Data Processing Tools

N. Baltzell

March 6, 2019

CLAS Collaboration Meeting

Introduction

- Talk about developing workflow tools, i.e. things that should promote efficient, large-scale, hands-free CLAS12 data processing

- The current, primary context is on utilizing JLab computing resources, immediately the decoding of Spring 2018 data (which has its own unique challenges)
    - although the work is intended to be extended to more components of our data processing and offsite computing resources

- Meanwhile explain some of the technical options, challenges, considerations
    - remember, our data volume is 100x higher than 10 years ago

- This talk might not be very interesting to everyone (anyone?), and I only have one image in these slides, but good management tools for data processing are important for the collaboration!

What is "Decoding"?

- A preliminary reconstruction stage
  - translation tables
    - e.g. crate / slot / channel → detector / sector / layer / component
  - FADC250 pulse extraction
    - un-bitpacking
    - time / integral / pedestal analysis
  - data integrity checks (e.g. coincidence of TI timestamps across many crates)
  - tagging and translation of special banks/events for easy access later
    - e.g. beam charge scalers, epics, delayed helicity flips, gemc/DAQ config banks
  - Input EVIO, output HIPO

- Output data file is 4X smaller than input

- Eventually we might decode ...
  - inline during reconstruction (i.e. without an intermediate file)
  - or even online in level-3
  - meanwhile, in the beginning of CLAS12, there are some advantages of it being a separate process and file (but that's a topic for another talk)

Why decode now? Why fancy workflows?

- It's true that decoding cpu time is very small compared to reconstruction, and it's simpler to manage on the batch farm because it's single-threaded, but tape/disk access is often a bottleneck with its own competition for resources.

- And Spring 2018 raw data has a particularly unfortunate situation on tape:
  - single runs (few TB each) are spread across many (we've seen 7, 10, 12) tapes (12 TB each, ~16 TB uncompressed) due to some combination of
    - unanticipated raw data rate combined with transfer style of large instantaneous batches
    - a tape reorganization by scicomp in summer 2018
  - for perspective, it's a total of 1 PB of raw EVIO, which in itself is 3 weeks on a single mover without accounting for multiple tapes

- And preparation for full, final reconstruction pass for Spring 2018 wasn't ready (e.g. software, calibrations, alignment, etc). Meanwhile we could ...

  - go ahead and decode and put the decoded files back on tape in a more orderly fashion to optimize future access
    - lots of conveniences in processing files by run (database access, geometry and magnetic field management, on-the-fly data quality monitoring)

  - utilize current time window for tape access rather than complicate with more bottlenecks later
  - gain some systematic batch farm experience for CLAS12, learn and start preparing workflows for large scale reconstruction etc.

Some Decoding Objectives and Limitations

- Objectives
  - Decode single EVIO files into HIPO
    - as independently as possible to allow the batch system to optimize tape read access
  - Merge N sequential files into 1 file
    - we chose N=10, so 5 GB files instead of 500 MB files
  - Write merged files to tape
    - as sequentially as reasonably possible, both run-to-run and within runs
  - The latter 2 are for the future, optimizing tape usage and minimizing bookkeeping

- This incurs the limitation of a predictable amount of reliable disk space
  - a staging area, e.g. for the single files until their N-1 merge partners are ready
  - it's a tradeoff between minimizing disk space requirement and optimizing tape access
  - how much disk space depends on the details of the workflow
    - to independently decode and then merge N files, we need a minimum of N GB

- Workflow options vary
  - from the disk-heavy, tape-optimal ones with multiple pseudo-independent parts that coordinate to shift stuff around and do lots of independent bookkeeping to keep things reliable (and are not so generally applicable in the future)
  - to a more centralized, self-contained workflow, that is more light on disk resources but not theoretically most optimal for throughput on the tape library

- In late-December 2018, we started fully exploring the details, and after some experience chose to utilize Swif and lean more towards the latter option above ...

What is "Swif"?

- It's a JLab product: https://scicomp.jlab.org/docs/swif
  - the "scientific workflow indefatigable factotum"

- Allows to interact with JLab batch farm *programmatically*
- Write jobs and retrieve job status in a standard, structured format (xml/json)
  - e.g. cpu/ram/disk usage, error conditions
- Retry failed jobs
- Modify jobs, e.g.
  - increase resource requirements
  - abandon jobs
- ***Organize jobs into a workflow***
  - group jobs into serial "phases"
  - explicit job-job dependencies, "precedents"

- For some/many objectives you might be right to not care, e.g.
  - your failed jobs can be ignored and/or just run another independent job
  - your jobs have no inter-dependencies
  - you'd rather spend time manually monitoring your jobs, doing bookkeeping, independent error/integrity checking, writing and monitoring cleanup scripts

- Meanwhile, Swif is also to be the JLab conduit to offsite resources (e.g. NERSC, OSG, etc)

Some features of the current CLAS12 implementation

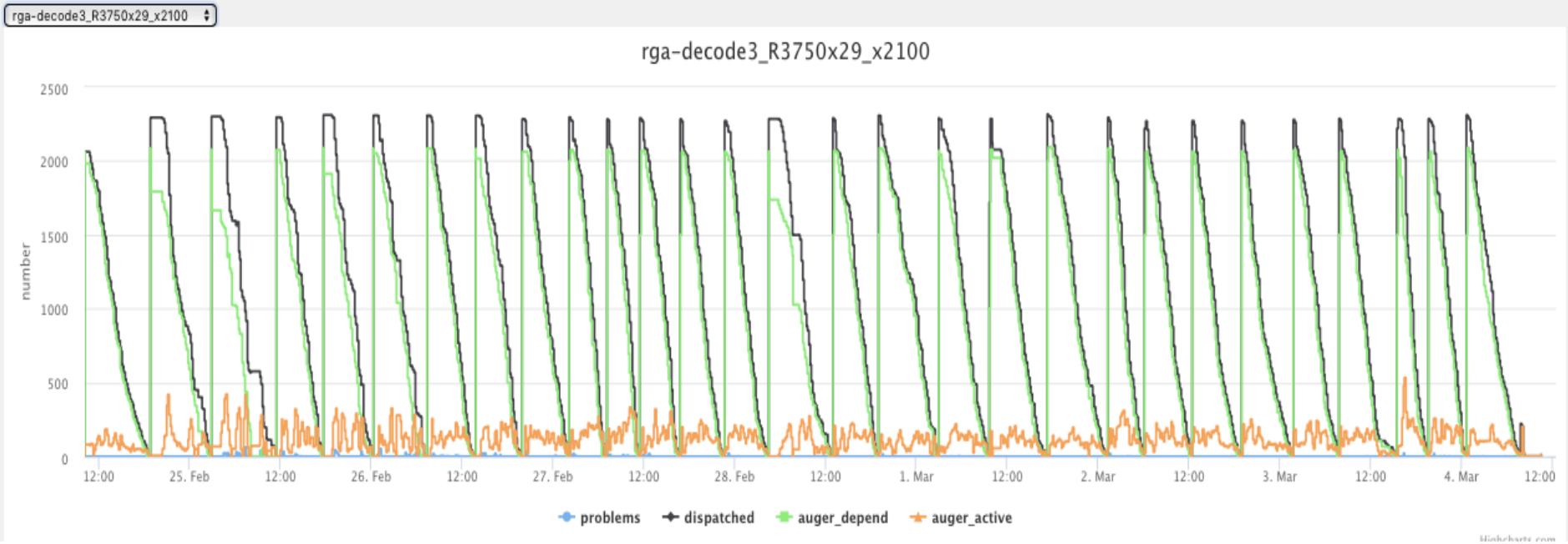
- For decoding, settled on a 3-stage, Rolling workflow
  - essentially 3 three-phase workflows interleaved, to give some reasonable optimization between tape usage and disk usage, including reliability considerations on disk space
  - decode, merge, move and delete
- file integrity checks during the jobs, based on hipo-utils, with status reporting to batch system
- retrieve torus/solenoid scales from RCDB during workflow generation
- utilize Swif's job tags (e.g. output directory, run/file numbers, coatjava version) for later automated bookkeeping, with named jobs and log files to facilitate debugging if necessary
- automatically retry jobs due to system failures and adjusts job resource reqs if necessary

- The software currently lives here: https://github.com/baltzell/clas12-workflow
  - will move to jeffersonlab's github organization by the time it's ready for reconstruction
  - written to be somewhat generic to accommodate more tasks, and now starting to incorporate reconstruction ...

Another feature of the CLAS12 implementation

It pushes to clas12mon, via standard Swif JSON format and a cronjob

workflow_name	jobs	succeeded	success	attempts	phase	dispatched	depend	active	update_date	current_date
rga-decode4_R3501x30_x2100	67620	10022	14.8%	11058	5	957	868	55	Mar 5,2019 23:56	Mar 6,2019 00:00
rga-decode3_R3750x29_x2100	66583	66583	100.0%	67803	31	0	0	0	Mar 4,2019 10:32	Mar 4,2019 10:40
rga-decode2_R3464x28_x1300	66415	66415	100.0%	66998	58	0	0	0	Feb 24,2019 04:54	Feb 24,2019 05:00
rga-decode1_R3432x25_x1300	64076	64074	100.0%	79290	52	0	0	0	Feb 11,2019 15:03	Feb 11,2019 17:42
rga-decode5_R3355x30_x1300	65222	0	0.0%	0	0	0	0	0	Jan 30,2019 19:21	Jan 30,2019 19:22
rga-decode6_R3466x34_x1300	67649	0	0.0%	0	0	0	0	0	Jan 30,2019 19:13	Jan 30,2019 19:16
rga-decode7_R3249x43_x1300	67252	0	0.0%	0	0	0	0	0	Jan 30,2019 19:05	Jan 30,2019 19:06
rga-decode8_R3191x65_x1300	65648	0	0.0%	0	0	0	0	0	Jan 30,2019 19:00	Jan 30,2019 19:02
rga-decode9_R3135x178_x1300	62672	0	0.0%	0	0	0	0	0	Jan 30,2019 18:56	Jan 30,2019 18:58



Lessons Learned

- For Spring 2018 data
  - Reading a single, whole run off tapes, runs at about 25% the hardware spec for reading a single tape (even though we often get multiple movers, i.e. read multiple tapes simultaneously), due to distribution across many tapes and non-contiguous within a single tape
  - Original studies suggested 8K files per day, or 2 months total. After moving to production mode in late January, there were a couple snags in the first couple weeks (system problems, no changes on our side to recover, but required feedback/repair from scicomp) that slowed things down, but lately it's back to around 8K/day
- Not uncommon problems with batch nodes
  - No space available on local filesystem, due to over-prescribing node resources
    - this should be specific to PBS and not an issue on SLURM
  - Cannot find input files (e.g. lustre or /work filesystems) or basic system commands (e.g. rm)
  - Flurry of corrupt files
    - presumably due to above system problems, since irreproducible
  - All resolved via
    - integrity checks (implemented internal to jobs) before allowing output files to be written
    - proper error status reporting to batch system
    - Swif retries
- Workflow performance
  - We've run 250K decoding jobs, plus 25K merging jobs (including final testing phase), with an ultimate success rate of 100% ***with zero human intervention***
  - Ok, there was actually 1 corrupt EVIO file, for 5E-6 failure rate, but avoidable if we automatically, programmatically fix corrupt EVIO files in the future

Summary, Future

- The existing decoding workflows for CLAS12, originally for Spring 2018, are proven to be very reliable, leveraging Swif to achieve ultimately 100% hands-free success rate.
  - Includes a few models, some of which are deliberate and slow, some of which are as fast as possible, depending on the purpose
  - Rrun Group B has been using the same workflow tools and given good feedback
  - We'll use this experience for the future

- These tools were written to be extensible to the next CLAS12 need, which is to incorporate reconstruction in CLARA in Swif
  - including strong inline error and data integrity checking and job status reporting for hopefully another hands-free automation
  - the goal is 1 month

- Swif seems very good for setting up reliable, large, data processing chains, within the constraints of JLab's computing infrastructure, without a clutter of independent scripts and cron jobs to clean up messes
  - we should try to leverage Swif for reconstruction, simulation, monitoring
  - Swif is also a route to offsite resources (e.g. NERSC)

- We may think to tie the software to a real CLAS12 (or JLab?) data catalog to
  - choose what data (runs) to process, based on the database
  - automatically update the database with completion for each data set
  - currently that data catalog is essentially just the read-only /mss filesystem