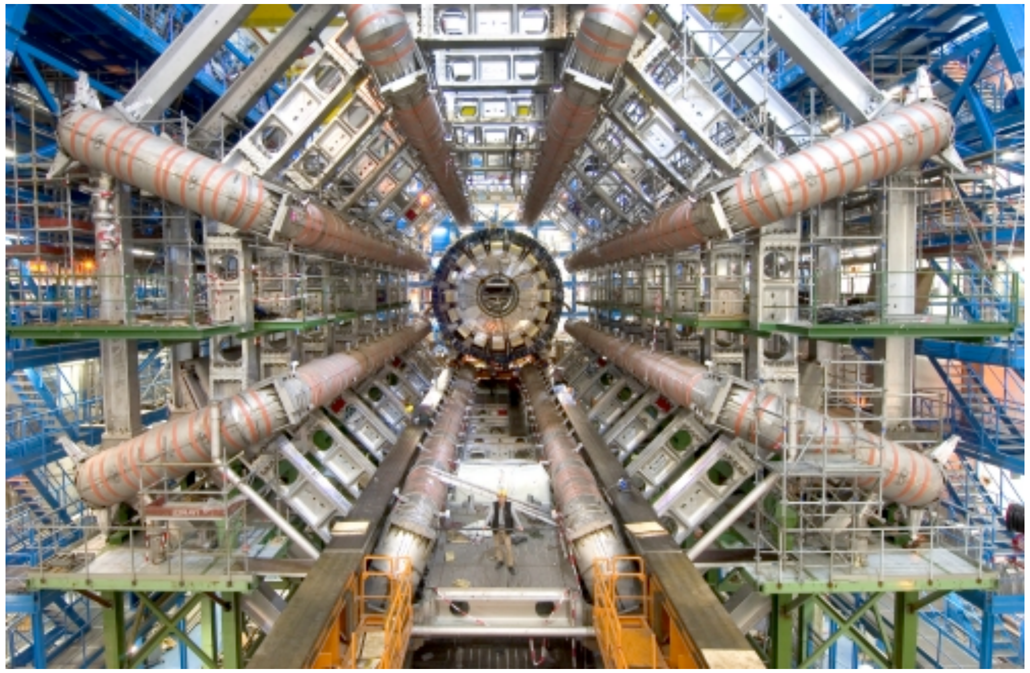
atlas handling of problematic events in quasi real-time

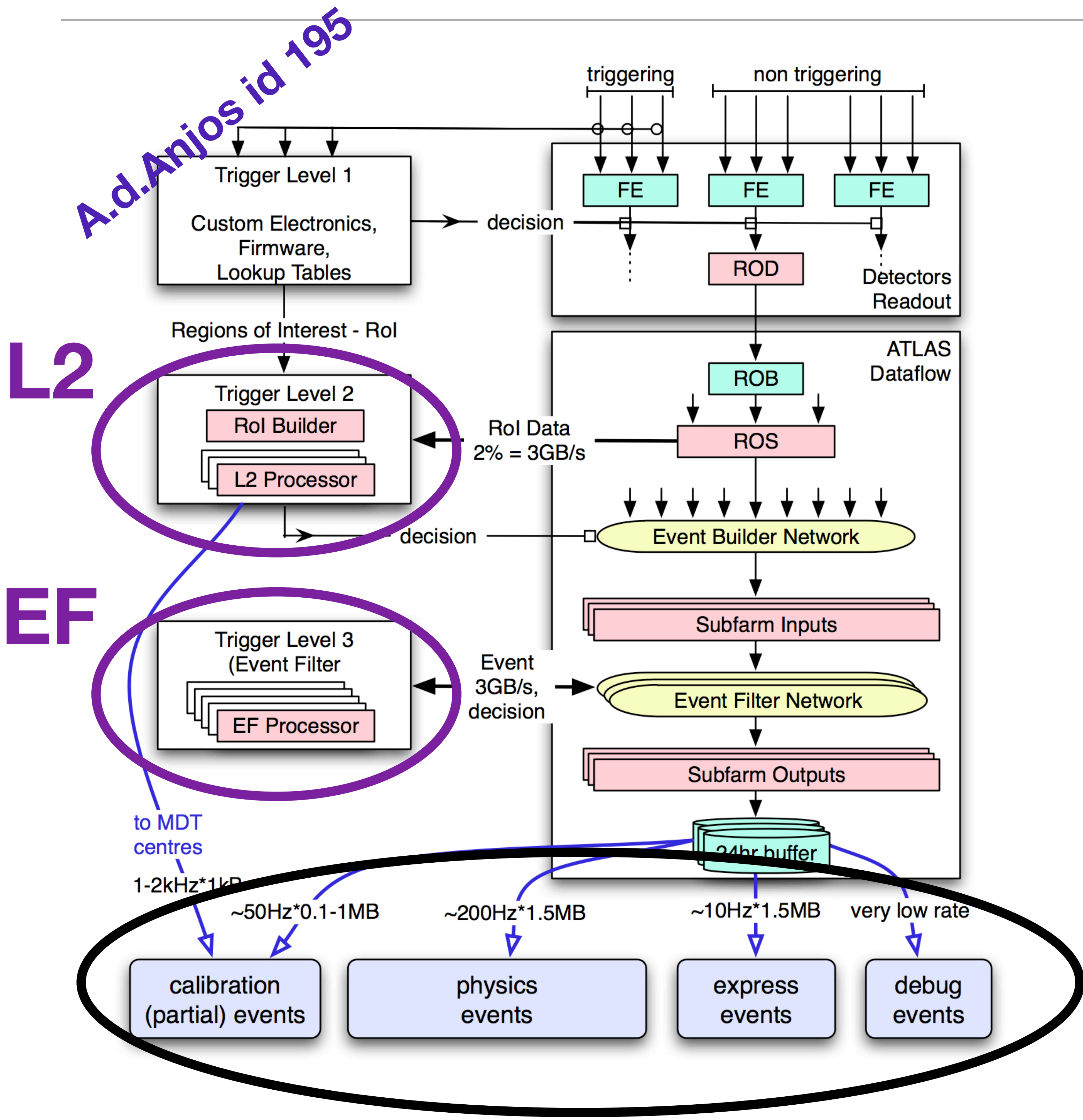
hegoi garitaonandia, sander klous, brian petersen, anna sfyrla

the Giant and its run-time errors

- in http://atlas.ch you can read about LHC/ATLAS:
    - 27 kilometres in circumference, collisions 30 million times a second
    - 45 meters long, 25 meters high, 7000 tons
    - **37 countries, 2500 physicist, 700 students**
- many, many sources of errors in SW
- some only show up at runtime
- the DAQ is the first system to see/suffer them
- **can we just ignore that small fraction?**



DAQ, streaming and the debug stream



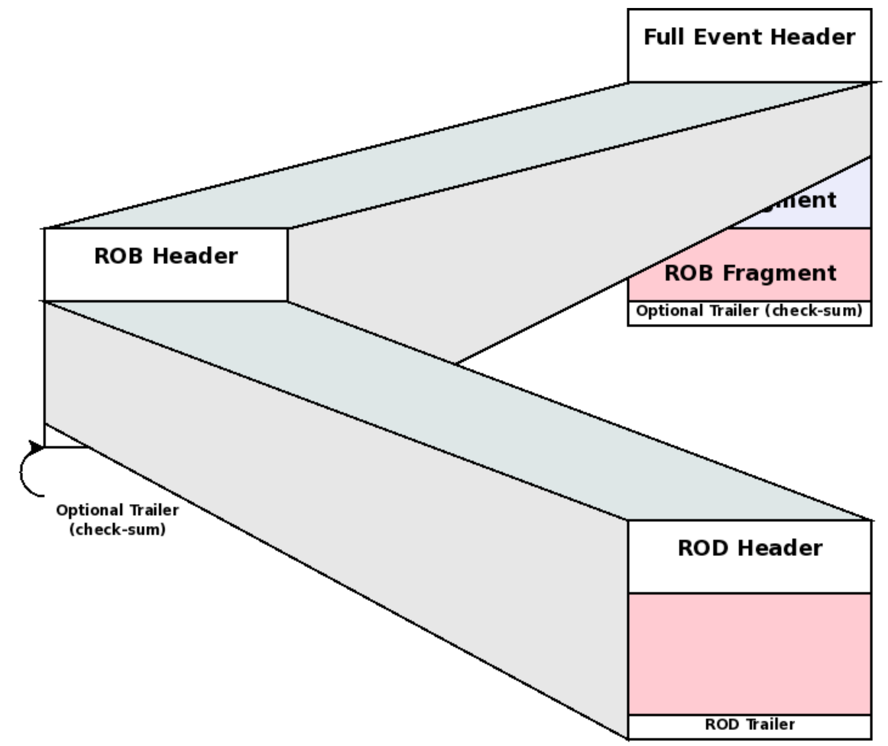
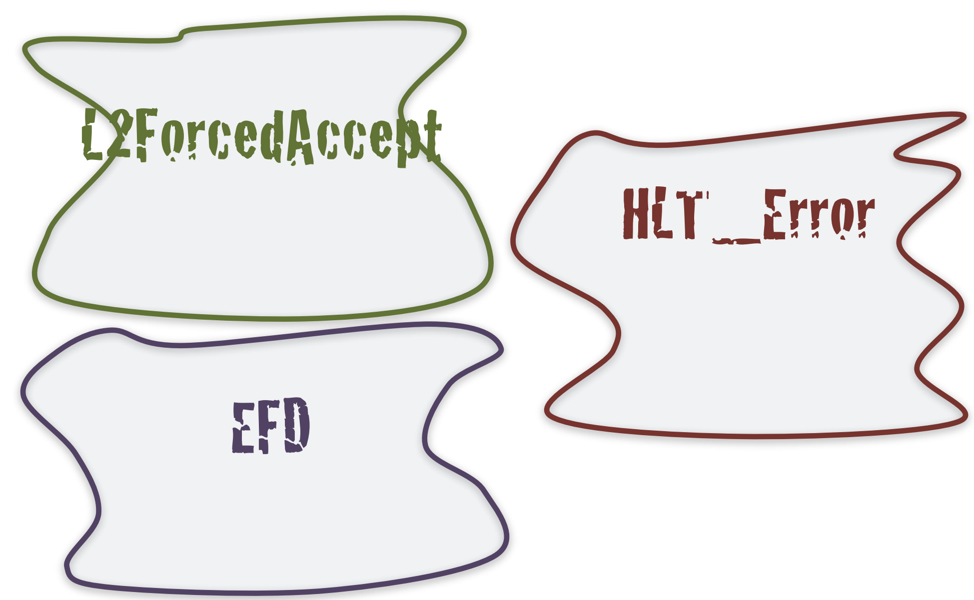


Figure 1: The general event format.



what to do with problematic events after the DAQ?

- understand and correct the errors to increase the quality of the data in following runs
  - need fine grained diagnosis: algorithm, timing info
- reduce the turn arround time for bug fixing, missconfiguration correction, etc
  - feedback to developers and shifters during an ongoning run?
- are timeouts correlated with specific physics processes?
  - place events back into data chain (or redimension DAQ to worst case scenario?)

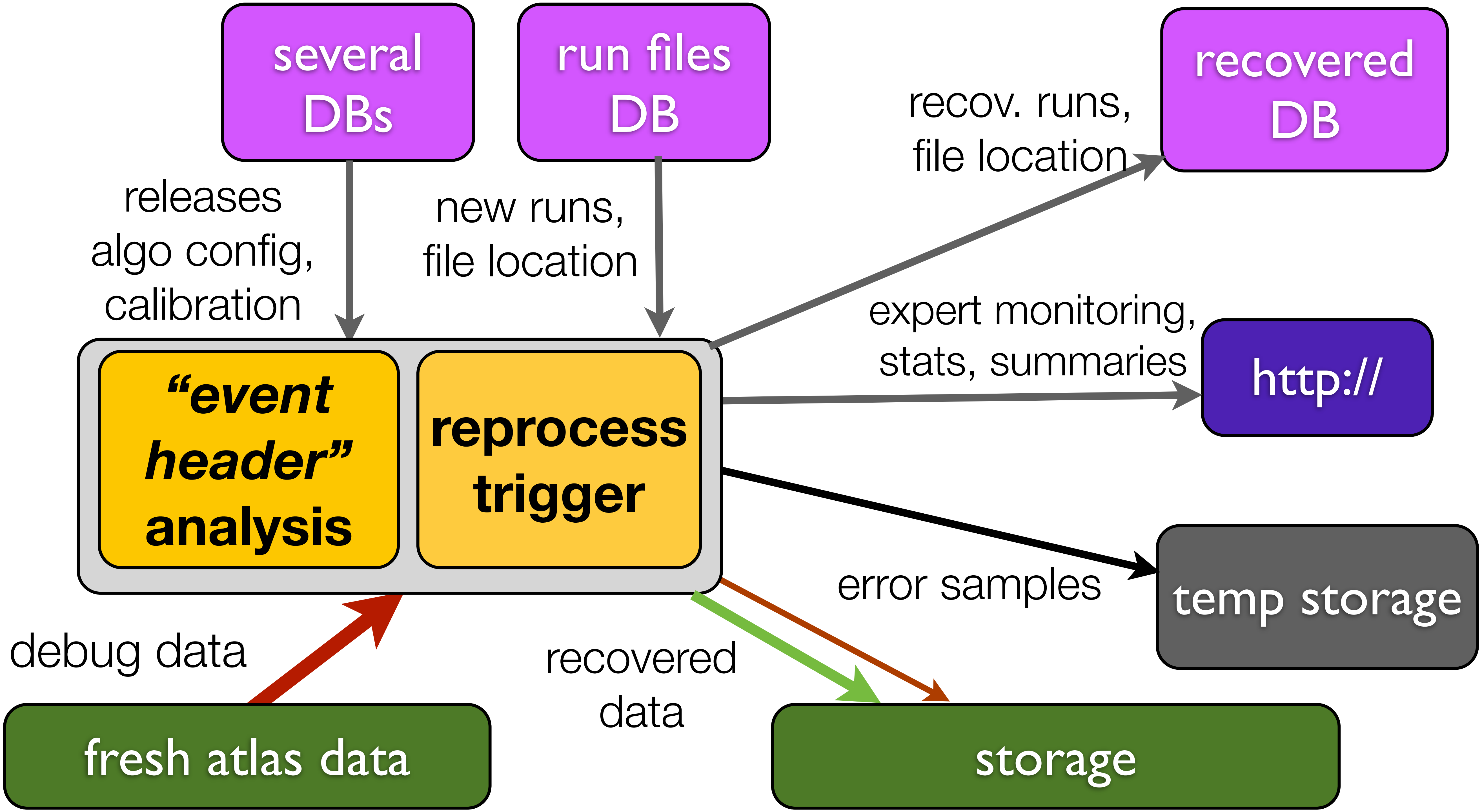
a peek into the debug stream

- ... it contains two types of errors:
  - algorithm persistent errors (exceptions, missing feature, etc)
    - imply SW fixes

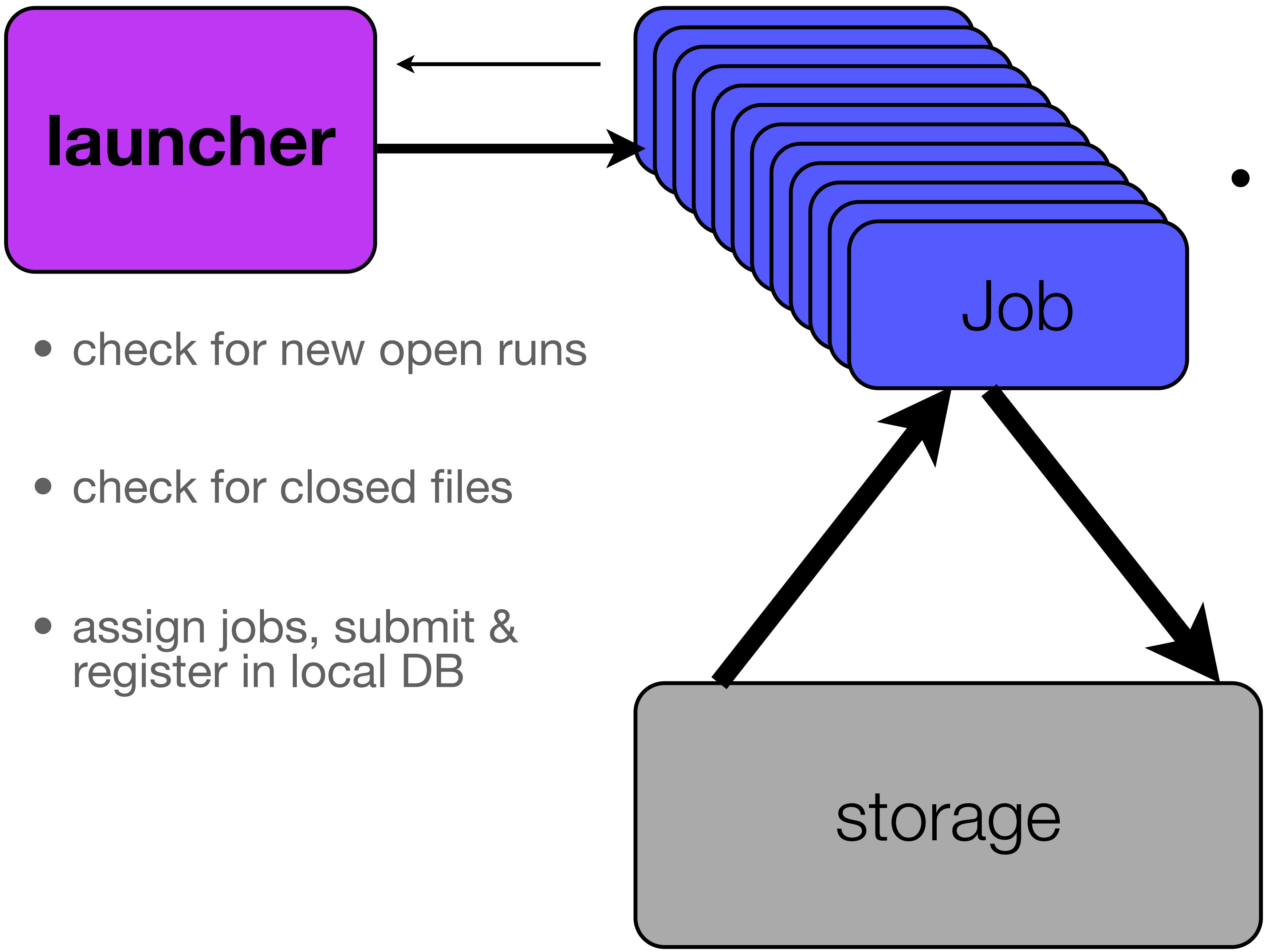
  - online transient errors (crashes, timeouts)
    - re-run selection algorithms

  - more detailed classification in latest configurations
    - (debug.efd.timeout, debug.efd.stoptransition, etc)

a framework for quasi real-time analysis/recovery



launching the parallel process



- check for new open runs
- check for closed files
- assign jobs, submit & register in local DB

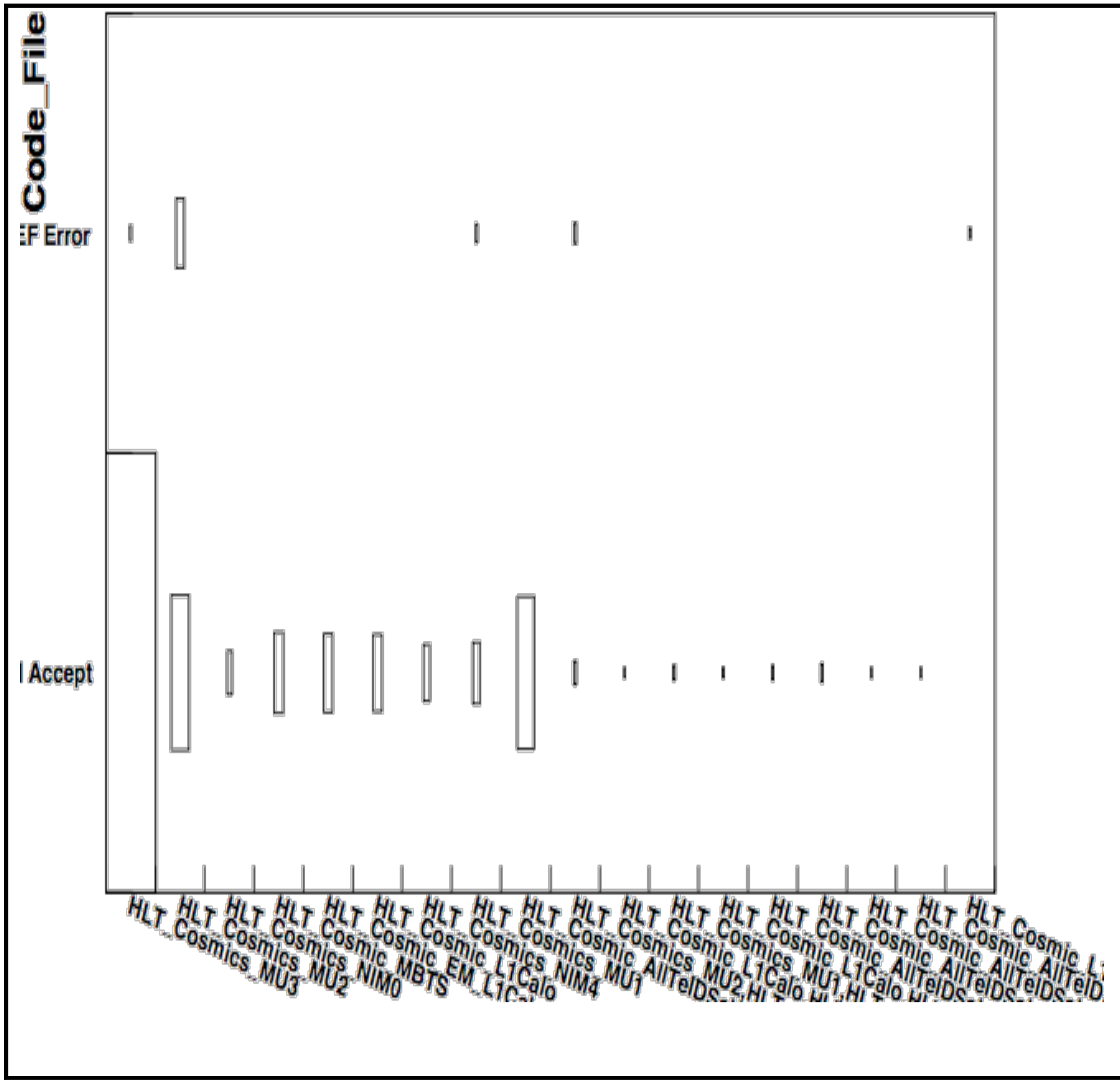
- splitting in subjobs is easy in event oriented experiments
- all events independent from each other

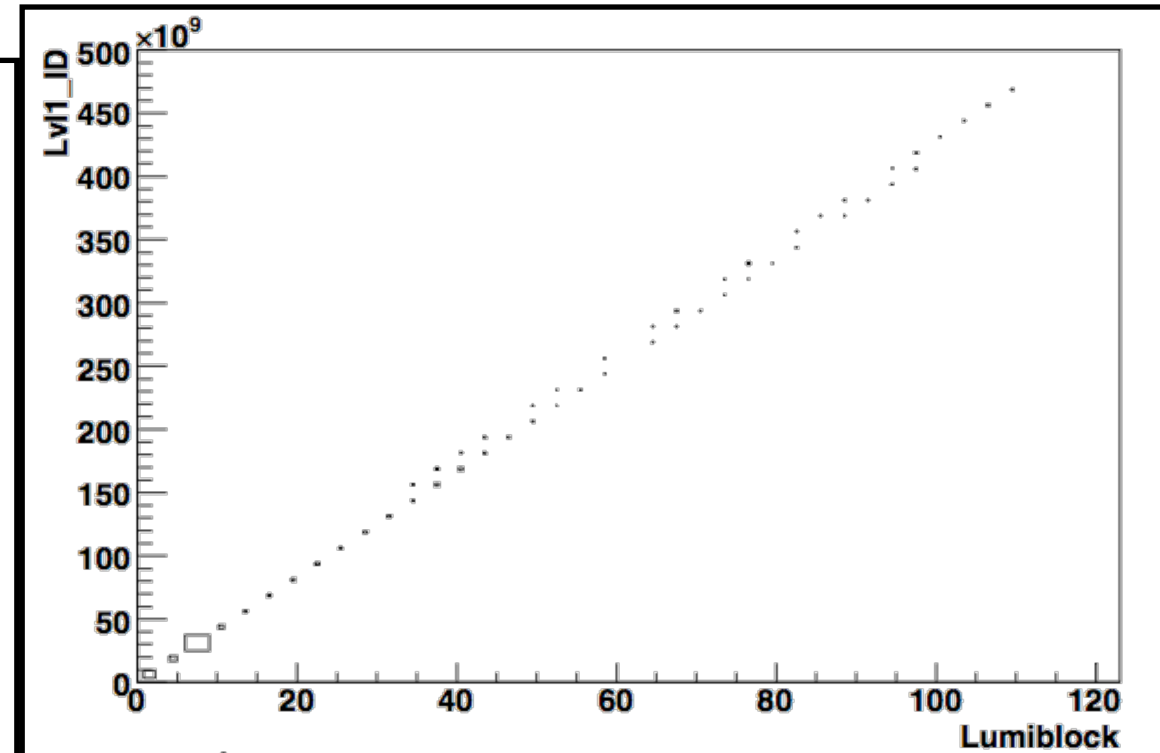
- next level debug stream?
- where to stop?

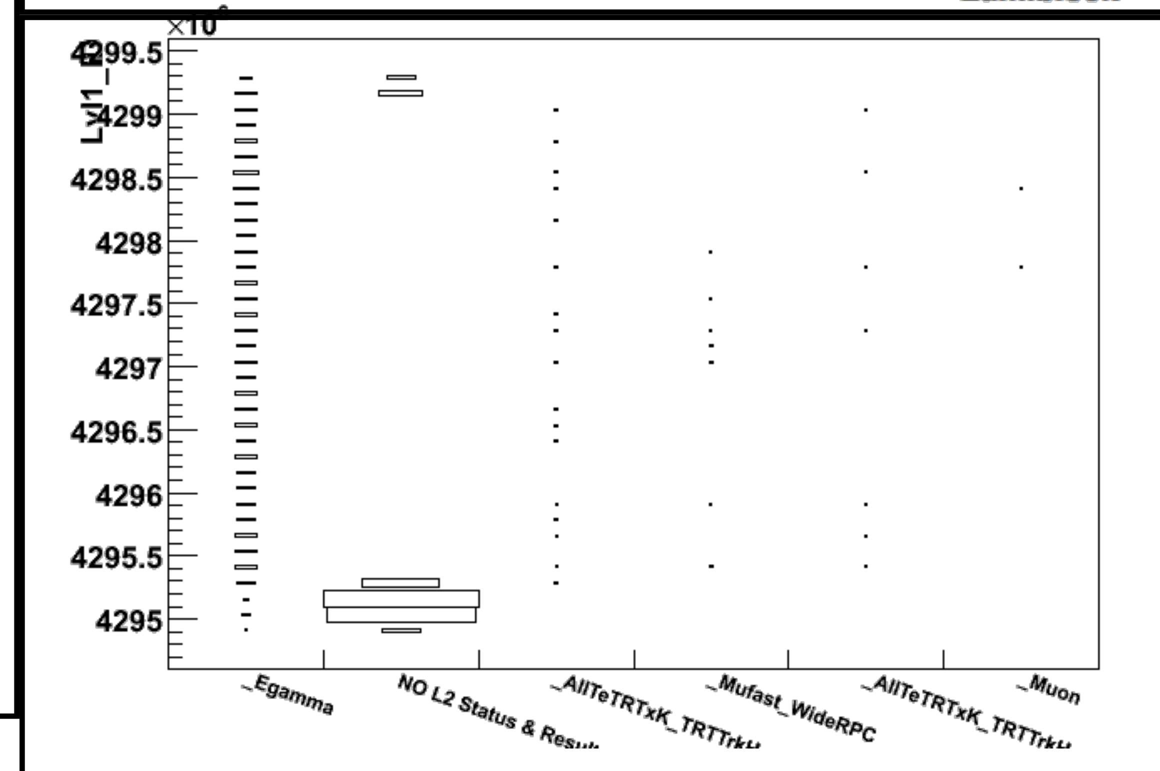
event header analysis

- plots that summarize the errors and other characteristics of the debug stream events
- done using the information of the event header
  - HLT/L2/EF result, L2/EF errors, chain ids, stream tags
  - present no matter what happened in the event
- leads bug tracking in selection algorithms
- create event samples with a handful of events for experts

event header analysis

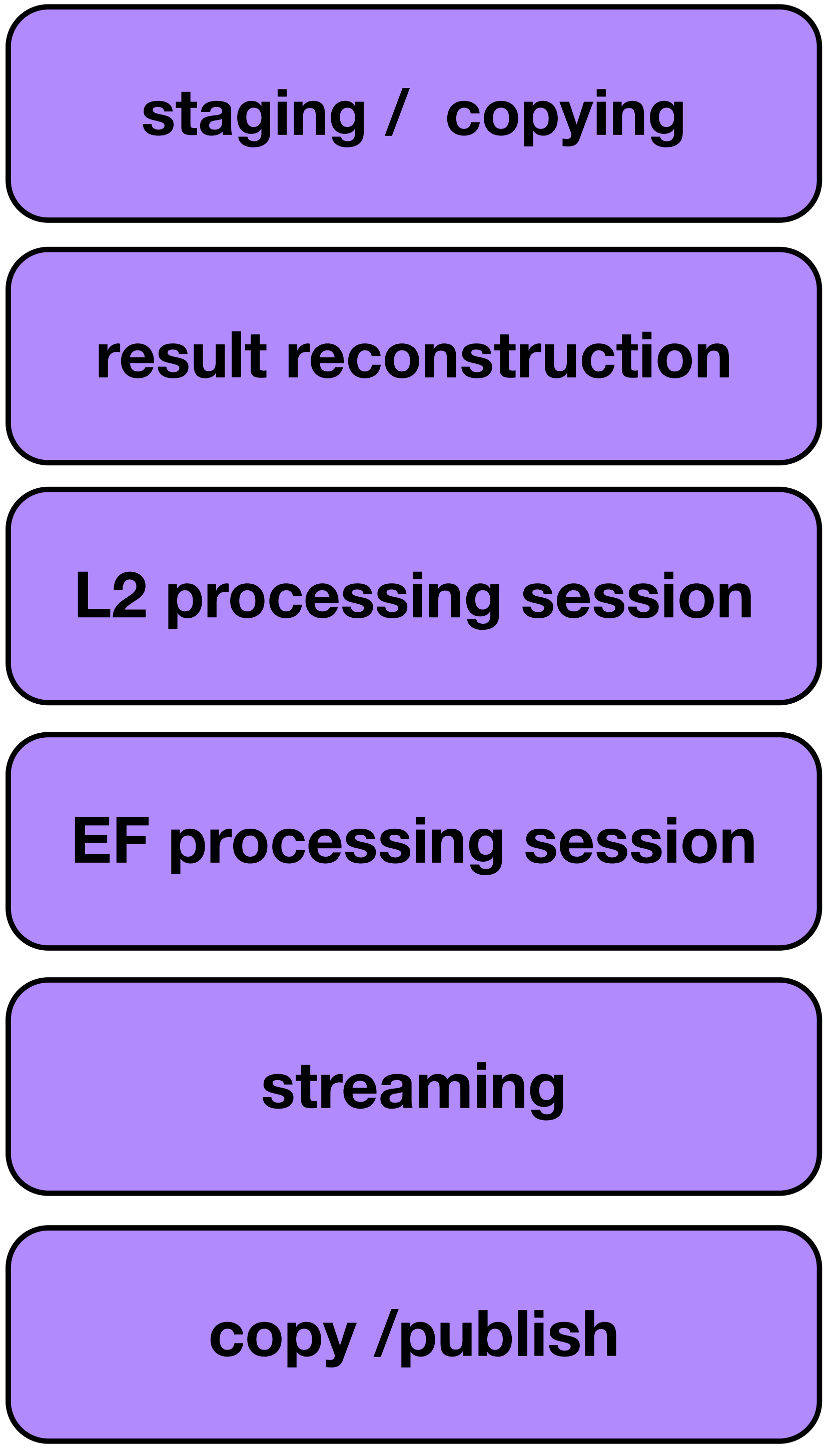






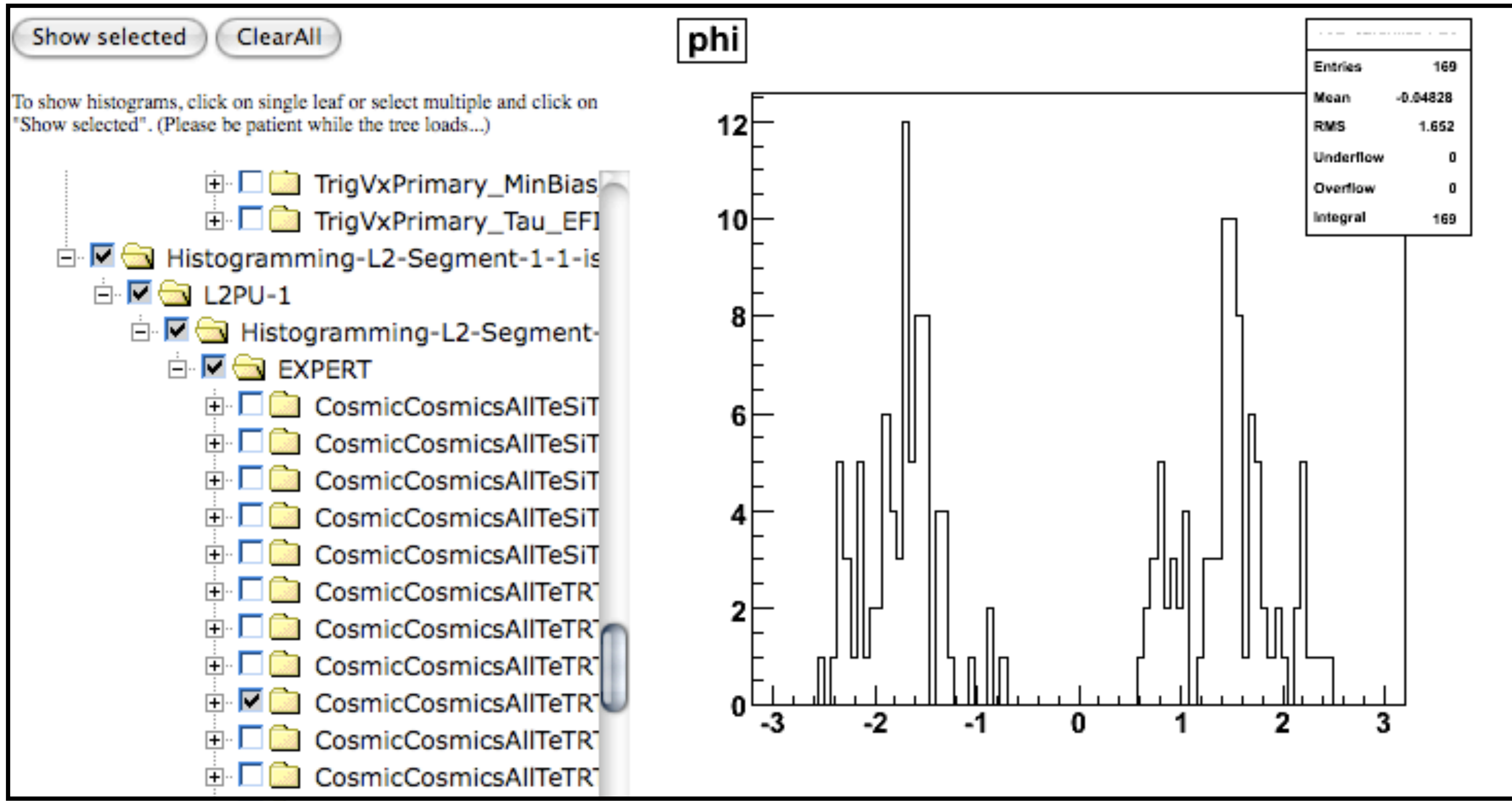
reprocessing selection algorithms

- selection algorithms repacked
- some functionality from DAQ replicated
    - failover, streaming
    - tempted to use DAQ directly
- detailed reports per stage
    - object passing
- validation of system in the data chain
    - redundant counters



reprocessing selection algorithms

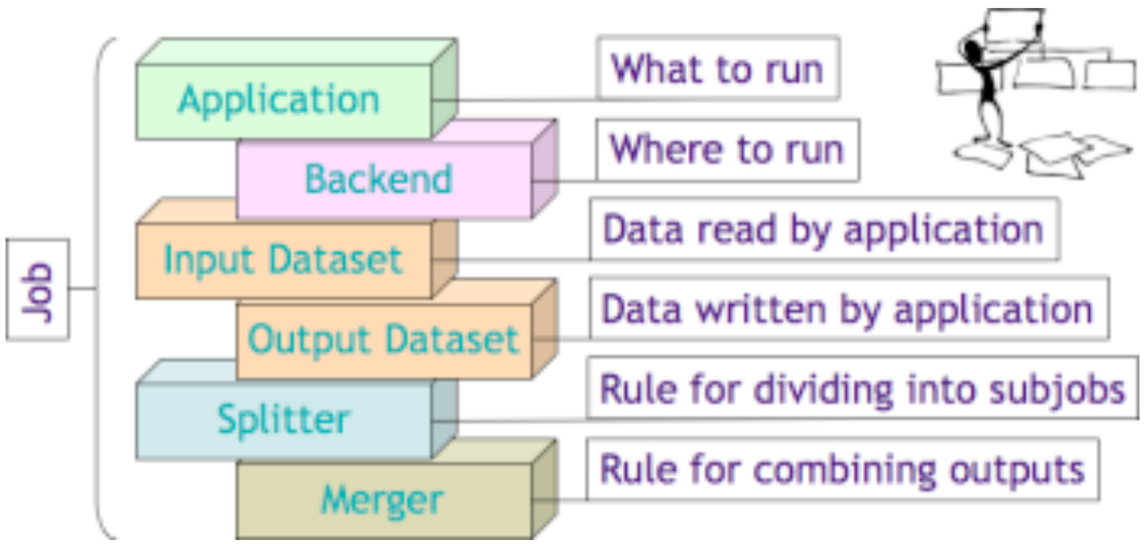
Good Jobs	46
Bad Jobs	0
Step	L2 Processing Session
Events In	5524
Events Out	5524
Events Crash	0
Files In (kB)	78355864
Files Out (kB)	78360668
Step	EF Processing Session
Events In	5524
Events Out	5524
Events Crash	0
Files In (kB)	78360668
Files Out (kB)	78362208



GID	file	offset	logfile
1918212	daq.ATLAS.0080963.debug.efd.LB0000.SFO-3._0008.data	2	daq.ATLAS.0080963.debug.efd.LB0000.SFO-3.processed.ef._0008_1.log.crash
2321029	daq.ATLAS.0080963.debug.efd.LB0000.SFO-3._0009.data	3	daq.ATLAS.0080963.debug.efd.LB0000.SFO-3.processed.ef._0009_1.log.crash
1812486	daq.ATLAS.0080963.debug.efd.LB0000.SFO-1._0007.data	6	daq.ATLAS.0080963.debug.efd.LB0000.SFO-1.processed.ef._0007_1.log.crash

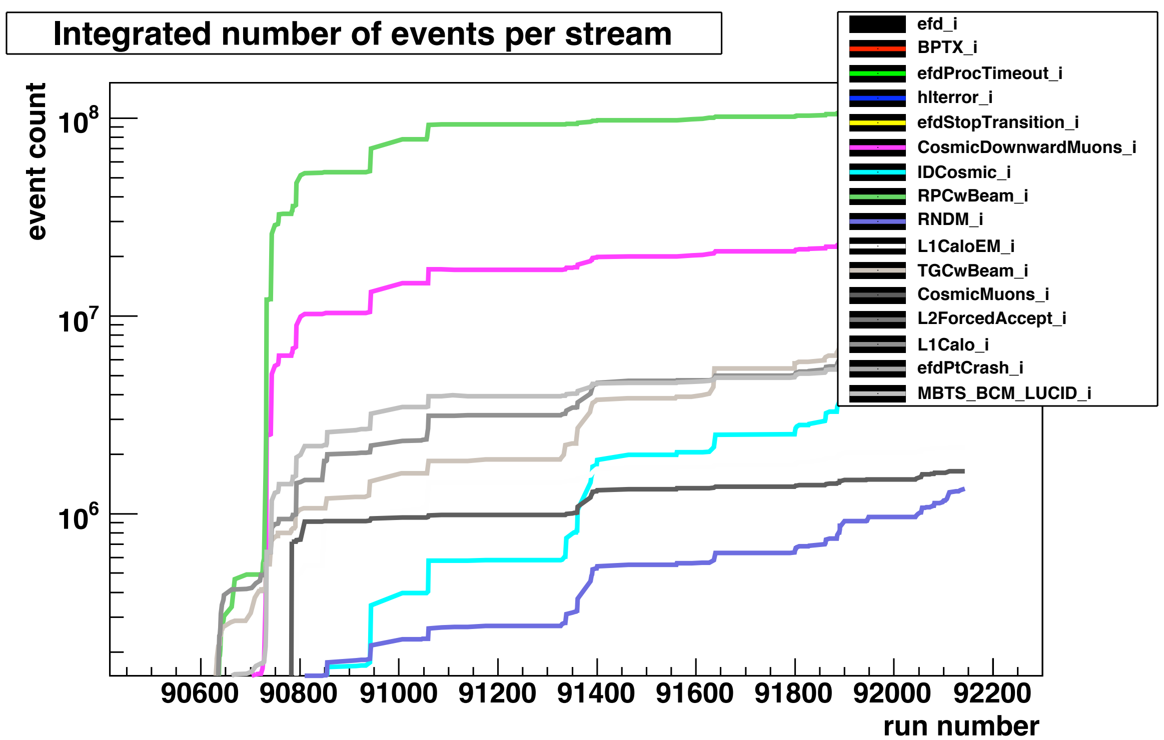
ganga real time extensions

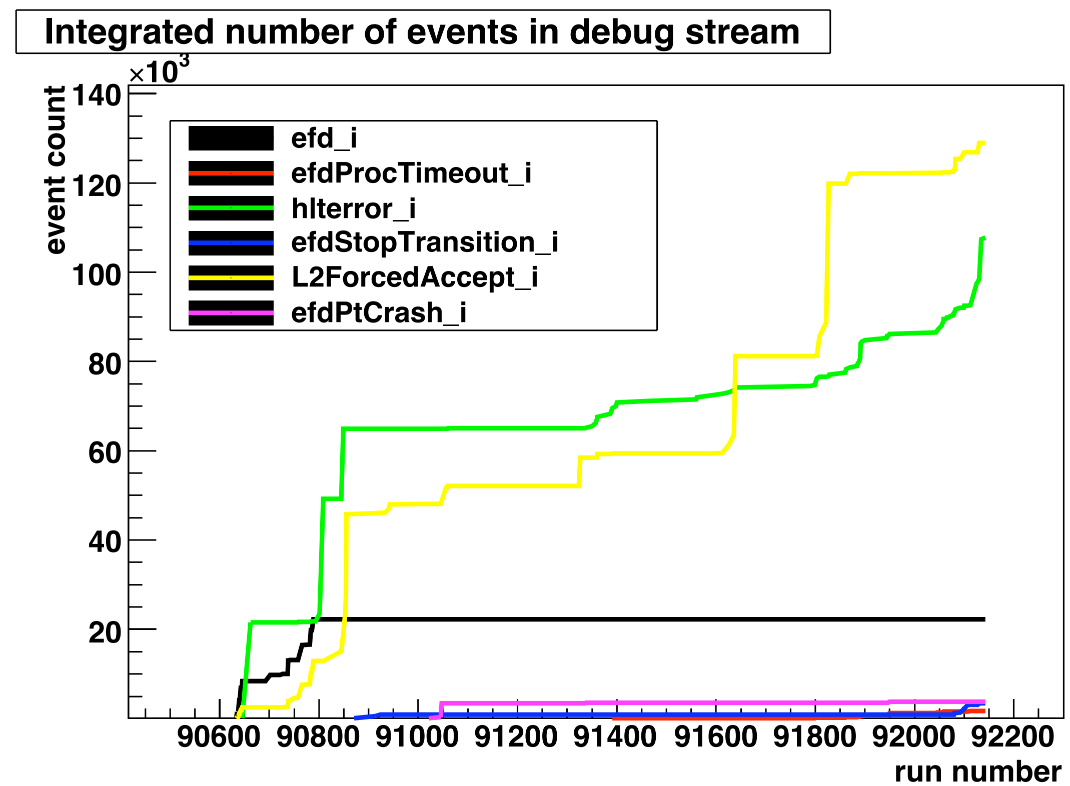
- ganga is a frontend for parallel job management
  - several built-in applications: **Atlas**, LHCb, ROOT, etc.
  - several built-in backends: Local, LFS, PBS, LCG
  - one major drawback: no progress update
- 2/3 hierarchy of client/server mergers
  - trap job completion message
  - implement append method in servers
  - **GangaTrigger module**

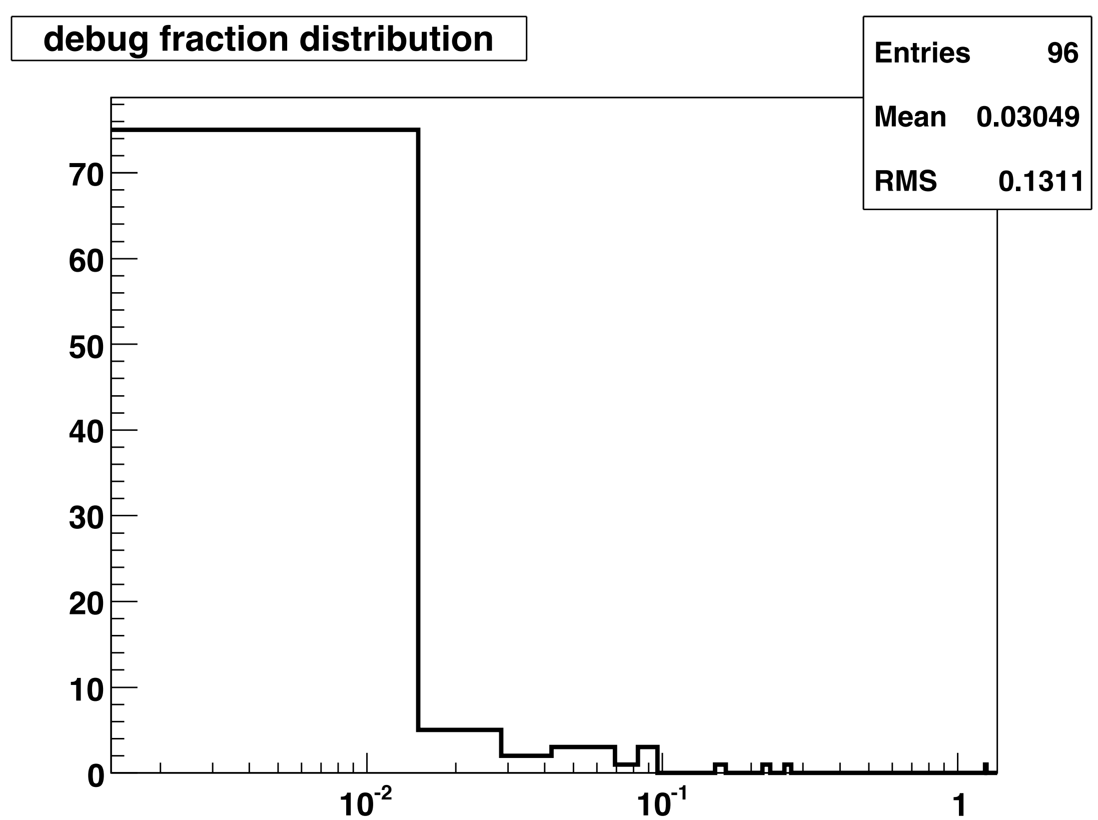


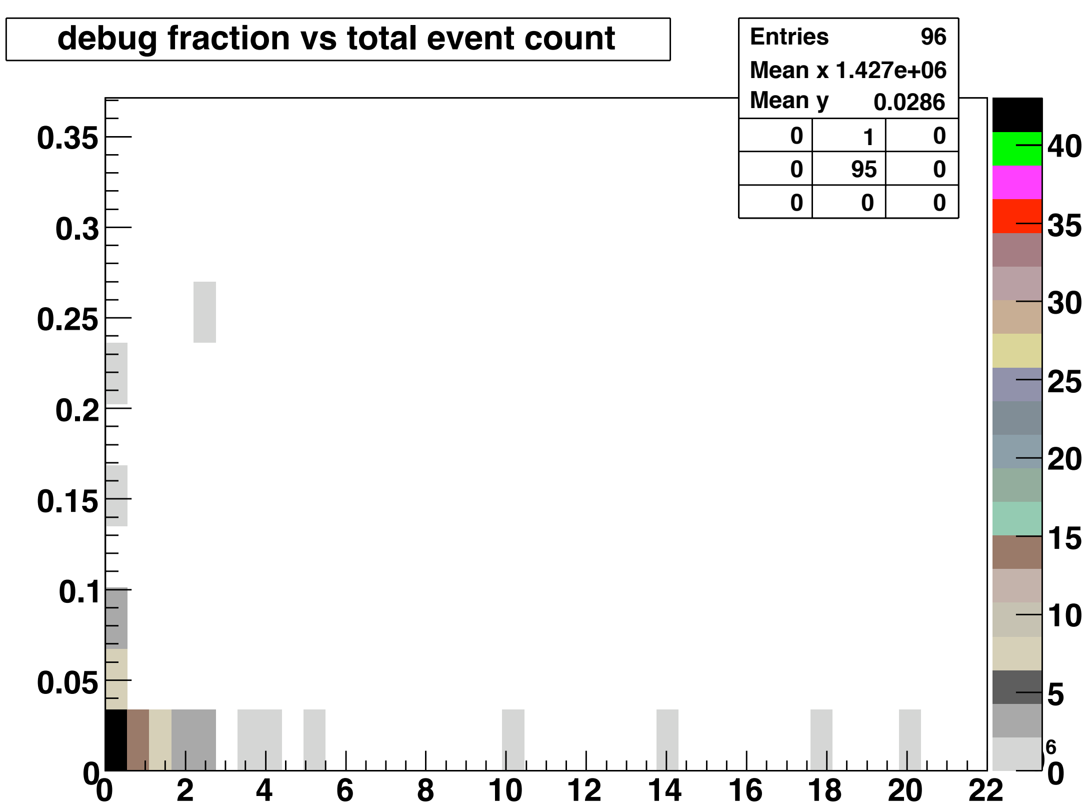
experience with cosmic data

how many problematic events?





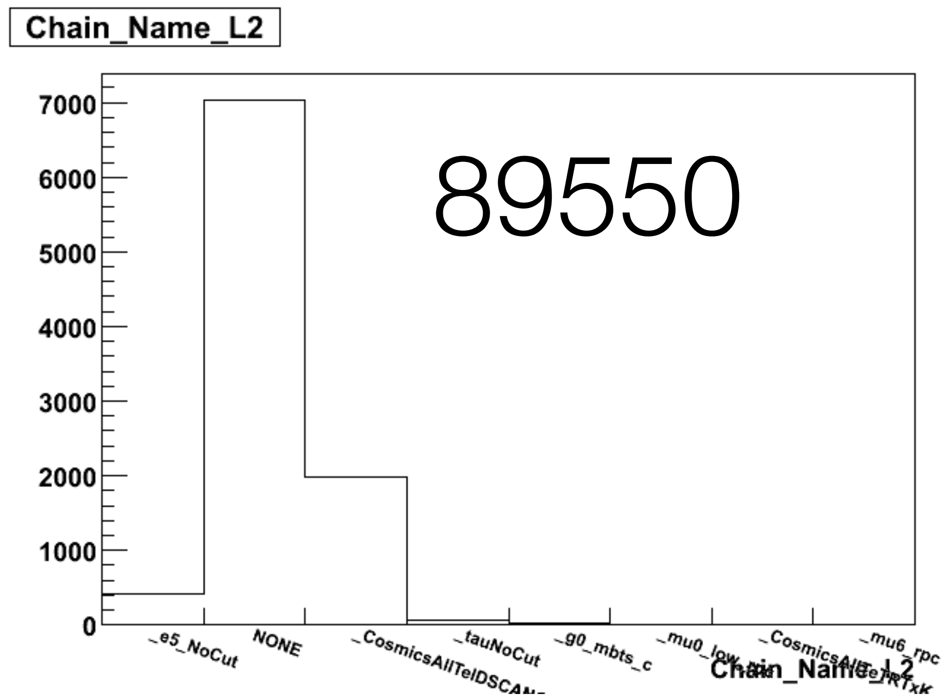


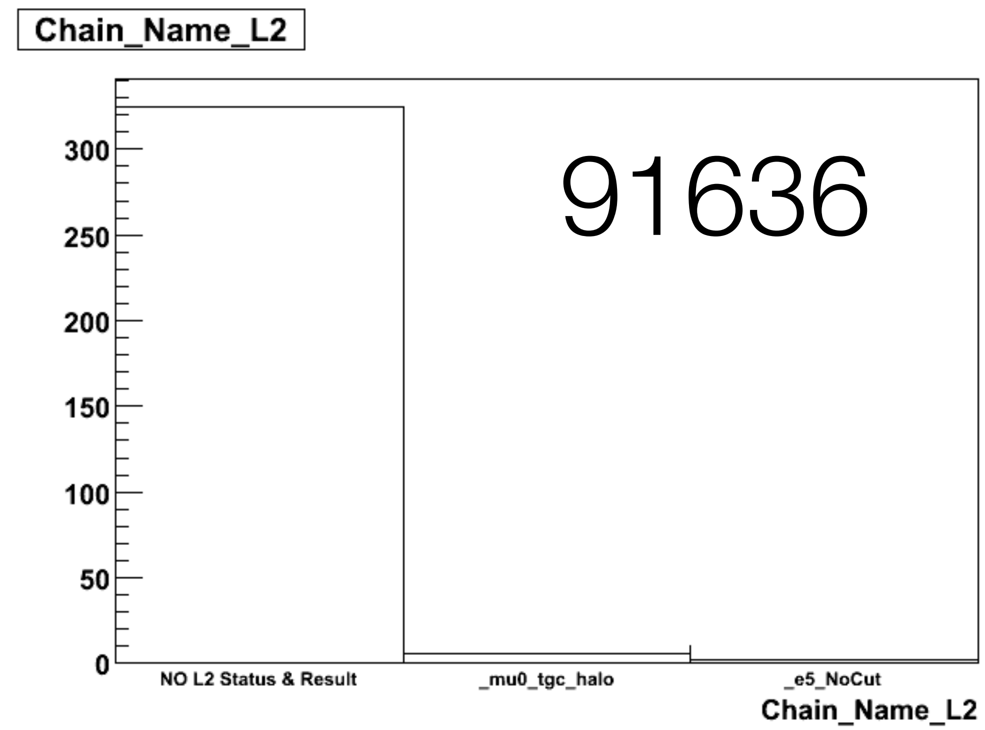


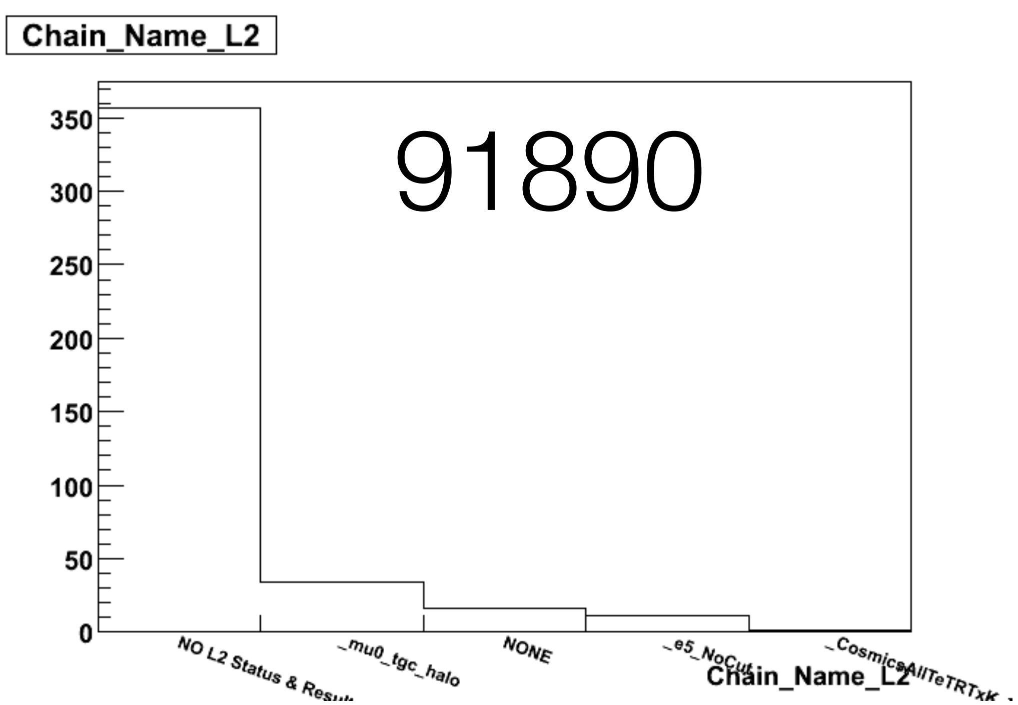
systematic analysis/recovery of the debug stream in cosmics

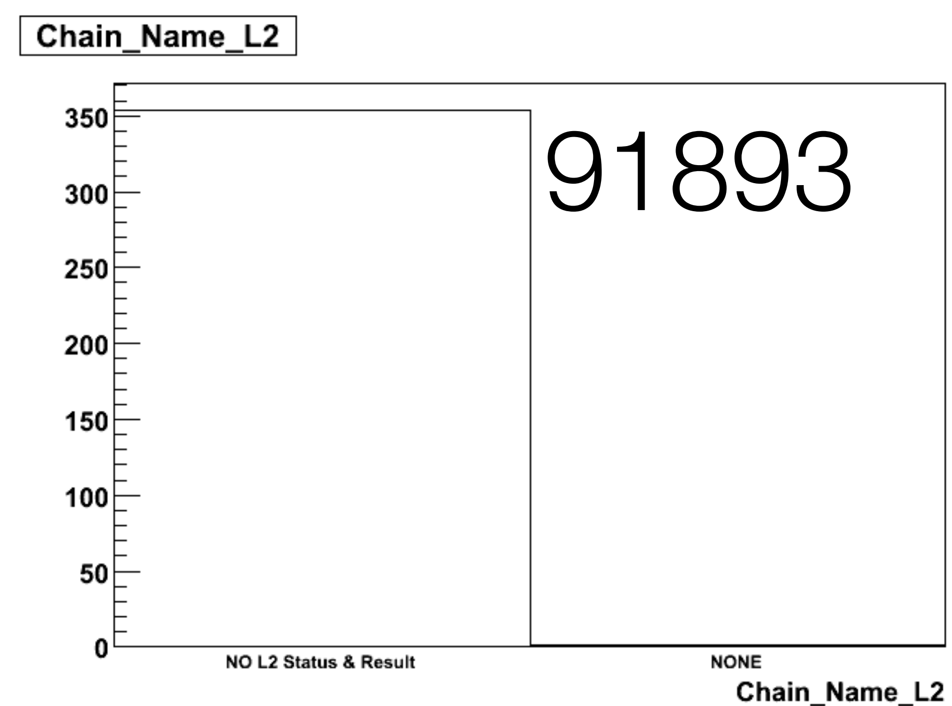
- the system is not fully operational yet
  - possibility to redesign some parts due to *“winter shutdown”*
  - still requires shifter supervision to start selected runs
- this is what a shifter could say:
  - most/all events from online transient errors are recovered
  - and crashes reproduced
- some studies on recovered data still missing
  - e.g.: timeout / event size / event content

a useful bug tracking tool

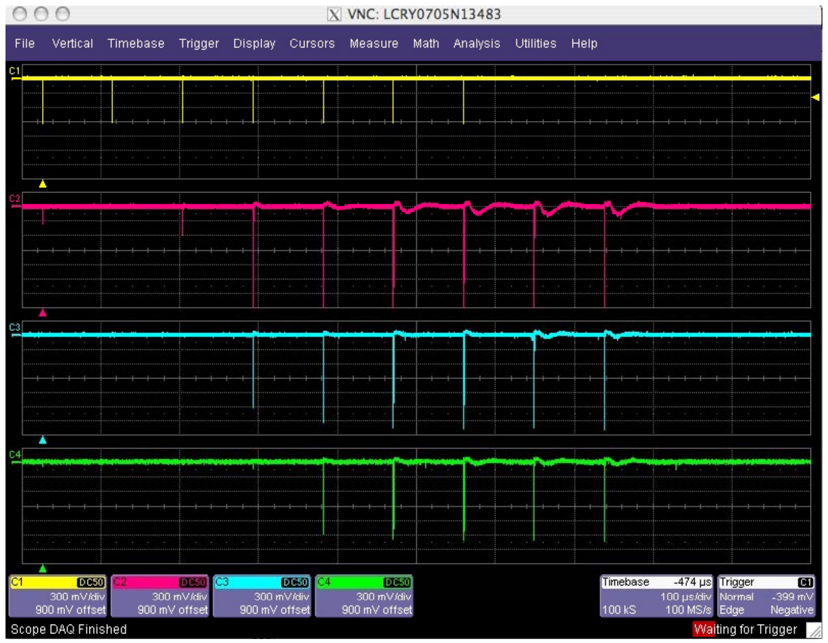




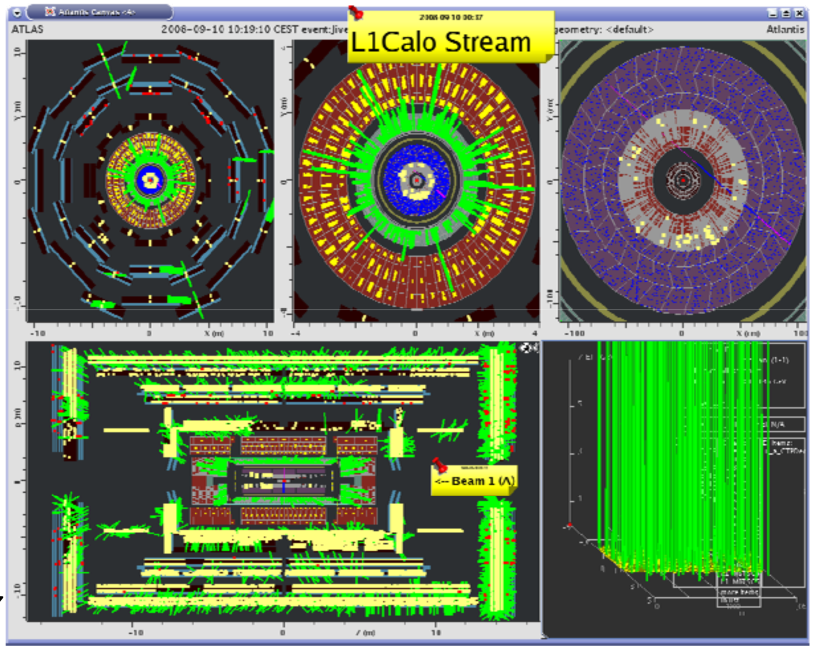




other uses: a happy start-up without trigger



7



3

so:

- we understand the elements that integrate the debug stream
- we have a framework for analyzing / recovering it (almost ready)
- will provide feedback during ongoing run
- let’s not forget: the debug stream is where the errors go
- you can’t just ignore errors, interesting physics might appear there for the first time