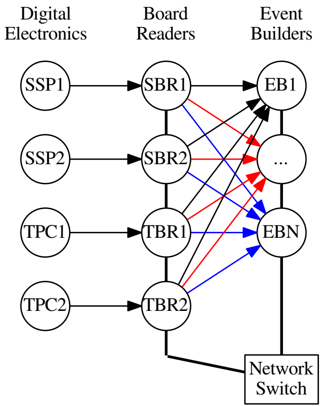
Online/Offline Buffer Options for protoDUNE/SP

Brett Viren and Maxim Potekhin

Physics Department

BROOKHAVEN NATIONAL LABORATORY

Model of Upstream DAQ



- DAQ as layered, directed acyclic graph.
- Figure glosses over distinctions in RCE + electronics (sorry).
- Board Readers route fragments from one trigger to Event Builders
- Event Builders concatenate fragments into contiguous readouts.
- Rates, bandwidths, processing time, switches, NICs all play major role in shape and size!

Online/offline buffer scope starts somewhere soon after EB's.

Data Scenario Implications on Buffer

Required to have **3 days buffer**.

Recent **upward revision** of *Data Scenarios Spreadsheet*:

- DocDB 1086-v6 (let's put any and all updates here)

→ 25-50Hz, 5ms, 6APA, 2-4× compression, 25-50M events.

Implications on **buffer requirements**

- 25-50TB buffer disk,
- 30-60 parallel HDD writes,
- 1.5-3.0 GByte/sec throughput.

Back of the envelope estimate

If assume **1Gbps NICs**:

- **25 concurrent network streams**
- NIC is bottleneck so **minimum of 25 hosts** writing **2-3 disks**
- Beneficial side-effect: leaves plenty of idle CPUs to load up doing other, useful and prompt tasks.

If assume **10Gbps NICs**:

- **3 concurrent network streams**
- SATA is bottleneck so **minimum of 6 hosts** writing **10 disks**
- Hosts: fewer, bigger, more expensive, less "off-the-shelf".
- CPUs loaded just doing I/O, little room for other processing.

A **quantitative investigation** based on assumptions of rates, bandwidths, processing times, etc is being developed with the **Ersatz Simulation** package. Links: DAQ Sim presentation, GitHub

Buffer Design - Two Options

UOOB Unified Online/Offline Buffer hosts

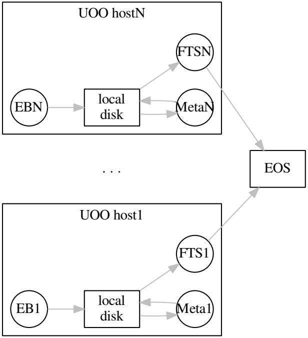
$\rightarrow$ Shared hosts, local disk for data hand-off.

DOOB Dedicated Online/Offline Buffer hosts

$\rightarrow$ Separate layers, network for data hand-off.

Unified Online/Offline Buffer

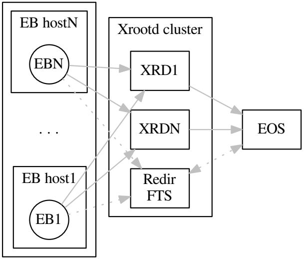
Online/offline interface: file-hierarchy on shared local disks.



Each UOOB computer must host:

- DAQ Event Builder node.
- File management glue scripts.
- File metadata producer.
- FTS instance.
- Logic to handle DAQ/FTS disk contention.
- Online/offline contract on detailed file locations.

Dedicated Online/Offline Buffer



Dedicated "layer" in distributed graph

- Each EB needs to know only about XRD redirector.
- Transfers still load-balanced.
- Single FTS instance, share XRD redir host.
- Direct XRD transfer to EOS, governed by FTS.
- Decoupled Online and Buffer specs.
- XRD_i nodes run metadata job after receive file.
- Interface specification = XRD URL namespace

Some pros/cons of each

UOOB:

- **pro** one less overall layer, file system hand-off maybe more familiar than XRD (?).
- **con** tight coupling in design and procurement, denser CPU requirement, O/O interface protocol more complex. Multiple FTS (not big problem), FTS→EOS mediated transfers.

DOOB:

- **pro** more distributed, leads to naturally more available CPU, decouples design and procurement, more fault-tolerant, O/O interface protocol simple. Direct XRD→EOS FTS-initiated transfers (EOS is native XRD).
- **con** requires extra layer, DAQ/EB needs XRD client lib (but not ROOT) or must use `xrdcp` unix command and thus its own local storage

Take Away

- We are looking to quantitatively understand the protoDUNE/SP online/offline needs.
- Likely major decision: 1Gbps vs. 10Gbps NICs
  - → want bottleneck at network or CPU/DISK?
- Two design options exist:
  - uoob concentrated complexity, tightly coupled, one less layer.
  - doob simpler, more distributed but one extra layer, requires XRD.

What are our external constraints? (how much $$$?)