Run 1 Level-2 Review:

DAQ and CAILB View

Jan Guida

May 22, 1996

CALIB in Level-2

CALIB was very important in the commissioning of the level-2
- first large application of a Level-2 filter
- in the beginning CALIB was much larger (memory-wise) than the physics filters being used at that time (test beam and early Run 1A)
  $\Rightarrow$ found problems downloading and running
- JH was able to reduce the size of the "system" and framework parts of L2 before the start of Run 1A.

CALIB worked well in level-2

Speed was important
- would never have been able to send the events to the host and analyze them during a shot setup

CALIB and L2 FRAMEWORK:
- CALIB fit well
- end run hook would have been nice, but we worked around not having that feature
- two-way communication would be nice:
  send ITC messages from L2 $\rightarrow$ CALIB_TAKER
  but not from CALIB_TAKER $\rightarrow$ L2
  (maybe just our implementation - may not be a fundamental limitation)

Run 2 Considerations

- shot setup much shorter: $\sim 1/2$ hour
- possibily **no** quiet-time
- CALIB needs to be sped up
  especially downloading between steps
  (CALIB→COOR→COMM_TKR)

  Longest calibration in Run 1:
  CAL pulser runs = 34 steps
  $\sim$ 35 minutes, if no other begin or end runs
  occurred during this time to slow things down

Important that CD, CAL, MU all use the same framework
(L2 and host), databases, etc.

1. All CALIB's need to look the same to shifters.
2. Gives more experts
   (CD, CAL, MU experts can help each other)
3. Don't have to write three separate frameworks.

Downloading:

All downloading should go by the same path
- crate, pulser, clock etc.

In Run 1:

  crate and pulser downloads → COMM_TKR
  clock downloads → CLOCK_SERVERS

This made life harder than it needed to be.

DAQ Issues

Nodes crashing:

- frequency varied during Run 1
  there were times of very stable running with ≤ 1 crash per shift
  there were also times with ~ 50 crashes per shift

$\Rightarrow$ need to test all new executables carefully and throughly
before installing
(Don't give into whoever wants to bypass the standard
testing procedures "today only" for speed.)

- need to speed up testing procedures

- need better ways to check for one filter "stepping on" another

- Should always have source code on ONLINE machines so that
  we are looking at the correct version of the source code
  from the debugger.

Hardware related problems:

- had a problem where a couple of nodes crashed more frequently reading data from one particular data cable
  - need a better way to identify these problems

  $\Rightarrow$ keep better statistics on type of crash, frequency and L2 node number over long term

- need better way to tell if a node is "running slow" or in any way behaving differently from other nodes
  (CALIB found slow nodes in Run 1)

- need to monitor hardware related problems better, so can catch them earlier
  - at present, nodes turn themselves off (if too many errors)
    - this is **good**, but can we do more?

- Would saving the "sup cable" information with the end store command be helpful?

- Would need some "automated" way of looking at the data and reporting problems.

But don't automate too much!

- important for people to look at things carefully, which doesn't happen if problems are "automatically handled"
- If things are too transparent to shifters, they will never understand the system well enough to be able to fix problems.

Conclusions

- Level-2 system served us well

- Even though there were painfull periods, we had a lot of relatively smooth running

- CALIB worked well in the Level-2 framework

- CALIB also served to find problems which were hard to spot using only physics filters.