Link to Matti's talk:
https://indico.fnal.gov/event/55536/

CMS

Based on the Talk given by Matti Kortelainen (Jan 20, 2023)

Structure of the CMS Data

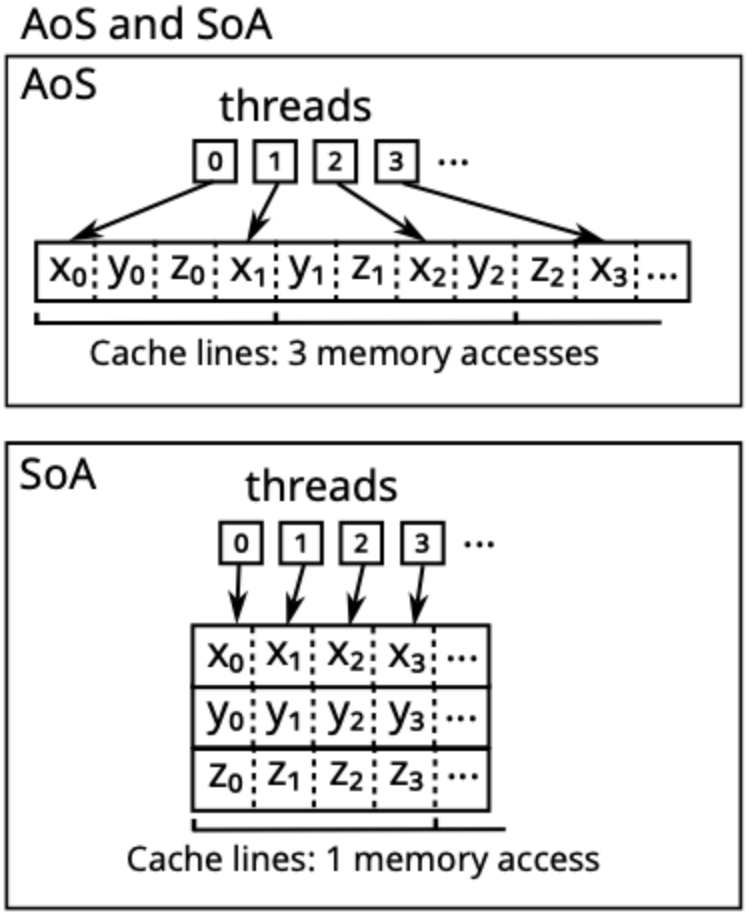
- Events are stored in anything that ROOT can serialize
- Collection of data products stored as std::vector.
- CMS Data model:
    - One data product can reference to other data products
    - Two collection of elements might be associated to each other

Making GPU Friendly Data Model

- std::vector<Foo> → Array of Structure (Traditional)
- (Array of) Structure of Arrays ((Ao)SoA)
- Structure of arrays → Contiguous memory

Early Work (Patatrack)

- Early exploration of memory management in SoA
    - Runtime-sized with memory for each array allocated
    - Runtime-sized with aggregated memory allocated (memory allocation for all arrays at once?)
    - Memory allocation during compile time with one call



SoA in CMS

- SOA with run time defined size with one (or minimal memory allocation)
- SoA can hold:
    - Array of “simple types”
    - Matrix (scalar of simple types)
- “Layout” to handle buffer
    - Buffer Size
    - Padding for cache line alignment
- “View” to interface with data
    - Access whole column
    - Access whole row
-

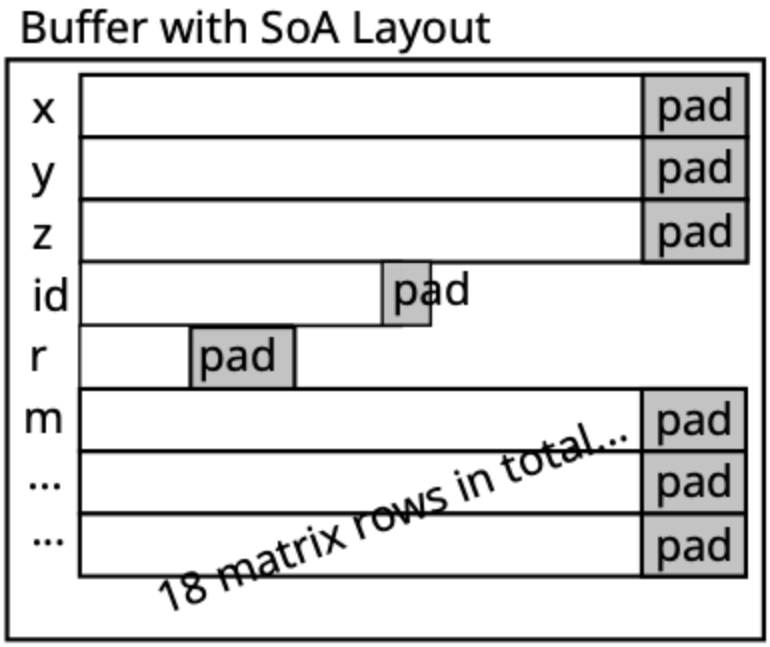


Figure: SoA Layouts in the Memory

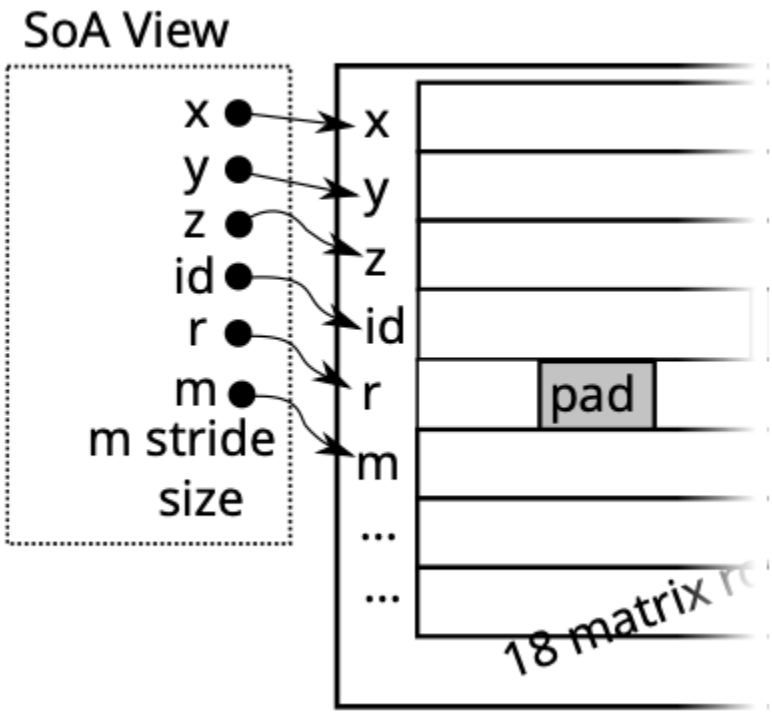


Figure: SoA View

Data Product:

- Cannot offload the data as View or Layout object
- Use of alpaka to handle the buffer memory with Layout given as a template argument
    - alpaka::Buf<...>

Persistency:

- ROOT related operations independent of Alpaka Buffer
    - Requires ROOT dictionary to read Layout Class