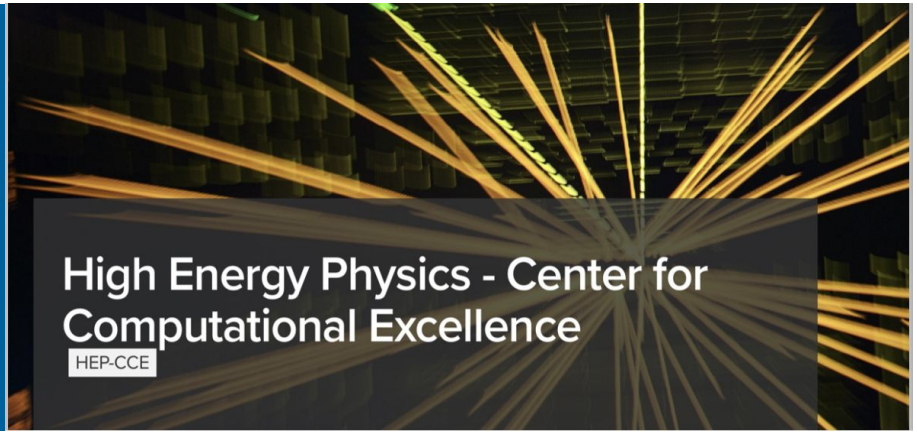
HPC FRIENDLY DATA MODEL



AMIT BASHYAL
On behalf of HEP-CCE IO/S
Argonne National Lab
Postdoctoral Research Fellow

AHM Meeting
Lawrence Berkeley National Lab

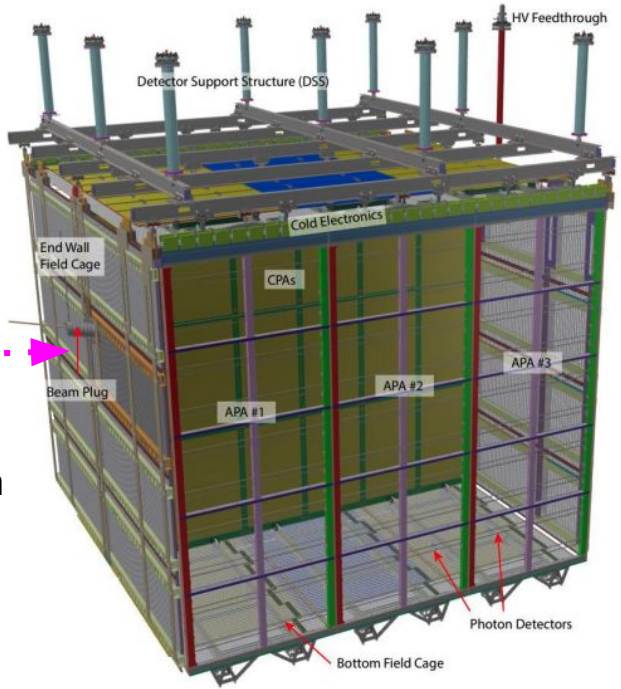
October 12, 2022

INTRODUCTION

- 2 aspects of the HPC friendly data model
  - Direct coupling of the [e.g. DUNE] data format with the HDF5
  - Efficient offloading for the GPU processing.
  - Link to the github repository.

- Using ProtoDUNE Raw Data for the test
  - DUNE being the future experiment, our experience will benefit the experiment.
  - Already using HDF5 to store their raw data and MC
    - Simple data model (easier to test)
  - Link to the (Proto)DUNE CDR.

Proto-DUNE



Beam Direction

Figure 1: ProtoDUNE-SP Internal Components

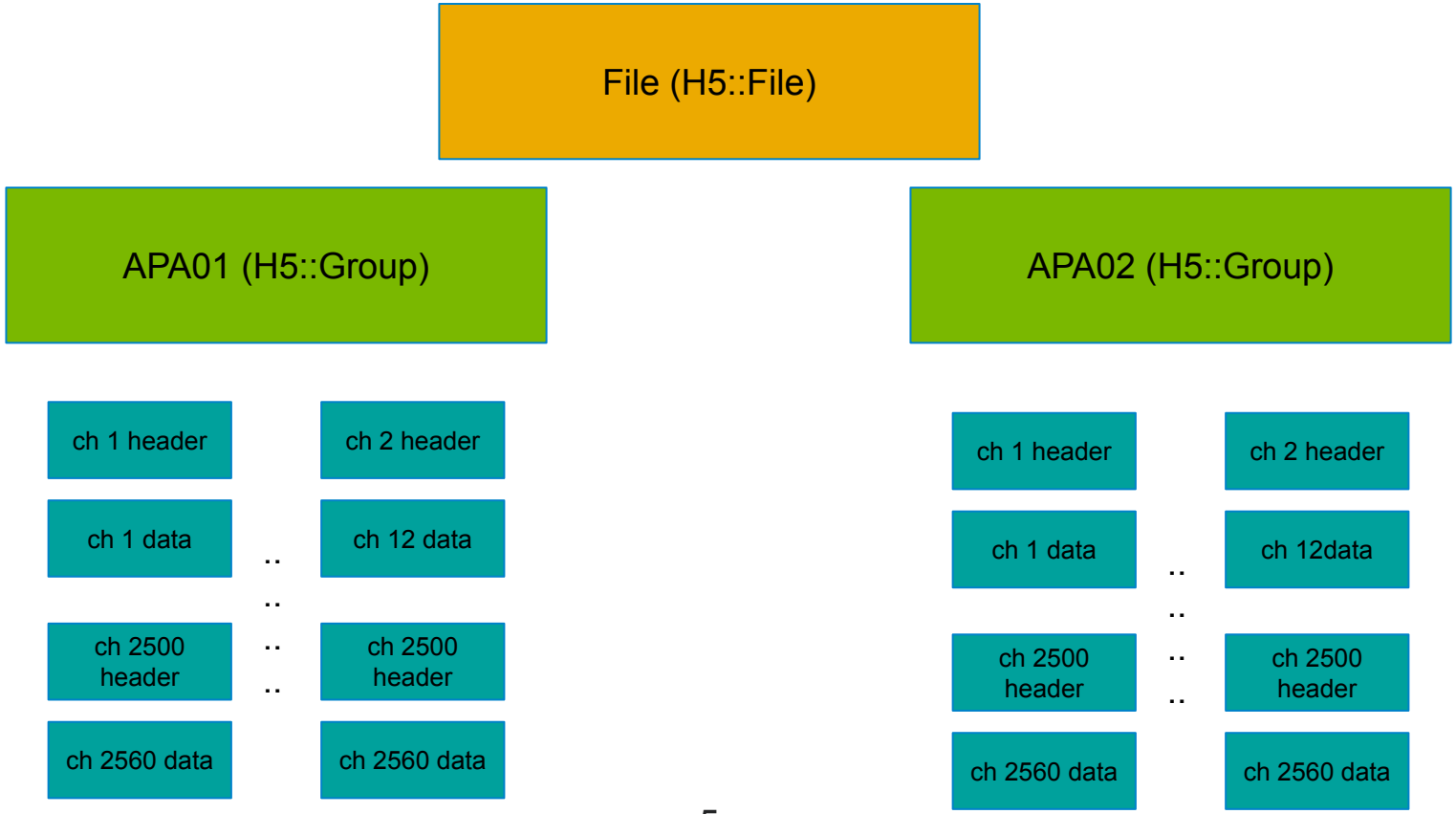
- ProtoDUNE LAr TPC currently operating at CERN
- Incorporates the full-size components designed for the DUNE Far Detector
- Aims to validate the detector and computational technologies for the DUNE Far Detector
- Quantify and reduce systematic uncertainties for the DUNE Far Detector by optimizing event reconstruction and PID technologies
- Detector being characterized with incident beam of known momenta and PID
- 2 APA (Anode Plane Assemblies) arrays
  - Each array has 3 APAs adjacent to one another.
  - Each APA has 2560 signal channels.
- More info on Proto-DUNE (Link)
- Second iteration of this detector has one additional APA array

Raw Data of Proto DUNE

The Proto DUNE DAQ (Raw) Data is stored in the HDF5 Format.

- For a given APA, each of 2560 channel readings are stored in individual data-sets (HDF5::DataSet).
- All of 2560 channel readings of a given APA are grouped together (HDF5::Group).
- For each channel reading data-set, a header file is written
  - Channel Number
  - Number of ADCs
  - Compression Status
  - ….And so on…

Data Structure of ProtoDUNE Raw MC



RAW (Header) Data of ProtoDUNE

```
GROUP "TriggerRecord00028" {
    GROUP "TPC" {
      GROUP "APA000" {
        DATASET "Link00" {
          DATATYPE  H5T_STD_I8LE
          DATASPACE  SIMPLE {}
        }
        DATASET "Link01" {
          DATATYPE  H5T_STD_I8LE
          DATASPACE  SIMPLE {}
        }
    }

DATASET "TriggerRecordHeader" {
      DATATYPE  H5T_STD_I8LE
      DATASPACE  SIMPLE {}
    }
  }
```

Data is the DAQ data copies of the fragments named as Linkxx.
Grouped for each APA

Data is reorganized to enable readout of whole channel during offline processing

```
GROUP "APA01" {
  DATASET "ChannelHeader_2500" {
    DATATYPE  H5T_COMPOUND {
      H5T_STD_I32LE "Chan";
      H5T_IEEE_F32LE "Pedestal";
      H5T_IEEE_F32LE "Sigma";
      H5T_STD_I32LE "nADC";
      H5T_STD_I32LE "Compression";
    }
    DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
    DATA {
    (0): {
         2500,
         2305,
         8.32,
         60,
         0
       }
    }
  }
```

MC is simulated as a single channel readout.
Header is **H5T_Compound** type.
Channel readout is **uint32_t type.**

Raw MC of Proto DUNE Content of HDF5 File

Channel Header

```
GROUP "7" {
   GROUP "APA01" {
      DATASET "ChannelHeader_2500" {
         DATATYPE  H5T_COMPOUND {
            H5T_STD_I32LE "Chan";
            H5T_IEEE_F32LE "Pedestal";
            H5T_IEEE_F32LE "Sigma";
            H5T_STD_I32LE "nADC";
            H5T_STD_I32LE "Compression";
         }
         DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
         DATA {
         (0): {
               2500,
               2305,
               8.32,
               60,
               0
```

```
DATASET "ChannelID_2500" {
   DATATYPE  H5T_STD_I32LE
   DATASPACE  SIMPLE { ( 60 ) / ( 60 ) }
   DATA {
   (0): 2542, 2964, 2042, 2994, 2197, 2852, 2226, 2929, 2288, 2773,
   (10): 2338, 2639, 2939, 2071, 2854, 2369, 2967, 2320, 2313, 2699,
   (20): 2885, 2771, 2519, 2748, 2590, 2533, 2288, 2967, 2049, 2249,
   (30): 2080, 2178, 2318, 2552, 2831, 2234, 2843, 2256, 2558, 2874,
   (40): 2223, 2659, 2780, 2083, 2782, 2508, 2672, 2040, 2139, 2234,
   (50): 2969, 2279, 2001, 2608, 2938, 2243, 2743, 2588, 2276, 2017
   }
}
```

Corresponding Channel Data

60 ADC readings only for illustration purpose

Simulation of ProtoDUNE Raw MC

- Fake simulation using toy MC.
  - HDF5 parameters available to optimize the I/O
  - Can affect the I/O time and file size

Chunk-Size	Write Time (seconds)	Read Time (seconds)	Size (MB)
6	36.35	25.81	317
60	6.17	2.70	153
600	3.34	0.60	133
6000	3.16	0.44	133

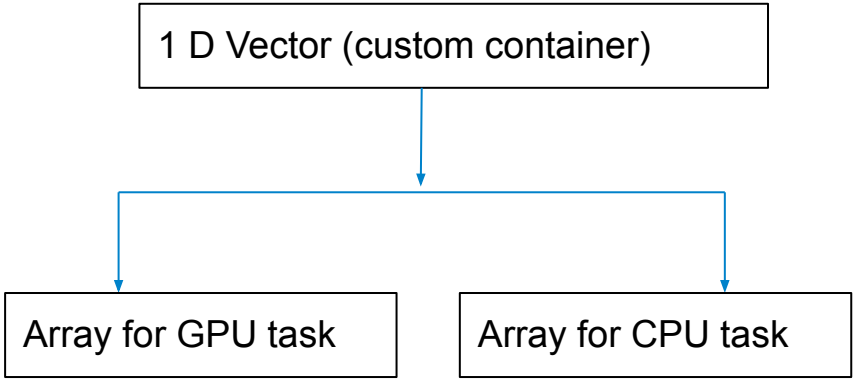
Table: 2 APAs (2560 channels each) with each channel writing ~6000 ADCs (unsigned integer 32 bits)
Basically for 6000 readouts, no chunking required.

Offloading Data into GPU/CPU

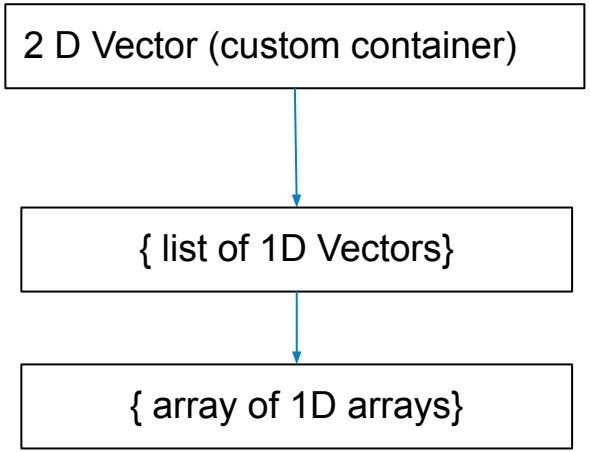
- Portable code that supports heterogeneous resources
  - Same function/task can be handled by both CPU and GPU
- Relies on CUDA, C++ libraries, tbb
- Offloading structure based on Athena::ATHCUDA
  - Written by Attila Krasznahorkay (Link to repository)
- Tests being done in the CORI machine.
  - CORI is scheduled to shut down by the end of this year
  - Will move the code development to Perlmutter

Data Models offloadable to the GPUs

- 1 D cpp vectors



2 D vectors



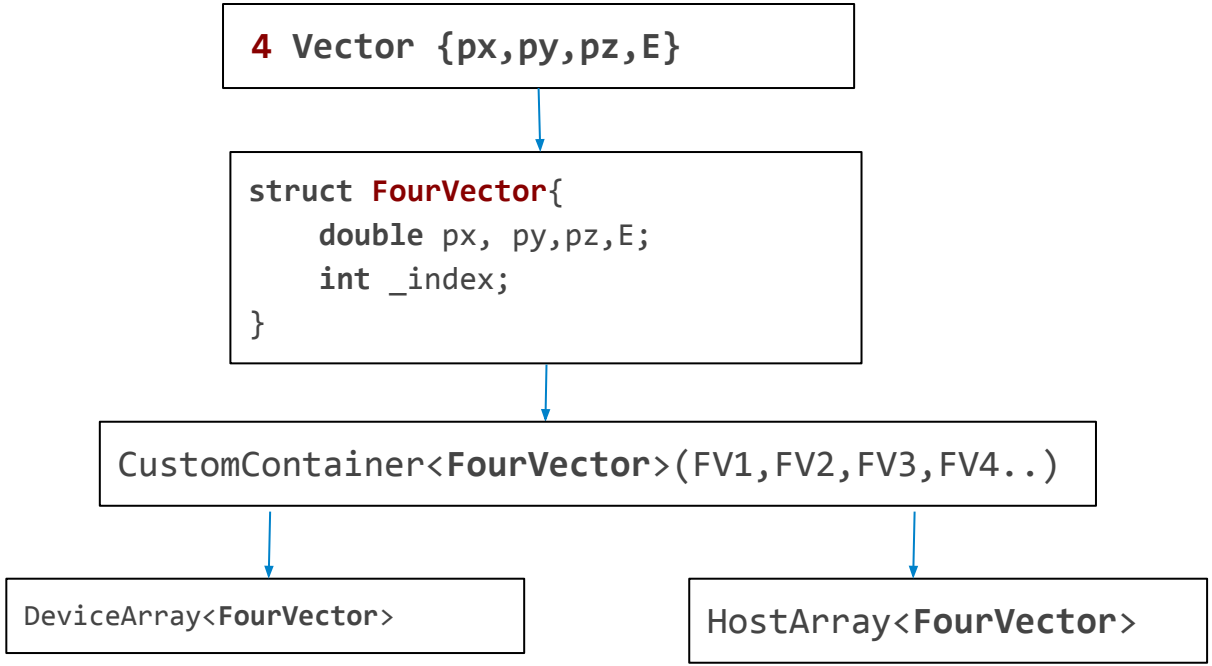
Data models currently supported

```
my1Darray* = {x,y,z......};

my2Darray** = {1Darray1*,1Darray2*,1Darray3*,......};
```

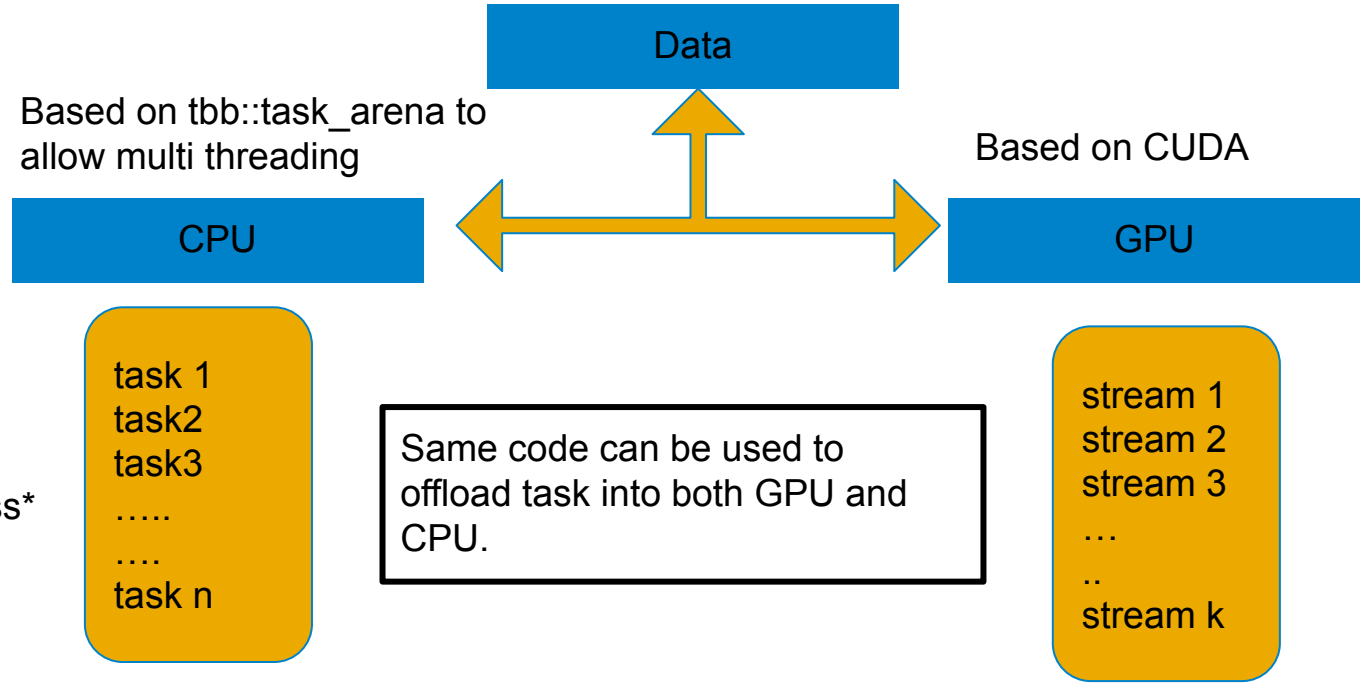
```
class test{
public:
    HOST_AND_DEVICE
    void operator()(uint32_t* a,uint32_t* out, int
arr_size){
       for(int i = 0;i<arr_size;i++)
        out[i] = a[i]*2.0;

    }

};     //1D Example
```

```
class test{
  public:
    HOST_AND_DEVICE
    void operator()(uint32_t** a, uint32_t**
out, int arr_size){
       for(int i=0;i<2;i++){
        for(int j=0;i<arr_size;j++) out[i][j] =
a[i][j]*2.0;
       }

    }
};
     //2D Example
```

Data models currently Supported

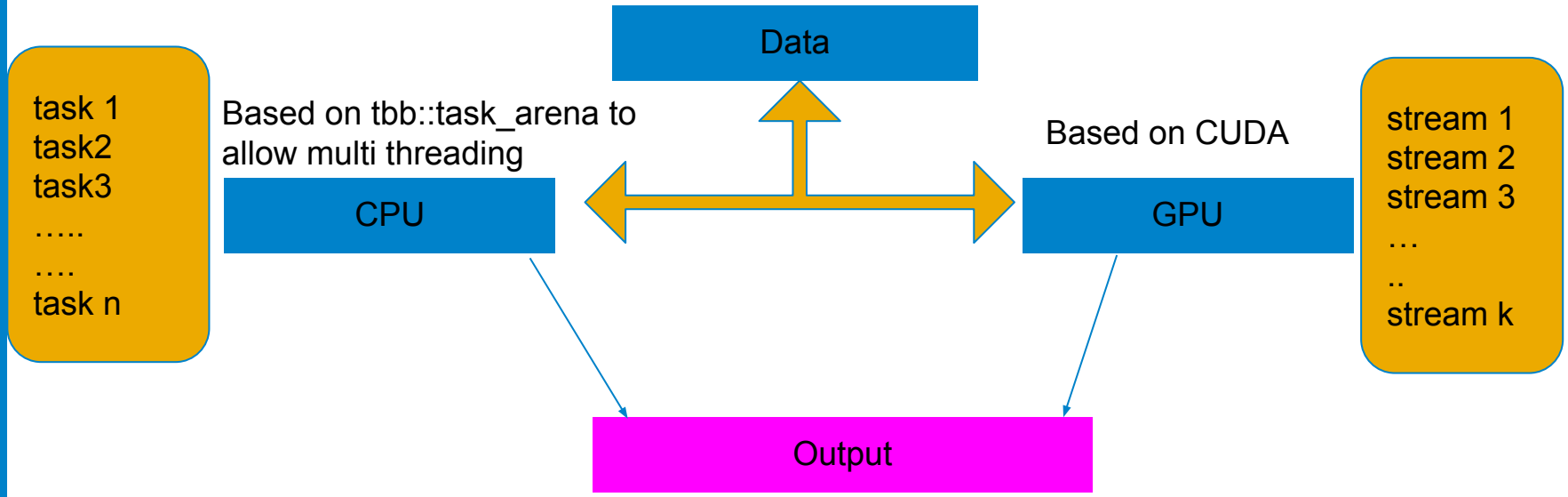


Execution Style



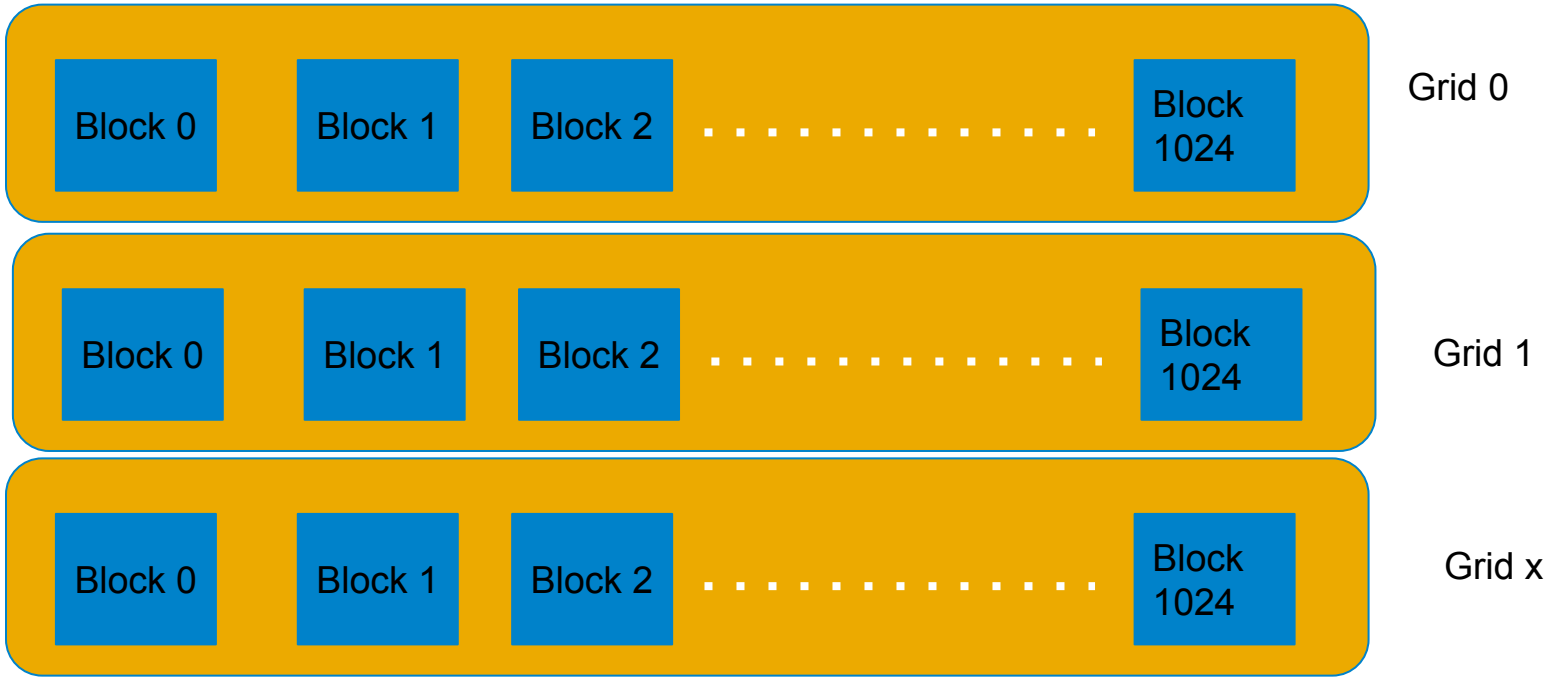
Work in Progress*

Output



Work Allocation in the Kernel

- Number of Grids and blocks per grid is calculated internally.
- Threading is set at Maximum threading per Block
  - Or number of iterations per block (if iteration number less than max threads)



Tests

- Performance verified by using “nvprof”

- 1 D array tests (Reading 2500 channels from ProtoDUNE Raw MC)
  - Host to Device Throughput ~5.3 GB/s
  - Device to Host Throughput ~ 3.88 GB/s

- 2 D array tests (work in progress)
  - Host to Device Throughput ~20.914 MB/s
  - Device to Host Throughput ~26.491 MB/s
  - Currently working on Improvements and proper implementation.

Future Works

- Further Work on 2D arrays needed
- Collective I/O Implementation
  - For HDF5 related I/O only
- Effect of precision on performance
- More customized data models that are closer to HEP data models currently used.
  - Build on the top of 1D and 2D arrays that the framework currently supports.
- Ideally would like to minimize (or remove at all) any CUDA API calls when needed.
- AOB

BACK UP

Additional Info:

- DUNE has workflow that uses GPU as server.
  - Could the data they are offloading can be restructured
  - Latency for communication between D to H and vice versa.
  - Do tests that are realistic (to get approximate time fraction spent executing tasks, copying data between host and device and so on).