Evaluation of a high-performance storage buffer with 3D XPoint devices for the DUNE data acquisition system

Adam Abed Abud (CERN / University of Liverpool)

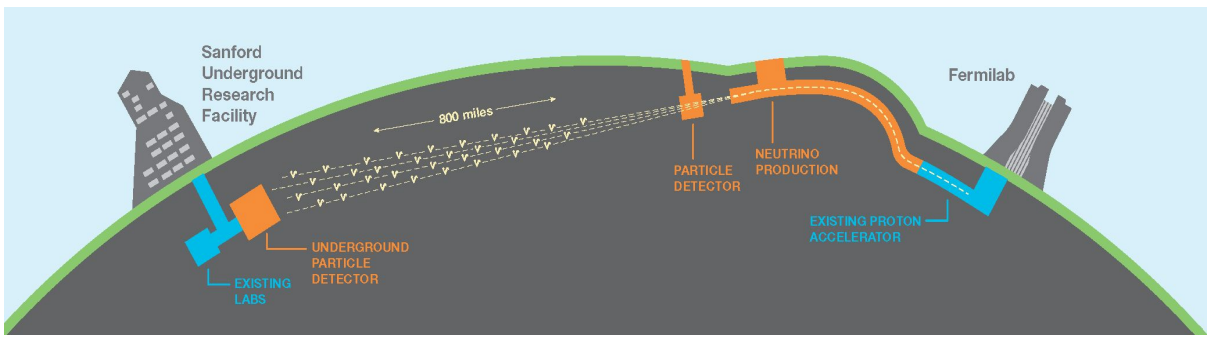
25th International Conference on Computing in High-Energy and Nuclear Physics
May 18, 2021

Outline

- The DUNE experiment
- DUNE Data Acquisition System
- Synthetic benchmarking results with the Micron X100
- The MiniDAQ application
- Integration with the DUNE data acquisition system
- Conclusion and outlook

The DUNE experiment

- The DUNE experiment is a leading international experiment for long-baseline neutrino oscillation studies, neutrino astrophysics and proton decay searches
    - Data taking expected for 2027
- Near detector close to the source
    - Accelerator (FNAL) generating an intense neutrino beam
- Far detector located 1300 km from the source and 1.5 km underground
    - 4 cryogenic modules, each with 17 kton of liquid argon
    - Detectors made up of a Time Projection Chamber and photon detector
    - Trigger and Data Acquisition:
        - 4 independent instances (one for each module), synchronized to a common clock



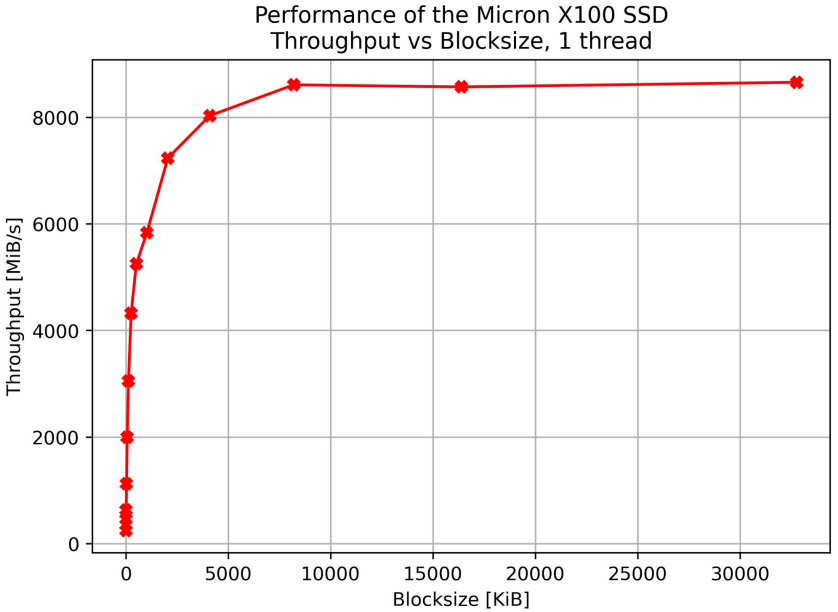
DUNE Data AcQuisition system (DAQ)

- DUNE uses a continuous readout system for the TPC detector
  - **Data throughput** for each readout unit: approximately **10 GB/s** from 10 optical links
  - Total of 165 readout units per detector module (X4 detector modules)
  - Total of ~1.5 TB/s for each detector module
- Role of Data Selection is to combine subset of readout data into time windows of interesting signals:
  - Time “window” can vary from < 1 ms to ~100s
  - Data size ranging from few MB to ~150 TB
- To avoid oversizing the resources in the DAQ architecture, a **local storage** system is used in the Readout System
- **Physics example:** storing supernova neutrino burst (SNB) events
  - One of the physics goals of DUNE
  - **Complex detection**: events with rare, low energy and distributed signatures
  - Long trigger latency
  - **DAQ requirements**:
    - Local storage capable of sustaining a stream of data for 100 seconds at ~10 GB/s
    - **Rare event:** no endurance problems for the storage system

Micron X100 as a high-performance buffer

Synthetic benchmarks

- The Micron X100 is an example of PCIe NVMe SSD based on the 3D XPoint technology
- **Goal**: evaluate the performance of the drives as a possible application for the DUNE local storage
- **Synthetic benchmarks**
  - Evaluate the sequential write throughput with a single thread
  - Max. write throughput achieved is **8.5 GiB/s**
    - Compatible with hardware specification



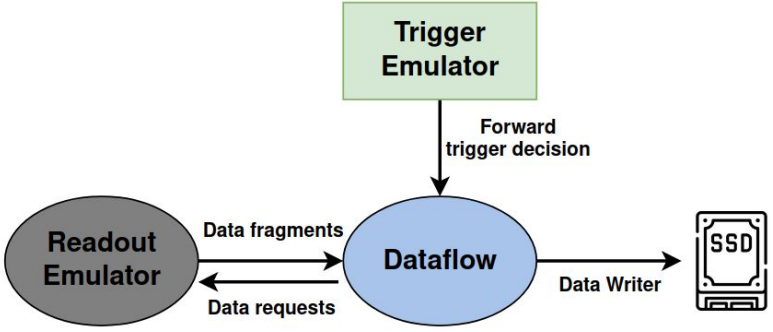
- Micron X100 device is capable of sustaining 85% the throughput of a single readout unit
- Promising technology:
  - **Example**: combine 3 Micron X100 SSDs for 4 readout units

Integration with the DUNE Data Acquisition System

The MiniDAQ application

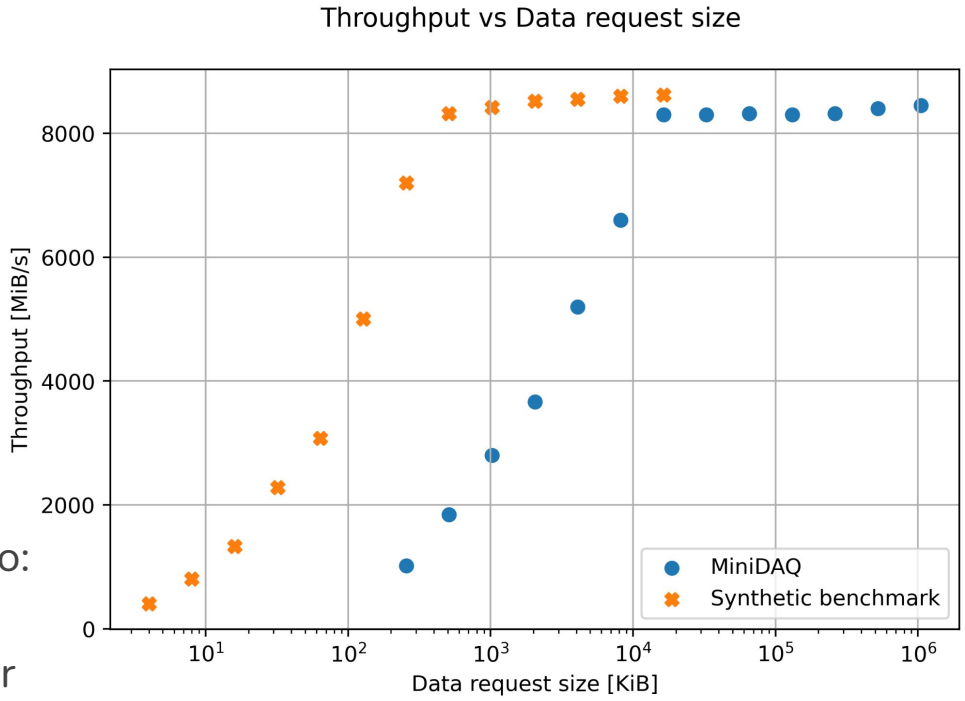
- Evaluation with a more **realistic workload**
- Use of a test application (MiniDAQ) that contains the most relevant features of the DUNE DAQ system
  - **Readout Emulator**: data input from 80% of the throughput of a single DUNE readout unit
    - Constraint due to the maximum write throughput of the drive
  - **Trigger Emulator**: responsible for the data selection
  - **Data Writer**: implements the interface to the local storage

- Realistic emulation of a single DUNE readout unit



Performance results with the MiniDAQ application

- In the MiniDAQ tests, selected data are sent with a configurable request data size and trigger rate
- The data request rate was chosen in order to maximise the throughput in a stable system
  - Running with no trigger inhibits or backpressure
- Similarly to the synthetic results it is possible to sustain in a stable condition more than 8 GiB/s
  - Use of data aggregation to achieve the highest throughput
- Differences with the synthetic results may be due to:
  - Additional operations (e.g. memory copy) for the interface with the Readout Emulator



Conclusion and outlook

- The Micron X100 is a high-performance storage device which can be an interesting fit for the DUNE data acquisition system
- Synthetic performance tests show that it is possible to saturate the bandwidth provided by the drive and achieve the nominal throughput of approximately 8.5 GiB/s
- The Micron X100 was also integrated in a prototype application for the DUNE DAQ
    - Performance tests demonstrated that a single drive can sustain steadily 80% of the traffic generated by one DUNE readout unit when using data aggregation
- Suitable technology for the DUNE local storage
- **Outlook**
    - Other suitable storage technologies will also be evaluated as a possible candidate for the DUNE local storage system
    - Investigation of streaming data directly from the Readout system instead of requesting data as it is in the current application

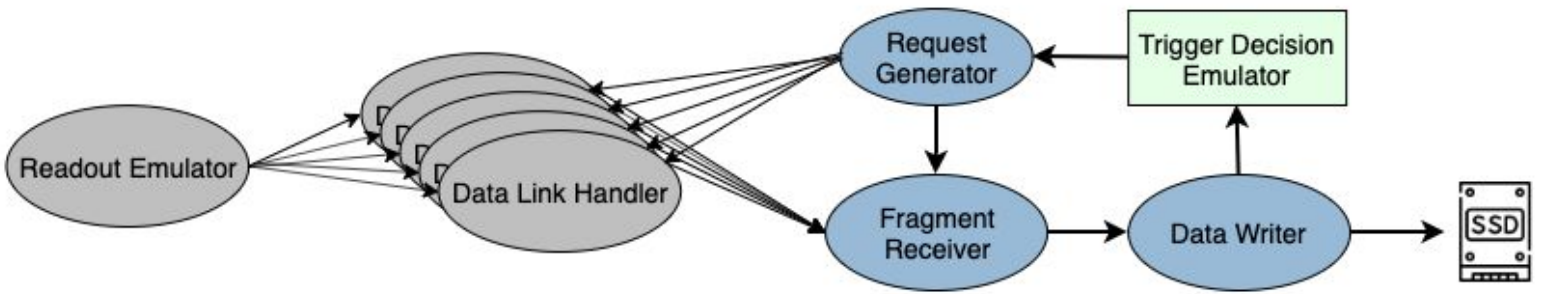
Thank you!

Adam Abed Abud

Further details

The MiniDAQ application

- Use of a test application (MiniDAQ) that contains the most relevant features of the DUNE DAQ system
  - **Readout Emulator**: data input from **half** of a DUNE readout unit
    - Data Link Handler: temporary buffering of raw data for each link. Throughput per link ~1 GB/s
    - 2 Readout Emulators to simulate 1 DUNE readout unit
  - **Trigger Emulator**: responsible for the data selection
  - **Request Generator:** Converts trigger decisions into data requests to be sent to the Data Handlers
  - **Fragment Receiver:** Receive the data from the Data Handlers and forward them to the DataWrite
  - **Data Writer**: implements the interface to the local storage



Integration with the DUNE DAQ

- The mechanism of the Data Writer process consists:
  - **Initialization**: creation of a memory aligned buffer
  - Setting up relevant CPU affinities
  - Write requested data if a trigger has been enabled
- An **inhibit mechanism** is also in place in case the Data Writer is not able to sustain the rate of incoming data
  - In this case a warning message is issued ("Data flow is BUSY")
- It is necessary to align the data extraction rate with the data production rate in order to avoid any data loss during operation
- The data writing is done by executing kernel I/O asynchronous operations (AIO library)

Algorithm 1: Data Writer mechanism.

```
initialize data store;
allocate mem-aligned buffer;
start data writer thread;
set CPU affinity;
while trigger_flag do
    receive requested_data;
    for fragment in requested_data do
        get ptr to fragment location;
        get fragment size;
        memcpy(buffer, fragment, size);
        flush_to_disk(buffer);
    end
    check inhibit(fragment);
end
```