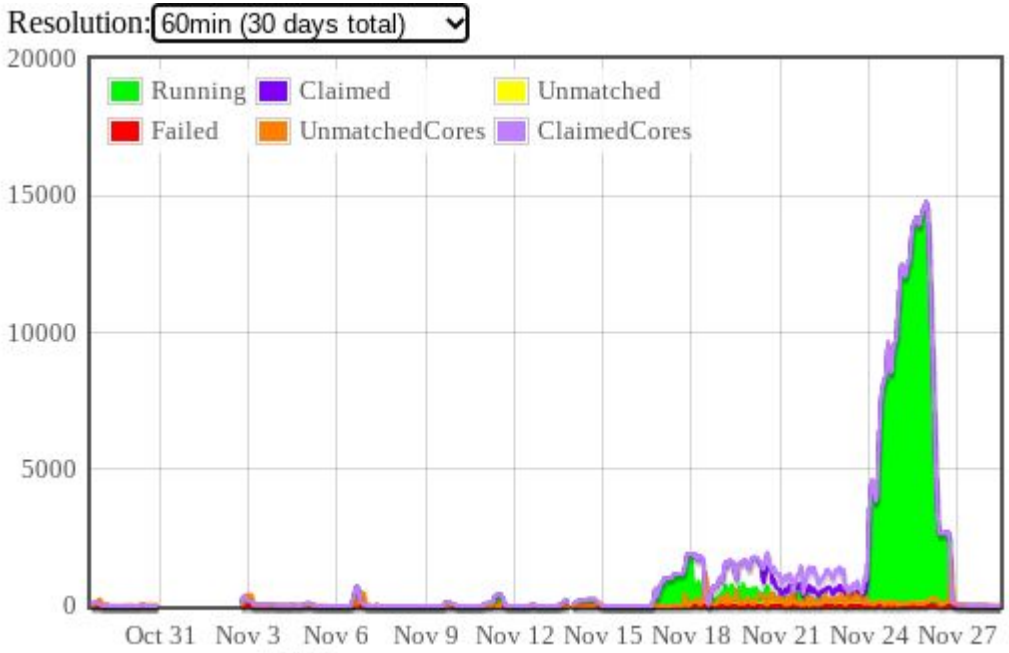
GlueX raw data production on the osg

Results from an Initial Test

Richard Jones, University of Connecticut

a feasibility study

- 10 runs 71728 - 717423
  - 1880 raw evio files
  - 600M events
  - 35 TB
  - ~500k core-hr
- maps onto osg workflow
  - 4 cpu-hr / 1-core job
  - 68 jobs / evio file
  - 126,000 jobs
- production highlights
  - completed in 3 days, see plot
  - 15,000 running cores, max!



what happened in this time?

1. data had already been staged from JLab to UConn *(staging time not included)*
2. jobs were submitted from gluex.phys.uconn.edu
   a. shared same gluex factory with regular MCwrapper jobs
   b. standard gluex queue, no special boosted privileges
   c. data accessed by osg worker over xrootd in real time
   d. output copied back to UConn SE at job exit
3. individual job output tarballs need to be verified and merged
   a. performed by cron job running on the UConn-HTC cluster (colocated with SE)
   b. one-step merging, verification, and push of merged results to jlab
   c. automatic requeuing of incomplete or failed jobs as results arrive
   d. workflow management using a postgres database
4. not everything worked perfectly...

what worked well?

1. postgres database as a management hub
   a. jobs requested their work from the database when they started up
   b. jobs returned their exit code and completion time when they finished
   c. no problem with scaling
      i. ~5 jobs starting/finishing per second
      ii. merger cron job multiplicity scaled to keep up with job output
      iii. management was hands-off -- ***watch but do not touch!***
2. xrootd feeding data from UConn to running jobs in real time
   a. required forking the xrootd client to incorporate evio file chunking
3. No data left behind!
   a. only failed slices needed to be reprocessed (1% of a full raw data evio file)
   b. reprocessing was immediate -- within seconds!
   c. verification + push to Jlab was part of the merging process

where were the difficulties?

1. hd_root memory footprint: RSS = 5.5GB @ 1 worker thread !!
   - hd_root + danarest alone = 1.4GB **-- *good!***
   - there are a lot of plugins -- 29 of them
   - most of the bloat comes from root histograms -- 17,000 mostly TH2I
   - remove the 5 worst offenders = 2.5GB -- ***kinda ok, not ideal***
2. thread explosion
   - after jobs reached >50 threads, they started to bog down but never quit
3. hanging transfers pushing output back to JLab
   - enagaged with UConn NetOps to resolve this
   - very difficult to diagnose problems at the JLab firewall from outside
   - connections were being unpredictably dropped -- ***hanging sockets***
   - picking up with JLab NetOps -- ***hanging communication***

follow-up: *hd_root memory footprint*

an elegant solution: **Compact_ROOT**

- a new header-only class in libraries/include
- just add this to the top of any file that make root histograms

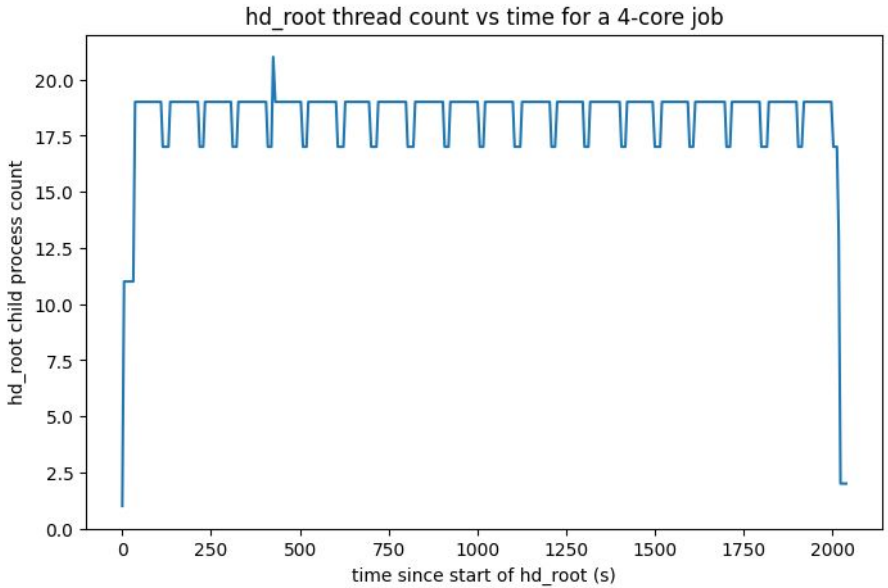
```
#include <Compact_ROOT.h>
#define TH2I Compact_TH2I
```

  - a thin wrapper around standard ROOT TH2I
  - standard TH2I is written to hd_root.root at job exit, as usual
  - contents of the saved TH2I object are identical with or without Compact_TH2I
  - reduces the memory footprint of a 200x300 TH2I by factor ~100
  - cost in terms of additional cpu overhead in production with hd_root is 1%
  - 1% cpu overhead matches overhead from multi-threading at 32 workers -- ***cost neutral!***

- reduces hd_root + all plugins (29) to RSS = 2.1GB -- ***perfect!***

follow-up: *thread explosion*

- "single thread" does not mean 1 thread
- one thread is started for each output evio file for the skims (and for other bg tasks)
- input files are broken into individual evio blocks, one block per file
- output threads are "duped" every time a new input file (evio block) is opened
- work-around was suggested by ***David L:***
  - *"name all of the input files the same"*
  - *don't ask...*
  - ***cool, it works!***



follow-up: *hanging transfers to JLab*

1. work-around for initial test -- write to /osgpool on sciosg16.jlab.org
2. today I received a warning from Kurt -- */osgpool is 95% full, please clean up!*
3. message from JLab scicomp managers

   > We're working on a more robust solution for sending data from offsite back to the lab. Do what you can for now, and watch for more news...

4. meanwhile I have a couple of tickets open with Scicomp Help