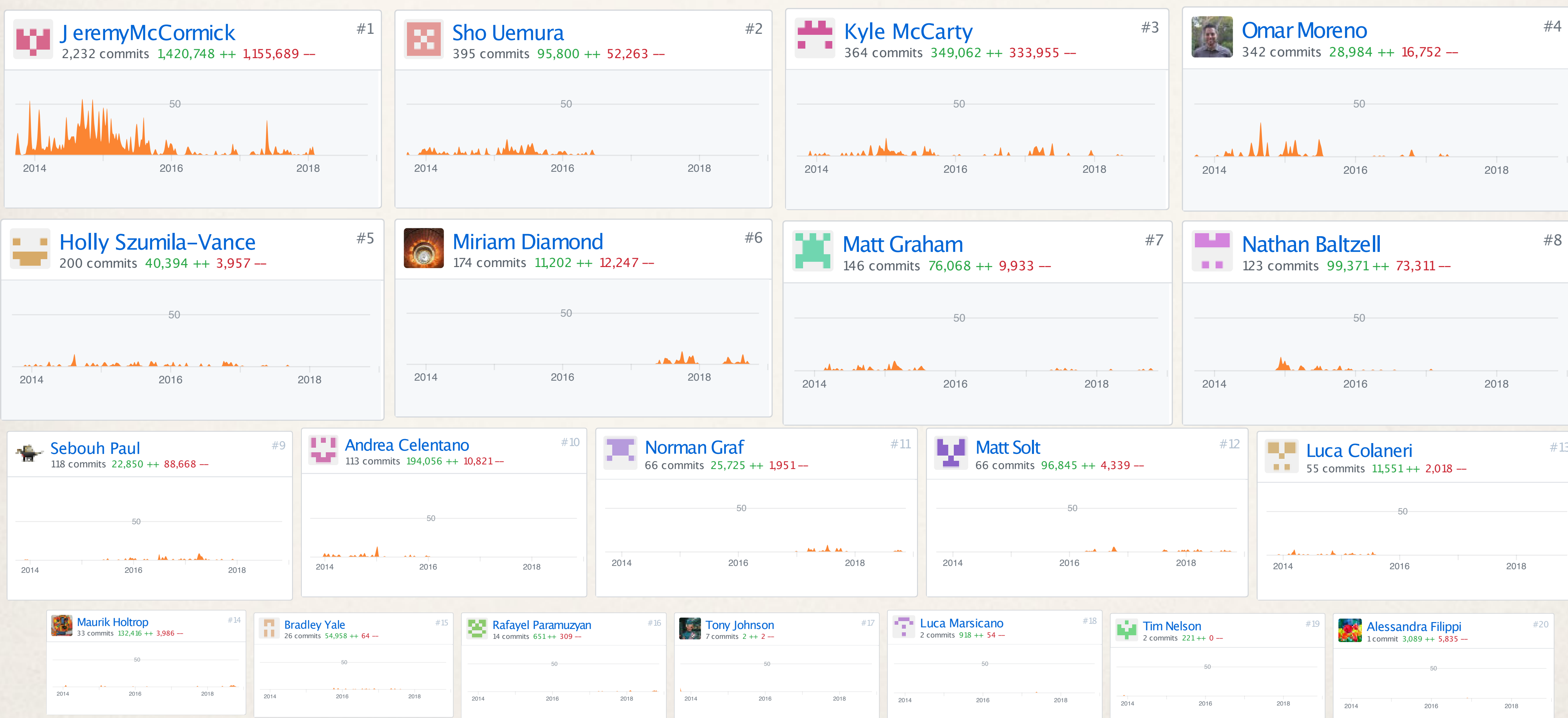
- **Important Tasks, highly desirable:**
  - Improve the alignment procedures.
    - We need to get detector alignment to be easier so results can be obtained more quickly.
- **Other Important Tasks:**
  - Revisit all other calibrations and see where updates are needed.
    - It has been a little while since we last needed a full calibration.
  - Improve processing speed of the code.
    - 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.

  - Lots of minor issues, maintenance issue, code improvements on issues lists.

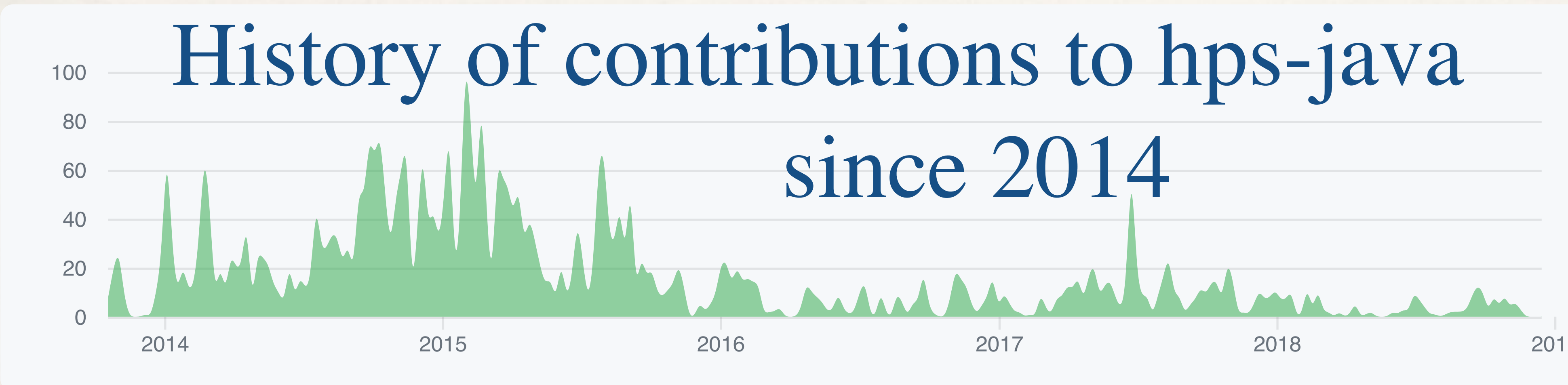
Software contributions



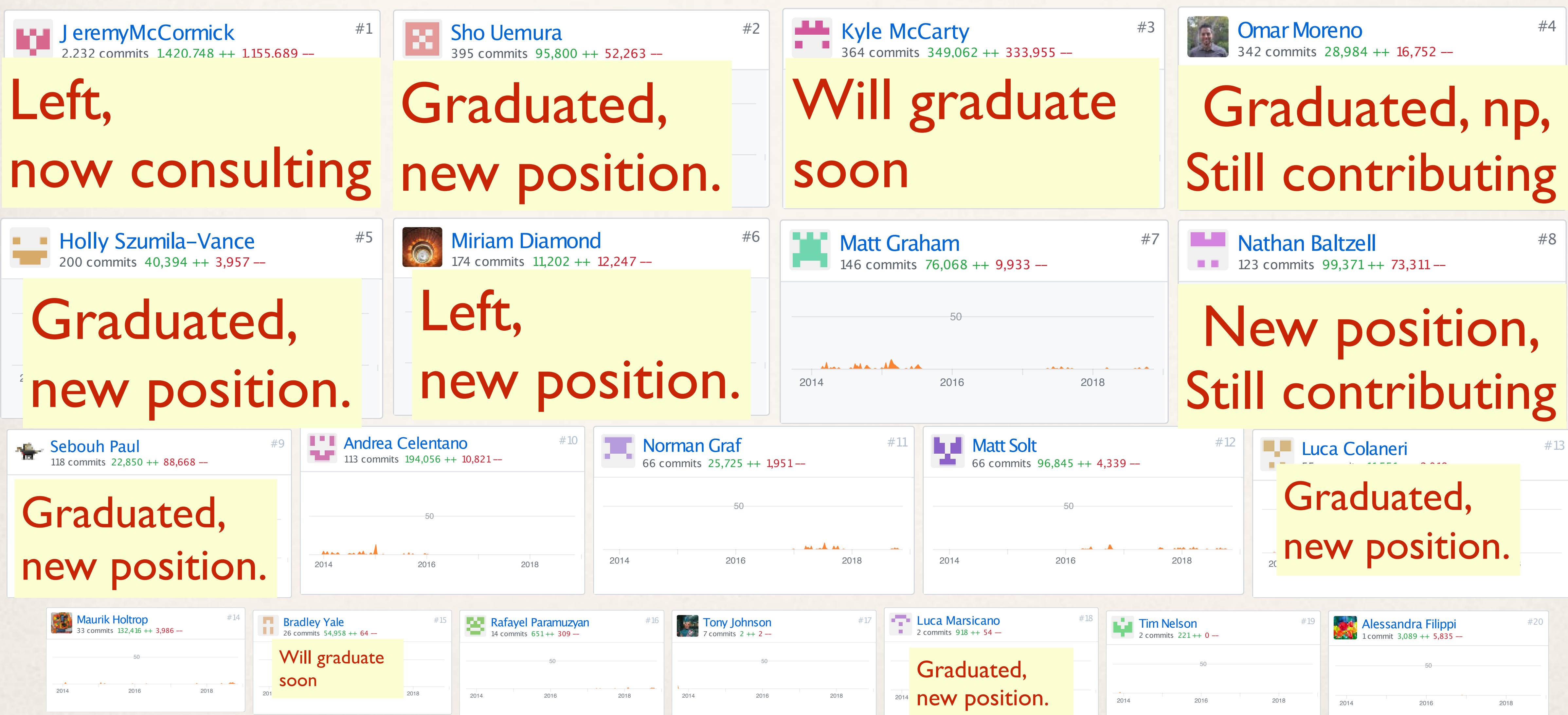
Caveat:
Number of lines or number of commits does not linearly correspond to software productivity



Software contributions



Not too surprising, there is a fair bit of turnover in the contributors to the software.



Software contributions

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

- There is still need for further development on improvements.

Conclusions