View from the DAQ

Paul Dauncey

Imperial College London

Background on the DAQ

- There are more than just events in the raw data
  - Generalised data "record"
    - Run start/end/stop – unchangable settings, e.g. serial numbers, firmware versions
    - Configuration start/end/stop – changeable settings, e.g. timings, DAC values (write and readback at start, readback only at end/stop)
    - Spill start/end/stop – mainly error reporting
    - "Slow" controls/readout/startup – equivalent for non-time critical data
- There are more than just ADC data in the event records (~9kB now)
  - CRC data themselves
    - Trigger counters from FE/BE
    - Buffer counters and status, error flags, data verification flags
    - ADC values (largest volume of course, now ~7kB, eventually ~20kB per CAL)
  - TDC (or hodoscope) data
    - Or whatever is in the beamline at FNAL
  - Trigger data
    - Counters, status, trigger inputs history (second largest, ~1kB)

Data format

- All data correspond to “flat” C++ class objects
  - Completely contained in contiguous memory; no pointers, virtual functions
  - But can extend beyond “official” C++ object size = variable length
- Records are purely single blocks of memory
  - They know how long they are; stored in a fixed length header
  - Makes them generally transportable; sockets, pipes, disk, etc.
- Objects within records (“subrecords”) depend on record type
  - E.g. CrcFeRunData, CrcFeConfigurationData, CrcFeEventData
  - Often wrapped in location identifier (crate, slot, board component, label)
  - Currently 38 different subrecord classes; will need more with HCALs
- All subrecord objects accessed through typesafe methods
  - Unique id associated with each class
  - Version numbers allow scheme migration with time
- This code all exists and has been used for over one year

Firmware is never finished…

- Subrecord data format usually direct dump of hardware output
  - Classes will have to evolve if/when readout format changes
- Definitely need to change CRC format
  - ADC data stored in 8MB QDR memory ~ 2k events; read by VME block transfer giving direct copy into record
  - Control info stored in FIFO in FPGA – space limited to only 512 events
  - Must reduce control info volume to get 2k FIFO; change to data format
- Trigger data (~1kB per event) not in QDR
  - Must be read per event via slow serial I/O data path (still VME)
  - Very hard to get trigger data into QDR when ADC data being stored also
  - Have nine boards, need six for ECAL; if all OK, use spare for only trigger
  - Trigger data then easier to get into QDR, but data format very different
- We don’t know what other data formats at FNAL will look like
  - Tracking, trigger, PID

Change to computer model in December

- Previously, model had been
  - Calibration and alignment studies using raw data
    - Assumed to be only needed by a limited number of people
  - Calorimeter hits with threshold suppression in more user-friendly format
    - Originally ROOT, more recently assumed to be LCIO (although never discussed)
- December meeting: change proposed
  - Analysis should be able to move smoothly from basic calibration level to high level simulation comparisons
    - A good idea; a common package to be used (almost) everywhere
    - Driven by realisation of the large number of people who need to get involved in calibration level analysis
  - Not sensible to extend the raw data up to the simulation comparison level
    - Conversion of simulation/truth information from Mokka to raw needed
  - Extend LCIO (or ROOT) down to calibration level
    - ROOT more widely used, more flexible, possibly quicker and more compact
    - Political, not technical, decision to choose LCIO

How does DAQ development get done?

- Need to be able to debug DAQ code
  - Where would this be done in the new model?
  - Seems to be forgotten/ignored in many discussions
- Conversion may not be able to handle all the information needed
  - Would have to be very robustly written to cope with
    - DAQ formatting errors
    - New classes for new readout types
- Likely this work would need to be done on raw data files
  - Need access to these offline
- Also, development of online monitoring/histogramming
  - This works from records
  - Developed on raw data files offline
  - Moved to online (transparent to code) when ready
- Need to retain access to raw data outside of the DAQ PC itself

New model: management issues

- Technical issues often not the “make or break” in software
  - Code management and dependencies usually cause most problems
- BIG advantage of new model – breaks coupling between online and offline code
  - These only coexist in the job which does the conversion; very limited number of people need to handle this part
  - Online code can be optimised for online tasks, offline for offline tasks
  - Online format is hardware-friendly, not user-friendly
- Disadvantage – calibration and alignment not very centralised
  - Possibility to redo these in analysis each time
  - Potential for “my” calibration to given different results from “your” calibration
  - Need to ensure (=force) people to work in common calibration/alignment framework; not always popular with physicists

Conversion of raw to LCIO

- I see four(ish) levels of conversion
  1. Direct bit-for-bit copy
  2. Intelligent reformatting
  3. Option 2 plus filtering and “digital interpretation”
  4. Option 3 plus “analogue interpretation”
- In my opinion, option 1 is not worth considering further
  - Have to reimplement whole raw data access structure in LCIO framework or use online code, losing big advantage of separation of the two
  - Completely unsafe under class changes so strongly couples the two; code releases to all users need to be coordinated with online firmware changes
  - User has to unpick hardware format every time data read in; will do the job needed for option 2 anyway, but probably not in a centralised way
- Option 4 is effectively back to the original computer model
  - Still doable (but have lost two months of work in this direction)
- Let’s look at the other options in more detail…

Option 2 – Intelligent reformatting

- Keep same information content as raw data but in a much more sensible format
  - Remove error-prone bit unpacking, separate ADC values from trigger counters, etc.
  - Some work required but needs to be done anyway so centralise
  - Immediate-term advantage; don't need to do all subrecords initially
- Reformatting breaks direct dependence between online and offline formats
  - Unchanged LCIO objects even if (when!) raw data changes dramatically
  - Uniform analysis of whole dataset; may require remaking LCIO with new classes but can do complete dataset
  - Allows repairs to data; e.g. currently missing ECAL stage settings from raw data; written in logbook by hand but can be added to LCIO data
- No other infrastructure needed
  - As (eventually) all data copied, no database access, etc, needed

Option 3 – Digital interpretation

- Effectively do any processing which is “digital”
  - I.e. things which can be done once-only (in principle)
- Main ones would be
  - Channel mapping; readout-to-physical ordering
  - Filtering of bad events (trigger timeout, readout failure, etc)
- Possibly also
  - Strip pedestals and slow data into database
- Would require a lot more infrastructure in place before starting
  - Databases for mapping, possibly also pedestals, slow data
  - Filtering cuts; may depend on data quality?
  - Could make this complicated for Grid-ification
- Could migrate from option 2 to option 3 with time?

Other factors

- Must have capability to reprocess all raw to LCIO several times
  - Mistakes, new information, class changes, LCIO new functionality/releases
- Must process to LCIO “immediately” (not clear to me why…)
  - Although DAQ can check data quality online
  - More sophisticated analysis access needed quickly?
  - Prefer not to need human intervention; implies no analogue interpretation
  - To handle new 2GB run file every 10 minutes, need significant CPU
- Want lightweight conversion job
  - Redo whole dataset (~10TB) within e.g. one week
  - Requires farm (DESY, RAL, LeSC, FNAL?) not single DAQ PC
  - Data backup from DAQ PC required anyway
- Outside of the DAQ PC, the online software only lives at these major computer farms
  - Experts only J

IHMO…

- Options 2 and 3 seem to be feasible/sensible
  - Whichever option, huge amount of other work to be done downstream
- If we want to do this “properly”
  - Pedestals stripped off to database
  - All slow and configuration data stripped off to database
  - Mapping, calibration, alignment calculated and entered into database
  - Automatic access to correct values for any event
- This is no time to duplicate effort by reinventing the wheel
  - Concentrate effort on missing parts
- Beware: “properly” can be the enemy of “good enough”
  - This is only a beamtest
  - The ECAL has already lost several months because of this; LCWS05 is coming very soon and we are up to 150GB and counting…