GPU Friendly Data Model

Amit Bashyal & Peter Van Gemmeren (Argonne National Laboratory)
Kyle Knoepfel (Fermi National Accelerator Laboratory)
Beomki Yeo (Lawrence Berkeley National Laboratory)

Original data-model proposal (link)

***Develop data structures, optimized to make good use of CPU, GPU, and accelerator caches**, which:*

1. *Support SIMD/SIMT parallelization targeting CPU vector units, and GPUs*
2. *Allow efficient communication between CPU and accelerators*
3. *Can be efficiently serialized and deserialized concurrently for multiple events within a node and across multiple nodes.*
4. *Have a persistent representation that provides efficient access on HPCs, optimized for HEP I/O patterns (specifically Write-Once Read-Many)*
5. *Support partial reads from storage or memory of only the data needed by a given algorithm*

***We will engage the software communities of major HEP experiments, specifically ATLAS, CMS, and DUNE** to align our research with their objectives and to receive early and continuous feedback on our project assumptions and results….*

Status of data-model work

- Work has only recently begun on this effort (within the context of CCE-IOS)

- Within IOS, the scope has narrowed to exploring ***GPU-friendly*** data models
    - Assumption is that any any GPU-friendly data model is likely to be HPC-friendly
    - IOS "kick-off" meeting between Kyle, Peter and Amit (link)

- Difficulties regarding the original proposal:
    - It is helpful but over-ambitious given available effort.
    - Relevant parties within HEP are heavily engaged elsewhere (effort constraints), are unaware of this work (communication problem), … or maybe not interested.
    - Given these realities, the timeline and deliverables should be rethought.

Attempts at revitalization

- Communication between the IOS and PPS groups will be necessary to achieve progress and to remain in sync with each other.

- Last week, Peter, Amit and Kyle (IOS) and Beomki (PPS) met to discuss thoughts (partly in preparation for the AHM)

- Clear upshot from the meeting
  - To gain traction, this effort cannot start from ground zero.
  - We must leverage what has already been done by ATLAS and CMS (and possibly DUNE)
  - The use cases we study should generalize to other experiments that do not have resources to explore data models.

Possible approach: Share experiences given X, Y, Z, …

- “For data that looks like ***X***, and algorithms that look like ***Y***, and a computing platform that looks like ***Z***, we found that a data model of ***U*** performed well on a GPU.”

- Impossible to formulate an exhaustive catalog.

- However, for the use cases we explore, we can:
  - Identify each of *X, Y, and Z*
  - Share the lessons learned in finding *U*
    - What changes were required to the data model?
    - What changes were required for the framework?
  - Allow the wider HEP community to infer based on our experiences

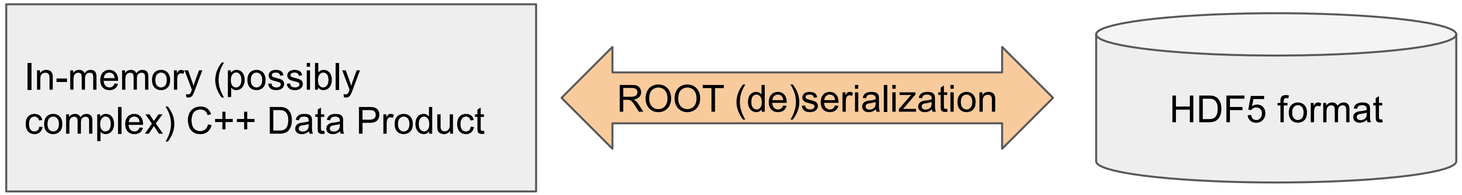
Possible approach: Share experiences given X, Y, Z, …

- **In the short term (e.g. the rest of this FY)**
    - We can survey the work that has been achieved and is still underway
        - Interaction with experiments who have already pursued GPU-friendly data models
        - Further interactions with PPS
        - Allows future experiments (like DUNE) to leverage past experience
    - We can also study how HPC-friendly storage layouts come into play (see next slides)

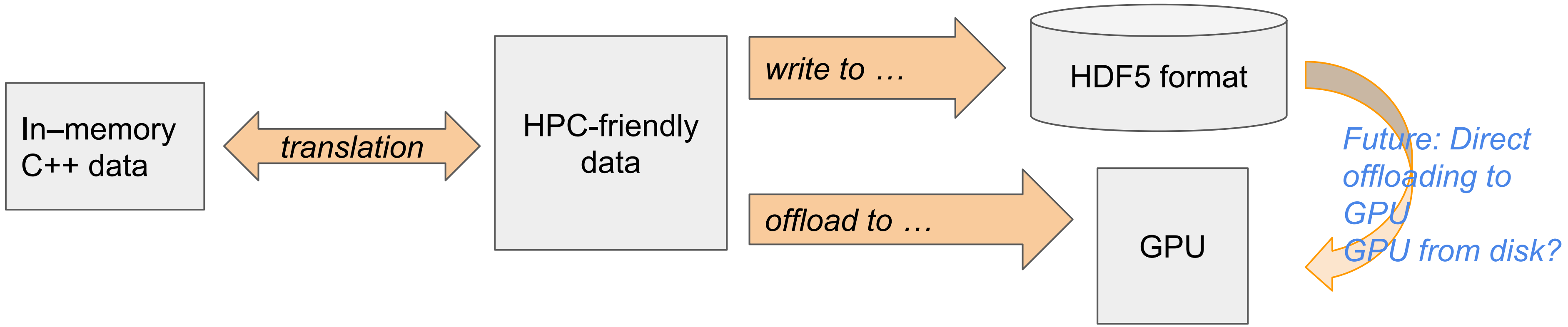
- **Longer-term** goals can be formulated once the shorter-term goals are realized.

Ultimately: The context is HPC

- The purpose of this effort is to explore data models that work well on HPC systems
- We've assumed that a GPU-friendly data model is also HPC-friendly
- But HPCs are not defined only in terms of their compute. IO/storage systems must be taken into account, especially as they differ from what HEP is used to.
- One avenue under exploration:



A Possible (Near Term) Use Case



Proto-DUNE is storing its raw data in HDF5 Format

- Could use real data in HDF5 format to explore these various translation layers

Backup

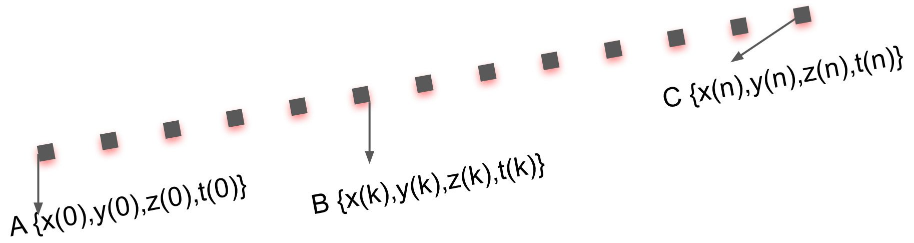
GPU Friendly Work-flow (Exploration – Amit (I))

- Simple case scenario (Dealing with the trajectories)
  - A Trajectory contains (x,y,z,t) or (px,py,pz,m)* or if indexing important (lamba_x,lamba_y, lamba_z, lamba_w,_index)
    - *(after some corrections)

- Transforming this information into GPU friendly format while exposing as less as possible GPU-API related nuisance (device to host communication, memory allocation to GPU etc)
  - Next step might be off-loading/

Current Status - Amit

Test with TVector type object (x,y,z,t)



```
Struct FourVector{
    double _px;
    double _py;
    double _pz;
    double _t;
    int _index;
};
 double4  _element (link for cuda vector_type (here))
(
  _element.x;
 _element.y;
 _element.z;
 _element.w;

)
```

Works for array of structures. Now working on structure of arrays (want to explore flexible arrays)

Both these data format works with the Template container CudaVector.

Note that above one has double type and int type and below one is all double type.