Data AcQuisition for COMET Phase-I

Phillip Litchfield, UCL
for the COMET DAQ group

UCL

DESIGN CONSIDERATIONS

Design Considerations I

The DAQ for COMET Phase-I is in fact two systems: one for for the Straw tube tracker / Ecal (**StrEcal**), and one for the Cylindical Detector (**CyDet**).

- There is some overlap, as some of the smaller systems such as the X-ray monitor will be used by both DAQs

For this reason, and to minimise work, the two DAQs should be as similar as possible

We do not have the expertise or personnel to create a DAQ from the ground up, therefore a conservative system using as much 'consumer' parts is desirable.

↳ **DAQ system will be based on MIDAS framework**

MIDAS Framework

- General DAQ framework designed at PSI and TRIUMF.

- Primarily intended for smaller scale ‘Nuclear physics’ experiments that use off-the-shelf hardware such (e.g. NIM modules, VME)

- Has been used by the ND280 detector of T2K → **Demonstrated that it can be used in larger-scale experiments**.

- Some differences with the original usage case are apparent, notably that the front-ends in HEP experiments are typically bespoke hardware (not VME controllers).

Therefore ND280 usage serves as important reference.

Other medium-scale experiments (e.g. g-2 at FNAL) also coming online that use MIDAS, and can be used as reference.

Design Considerations II

The hardware front-ends are being designed by the detector groups.

- We have some input into these designs, but must mostly consider them as constraints of the system.

The majority of detector channels will be read out via 2 front-end boards:

- **ROESTI** (mostly StrEcal, 16 channels/board)
  - ROESTI boards can be daisy-chained.
- **BELLE-II RECBE** (mostly CyDet, 48 channels/board)

Both boards can readout via SiTCP link which is a hardware TCP implementation.

- **Communications with the front-end can use standard protocols/libraries on the DAQ side**

UCL

USE OF MIDAS, DAQ OVERVIEW

Interfacing to MIDAS

MIDAS has many drivers available for standards like VME, but cannot not natively communicate with our readout cards. There are various approaches to solve this.

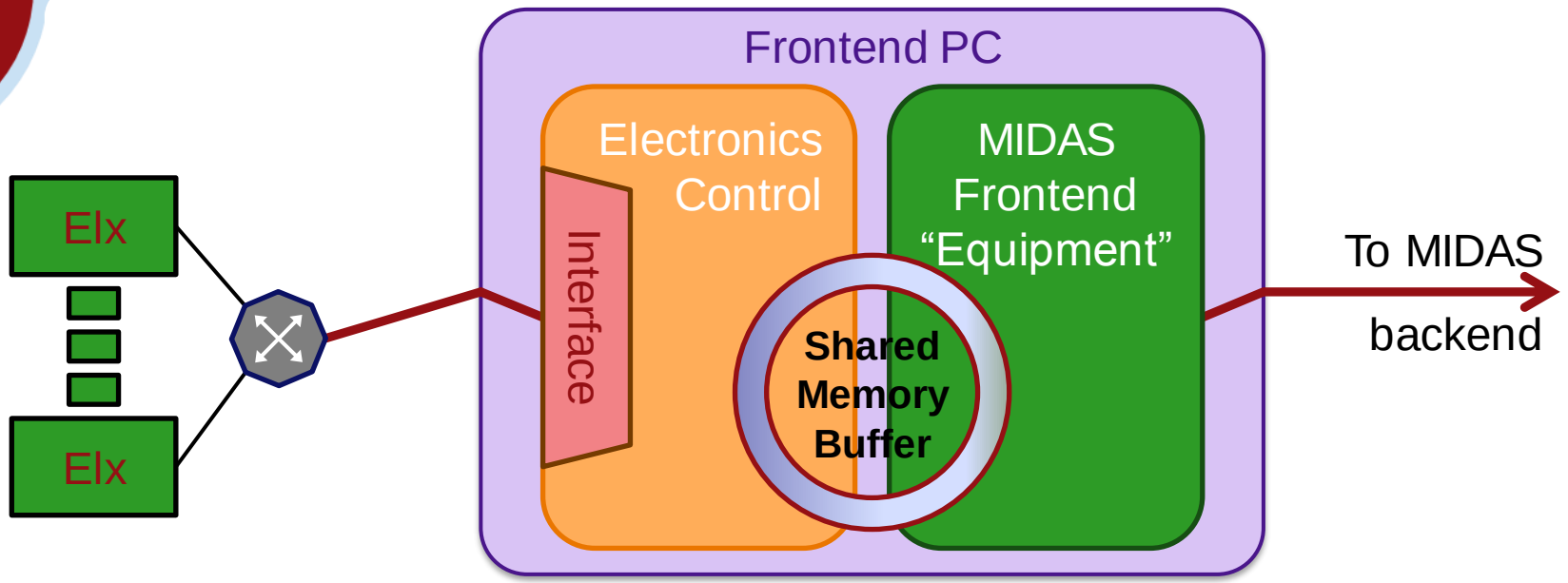
We chose to have PCs interpreting between the frontends and MIDAS. On each PC run two processes, which communicate using a shared memory buffer. [See next slide]

Alternative approaches are possible; for example having the MIDAS frontend implemented in firmware on an intermediate PCB.

The decision is partially informed by ND280 experience, where three approaches were in use. Most important factors are:

- Relatively cheap and easy – PCs are commodity hardware
- Can be scaled to channel counts quite easily, simply by adding more PCs (and switches in our case).

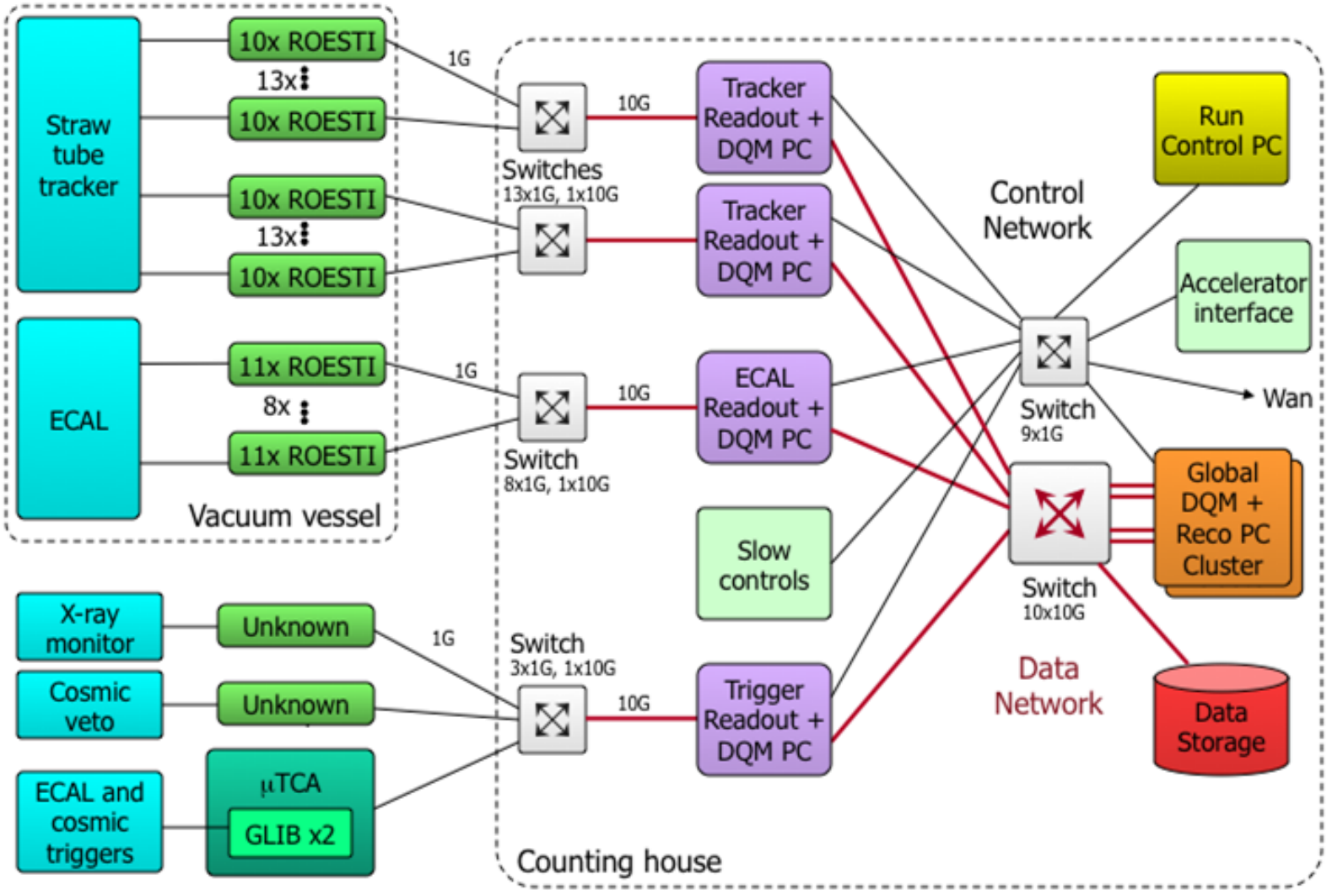
Interfacing to MIDAS



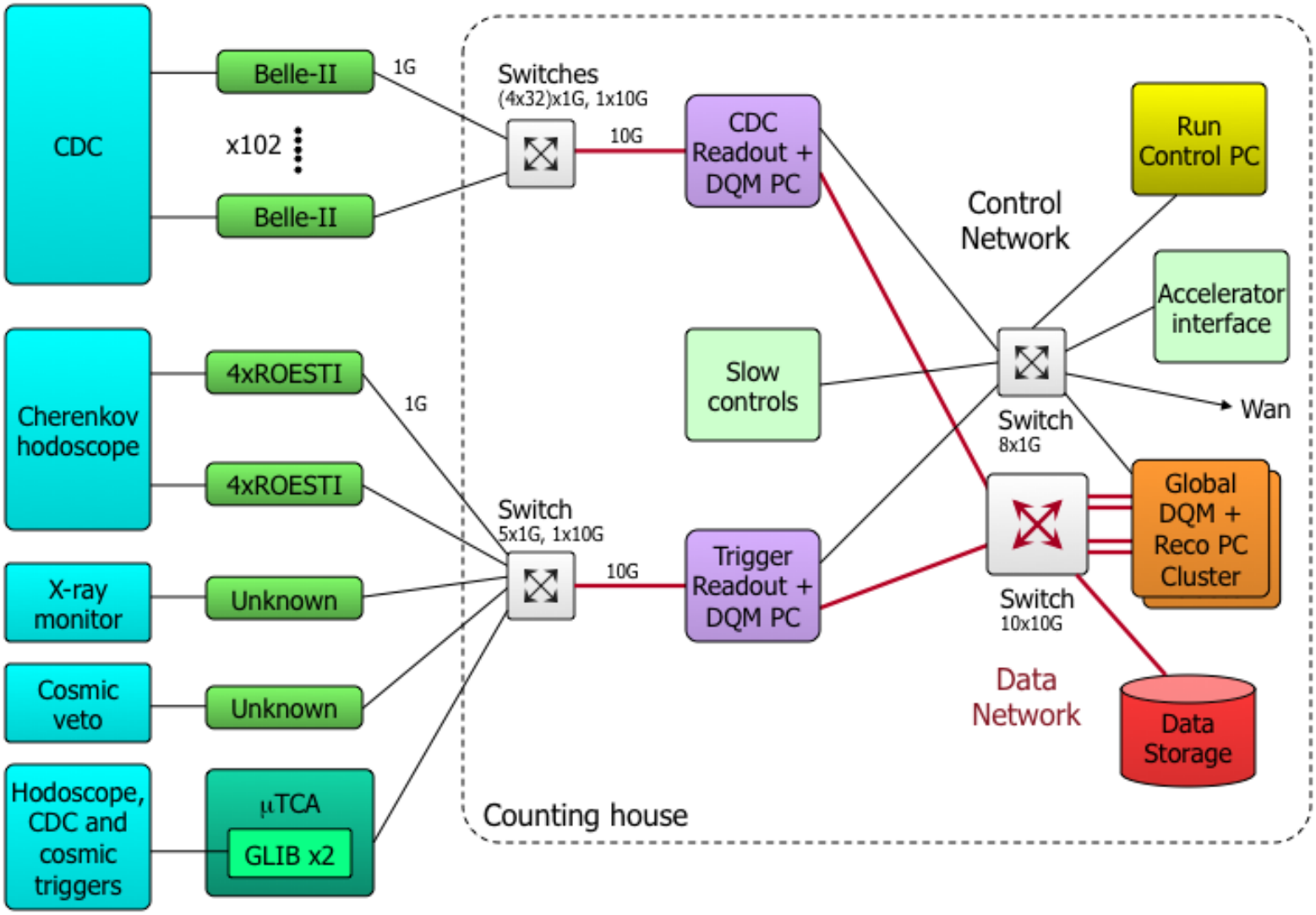
Each PC connects to several cards via N1 Gbit/s to 10Gbit/s switches.

- One process communicates with the electronics, providing TCP interface & higher level functions, such as aggregation.
- The second process communicates with the MIDAS backend. **As far as MIDAS is concerned each Frontend PC counts a single piece of equipment**, even though it reads out many channels.

StrEcal DAQ overview



CyDet DAQ overview



UCL

DATA RATES & BOTTLENECKS

Data Rates

The rate at which event data will be generated will be quite significant given the scale of the experiment and the resources available. To estimate the data rates (D), we need to estimate the trigger rate (R_t) and the event (readout from a single trigger) size S_t:

$$D_{(\text{short})} = R_t \times S_t$$

However the spill structure is such that we have a **2.93s SX spill every 6s**, so on longer timescales the average rate is reduced by this duty factor (0.49):

$$\langle D \rangle_{(\text{long})} = 0.49 \times R_t \times S_t$$

- The former quantity is of importance for the front ends where there is little buffering.
- But for the backend, where buffering on the order of 1s is available, we need only to handle the lower averaged rate.
- This assumes a low rate of out-of-time calibration triggers

Data rates in the StrEcal

For the StrEcal the most important trigger originates from Ecal clusters. The rate of these triggers can be tweaked by raising the energy threshold of these clusters.

- Estimate that we can tune this trigger to a rate of 1kHz

Readout from trigger dominated by the Str (1 Mb) and ECal (750 kb).

- (Other subsystems will generate about 80 kbit total)
- Both use the ROESTI: **waveforms at ~1GHz and 12 bits/sample**
- Expect that noise can be tuned out, so **data size primarily depends on number of tracks**, not number of channels
- Ecal expects a few channels per particle & O(100) samples
- Str expects ~20 channels per particle & O(10) samples
- For *individual* frontend PCs expect $\boldsymbol{D^{FE} \sim 50\ \mathrm{MB/s}}$
- For the backend $\boldsymbol{\langle D \rangle^{BE} \sim 108\ \mathrm{MB/s}}$

Data rates in the StrEcal

For the CyDet the most important trigger originates from the trigger Hodoscope. Can't tune this so easily—set by geometry and the beam particle trajectories

- Estimate that the design will result in a rate of up to ~30 kHz

The BELLE-II output for a channel is a simple **Channel / ADC / TDC triplet (16 + 32 + 32 = 72 bits)**

- Overall rate from BELLE-II cards is small; only 1 Frontend PC
- Estimate equal contributions to hit rate from tracks and noise
- **GLIB system** is a large contribution to event size in this case
- Overall get about 80 kb per event, more than 50% of which from GLIB
- At 30 kHz: CDC frontend sees $\boldsymbol{D^{FE} \sim 80\ \mathrm{MB/s}}$
- And the backend sees $\boldsymbol{\langle D \rangle^{BE} \sim 180\ \mathrm{MB/s}}$

Bottlenecks I

Even in Phase-I the data rate will be quite high. The major concern in operating the DAQ is that the system will be saturated, forcing us to use some throttling of triggers.

The bandwidth of the **front end** of the DAQ is **not** expected to be a problem.

- The first potential bottleneck is the front end 'fan-in' switches which consolidate up to 13× 1Gbit/s links into one 10Gbit/s link.
- However the data rate across these links is estimated to be of order 0.1Gbit/s, so the output should not be saturated even if a non-trivial fraction off packets need to be resent.

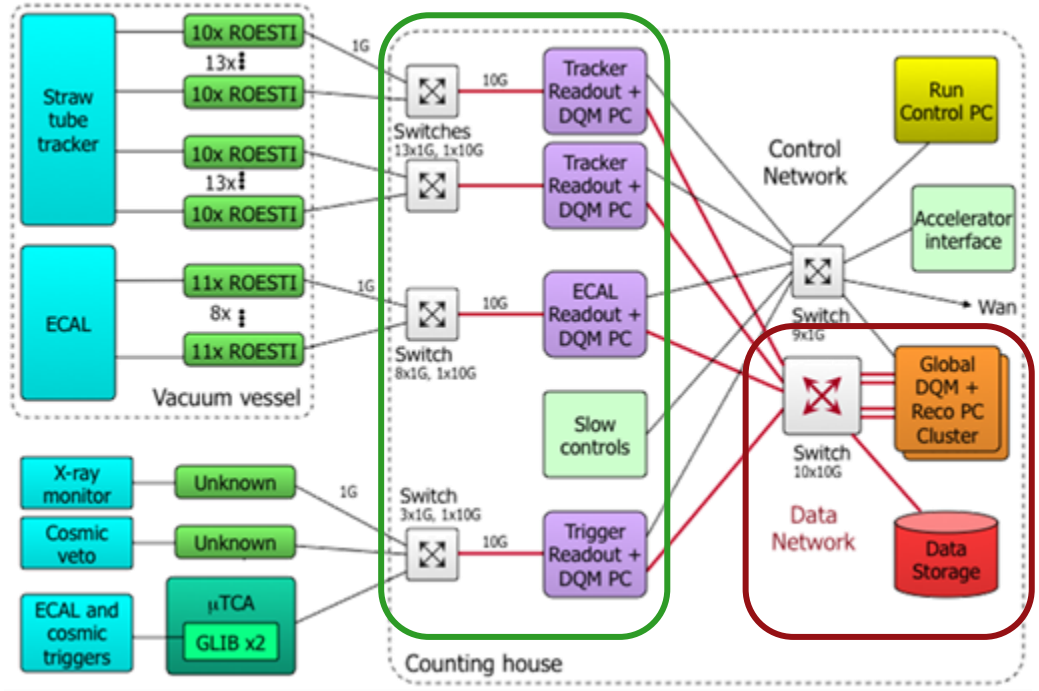
The **Frontend PCs** need to be able to handle input at a few Gbit/s. This likely places **some constraint** on the amount of processing they can do, but is not expected to be critical.

Bottlenecks II

In MIDAS, the equipment (Frontend PCs) assemble data into fragments and deliver it to a master **Event Builder** process, which runs on a backend PC.

- The input to this PC will be consolidated from the frontends as a single 10 Gbit/s link
- **Frontends** can be made more parallel as necessary…

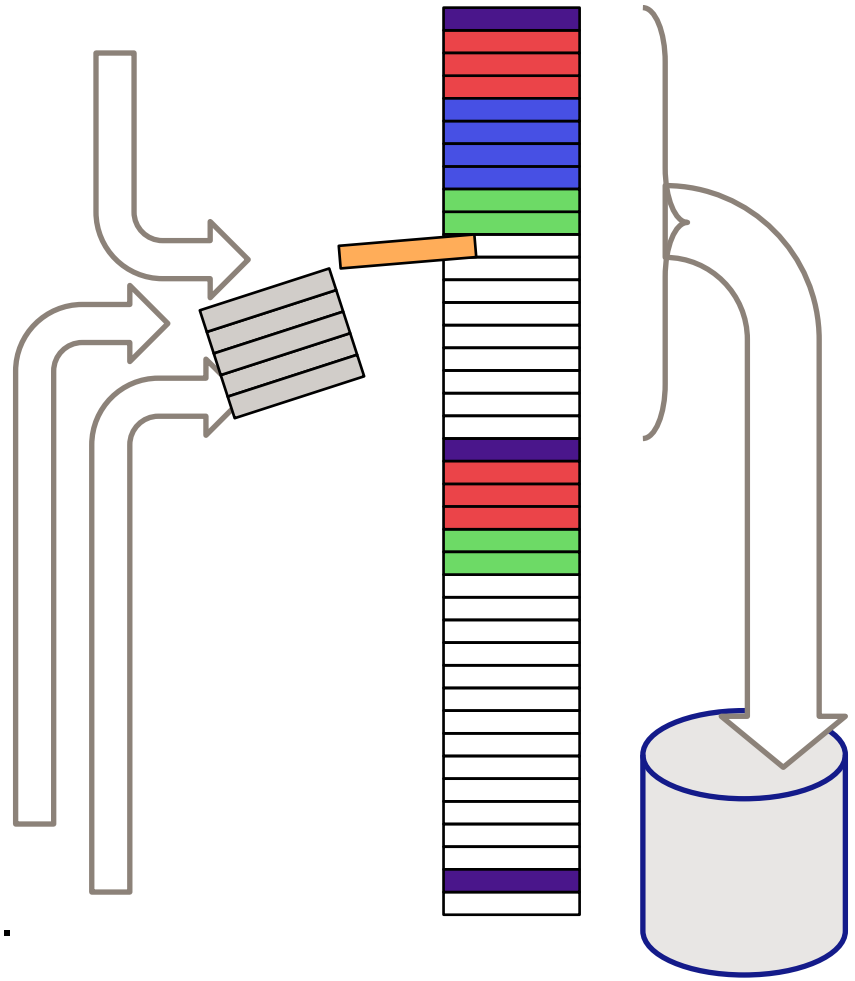
  …but the **backend** cannot [unless the DAQ is split into independent parts.]
  - Note that the CyDet and StrEcal are already independent DAQs



Bottlenecks III

Starting when the first fragment for an event arrives, the event builder maintains the event in a shared memory buffer, until all expected fragments arrive.

- Smoothes data flow between online and disk
- Potentially requires a large buffer
  - Developer recommends “5~10s or 10~100 events of buffer ” (!)
- Large latency skews are a risk
  - But for us ‘large’ means ~ 0.1s
- In practice we will want to buffer for O(3s) of events
- Require memory buffer of at least $2 \times 3s \times 180\ \mathrm{MB}/s \sim 1\mathrm{GB}$
  - Large, but feasible for a x64 system.



Bottlenecks IV

The most concerning bottleneck is writing output to disk.

- This is done by a logger process, which will run on a dedicated PC, connected to the same switch as the Event Builder.
  - With 'normal' disk technology (SATA 6Gbps) we can theoretically get a throughput up to ~4 Gbit/s… ***but***
  - Current HDDs manage a sustained average (block) write of around 500~1200 Mbit/s
  - Minimum speeds can be lower, 400~1000 Mbit/s

This is **only just enough** for our expected data rate.

- It will be important to get good quality hardware (and monitor performance over lifetime)

However:

- Technology improves (High volume SSDs by then?)
- Can use hardware tricks, e.g. RAID0
- Must also consider connection, esp. for swappable drives.

UCL

INTEGRATION WITH OFFLINE

Slow control & readout

Slow control readout is supported by MIDAS, although it has some limitations, e.g. not synchronised with data taking.

Configuration settings will be recorded in the data stream at the start and end of runs, and more frequently in the case of long runs.

Monitoring readout can be stored as a MIDAS history file. These can be distributed along with the data files, or through some other mechanism.

- Summarisation and conversion of files to a ROOT based format (or preferably) offline database seems the most practical route.
- [Probably don't have enough tech. support for real-time DB writing]

- This must also include information derived from accelerator.

Currently do not have much detailed plans in this area.

But expected not to be as critical as (e.g.) fast readout.

Archival format

The DAQ will write out data in MIDAS file format, probably compressed (reduces disk usage, but higher CPU on logger)

- Compression is done by MIDAS, but uses **standard zlib** (gzip)
- This can be used as **archive format**.
- Special 'events' can be used to store configuration (Online Data Base) data structure.
  - Care needs to be taken with configuration variables not in ODB

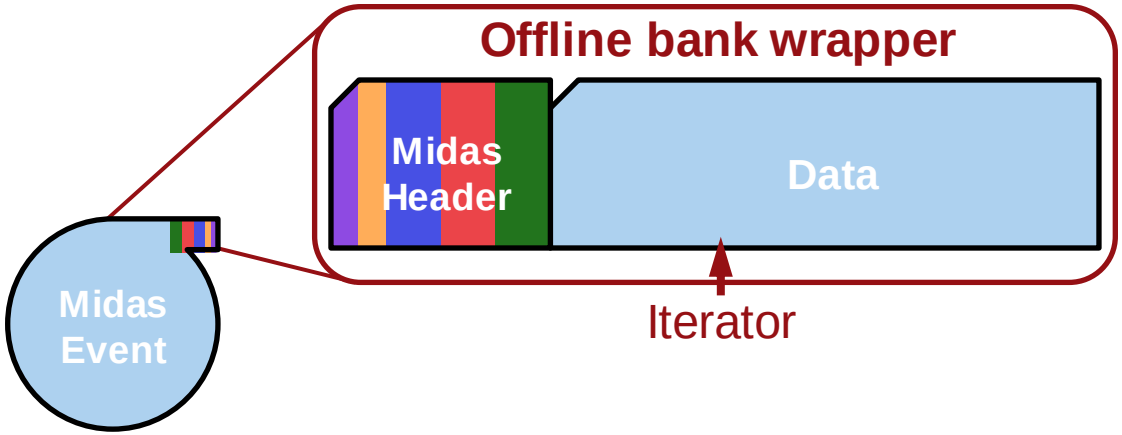
- First task of Offline software is to decode ('Unpack') MIDAS files into an object-like ROOT structure.
  - **Principles**: minimize processing before archival; offline system should accommodate needs of online more than vice-versa
  - Output is not necessarily saved to disk, but saved and in-memory versions are the same.
  - **The unpacked format is also the target 'equivalent format' for MC.**

Reading data offline

The offline interface is based heavily on **ROOTANA** (a supplement to MIDAS developed at TRIUMF), and extensions written for ND280.

MIDAS events are sequences of banks. The format of banks is expressed through simple **shared C headers**.

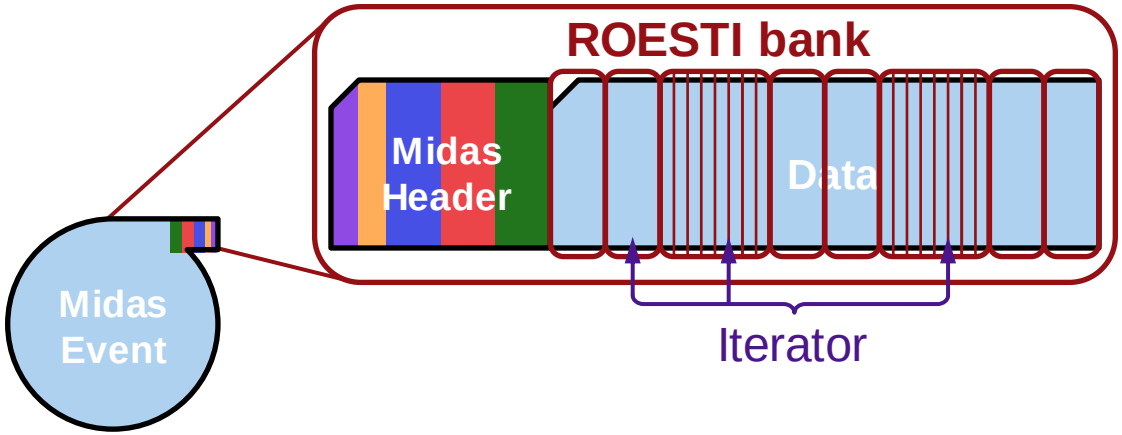
- Only point of dependence between offline and online systems.
- At the lowest level the offline simply loads in MIDAS bank and wraps their payload as an iterable container of bytes.



Reading data offline

- User requests a particular type of data (e.g. ROESTI output) with a template method.
  - The offline looks through the event for the next bank of that kind.
- Behind the scenes the flat container is *promoted* to the specialised version, and returned to the user.

  [promoted == replaced, but leaving the data untouched]
- Specialised iterator objects gather the data in whatever form makes sense to reconstruction code.
- **Flat data structure now looks like container of 'proper' C++ objects.**



Reading data offline

Finally, the promoted banks are copied into real offline-native containers.

- Objects contained are self contained representations of detector activity.
- Representation is typically a generalised [Location, Charge, Time] triplet where
  - **Location** is a index for a readout channel, or logical group of channels
  - **Charge** might be a single ADC value, or a waveform
  - **Time** would normally be a TDC value relative to some clock reset

- **Simple, known, transformations of the data allowed**
  - e.g. aligning waveforms to TDCs.
  - *Even allows for on-the-fly encoding changes from the online*!
  - More complex transformations (i.e. calibration) are done later.
- **This format is the basic target for MC simulation.**

NOTES, EXTRA SLIDES

TODO:

Rate estimates;
Trigger rate is ~1/1000th of bunch rate… is this correct?

Dead time & buffering?

~~!(Slow control)~~

~~Offline interface~~

Make pretty

Implications

- SiTCP: Using TCP means data packets will be resent if not received correctly. This slows down transmission but makes it more robust. Resending should hopefully be rare because we are using small dedicated networks – not (e.g.) communication over the facility LAN.

- Using a single PC to read out several cards via a switch may introduce a CPU bottle neck. This can be averted using more PCs and (possibly more switches) as necessary. Testing of this bottlenecks should be done once a relatively complete system is available.

- What happens if CyDet trigger is higher? Possible even saturate from GLIB?!? Rates quoted in Kyushu seem very scary. If we had to prescale, is that bad? How bad?

- What is bits/sample of DSR4?

- Do we need any need out of time triggers? What is the rate (Hz) & bit rate of those?