

# ALICE Analysis Framework for LHC Run III

Markus Zimmermann for the ALICE collaboration

07.07.2018



# ALICE Analysis

## LHC Run II

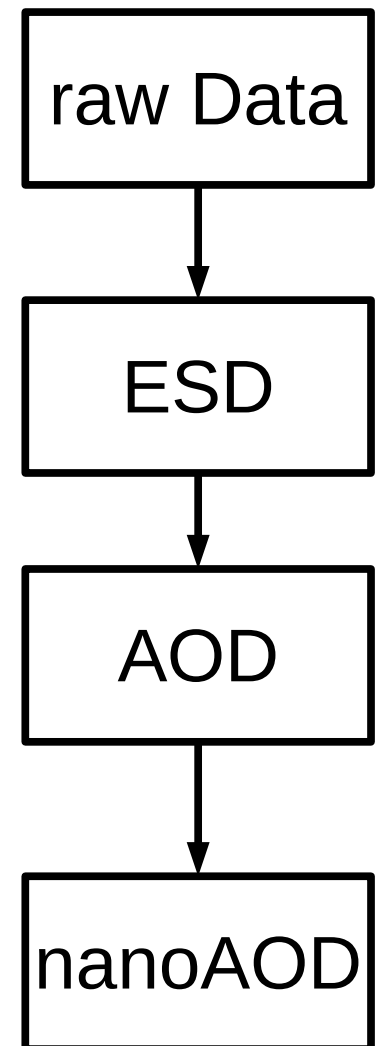
- 2015-2018
- Triggered readout
- $5.7 \cdot 10^8$  recorded Pb-Pb collisions in 1 dataset
  - $1.7 \cdot 10^5$  files
  - 121 TB
  - 2 CPU years to analyze
- Sequential Analysis Tasks

## LHC Run III

- 2021-2024
- Continuous readout
- $\approx 10^{10}$  recorded Pb-Pb collisions in 1 dataset
  - Similar amount?
  - ???
  - ???
- Parallel Analysis Devices

# Run II Data Format

- ALICE detector measures collision
  - Reconstruction in the High Level Trigger
  - Compression into raw data format
- Full reconstruction into ESD format
  - Event Summary Data
  - Full event and track information
- Smaller AOD format is extracted
  - Analysis Object Data
  - Sufficient Event and track information for most physics analysis
- Apply event and track cuts to create Nano AOD
  - Can only be used by 1 analysis



# Run II Analysis Flow

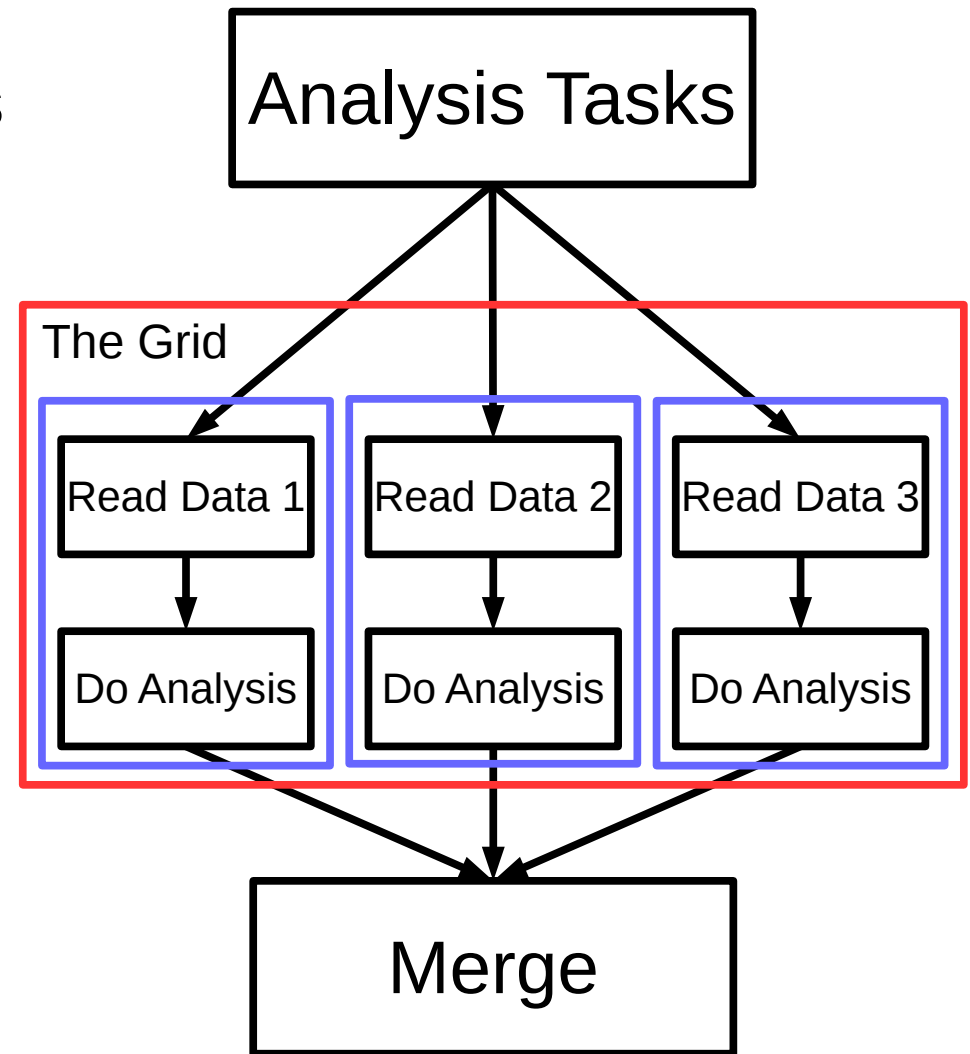
Create chain of analysis tasks

Send analysis jobs (blue) to  
the Grid (red) sites

Create event chain

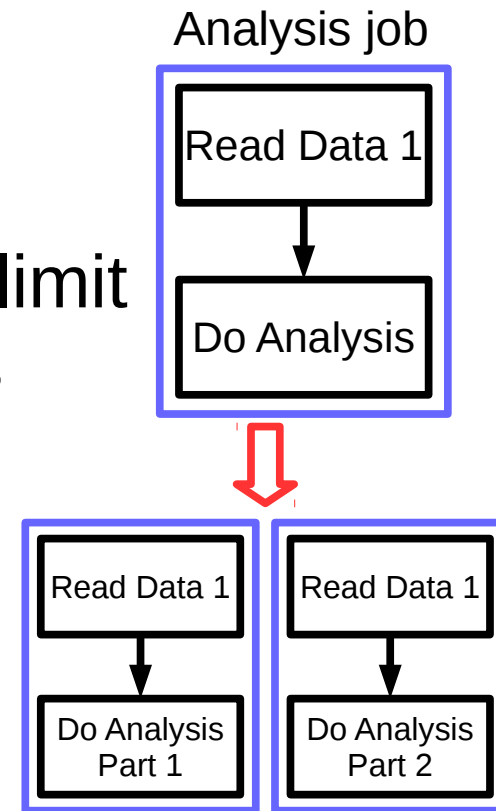
Loop over all events  
Do ROOT based analyses

Merge analysis results



# Bottlenecks

- ALICE analyses are I/O bound
  - High deserialization cost
  - If analysis is over analysis job memory limit
    - split analysis in multiple analysis jobs
    - read the same data multiple times
- Too many different data structures
  - Separate analysis procedures for ESD/AOD/nanoAOD datasets
- Complicated to add new variables to the data structure (growing)



# The future O<sup>2</sup> Framework

- Unify online and offline analyses
  - Use the same reconstruction and data format
  - Flat data format to reduce deserialization cost
- Continuous data readout from the detector
  - Tracks are organized in so-called Timeframes
  - Event loop only available on a high level workflow
- Use slow compression and fast decompression algorithms
- Use of O<sup>2</sup> Analysis Facilities (AF)
  - Dedicated Grid sites with a large amount of memory
  - Keep datasets in memory for all user analyses



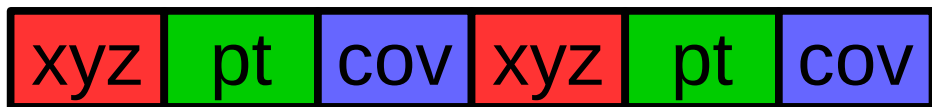


# Data Format Requirements

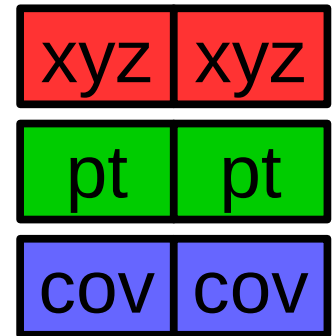
- One data format containing
  - Reconstruction and analysis variables
- Fast deserialization
  - Flat data format
  - No pointers
- Support of
  - Pruning (not loading some some attributes for all tracks)
  - Skimming (not loading any attributes for some tracks)
  - Growing (adding extra track attributes)

# Data Structure

- Array of Structs (AoS)
- Advantages
  - Skimming
  - Efficiently reading full track
  - Same as run II data structure
- Disadvantages
  - Pruning
  - Growing

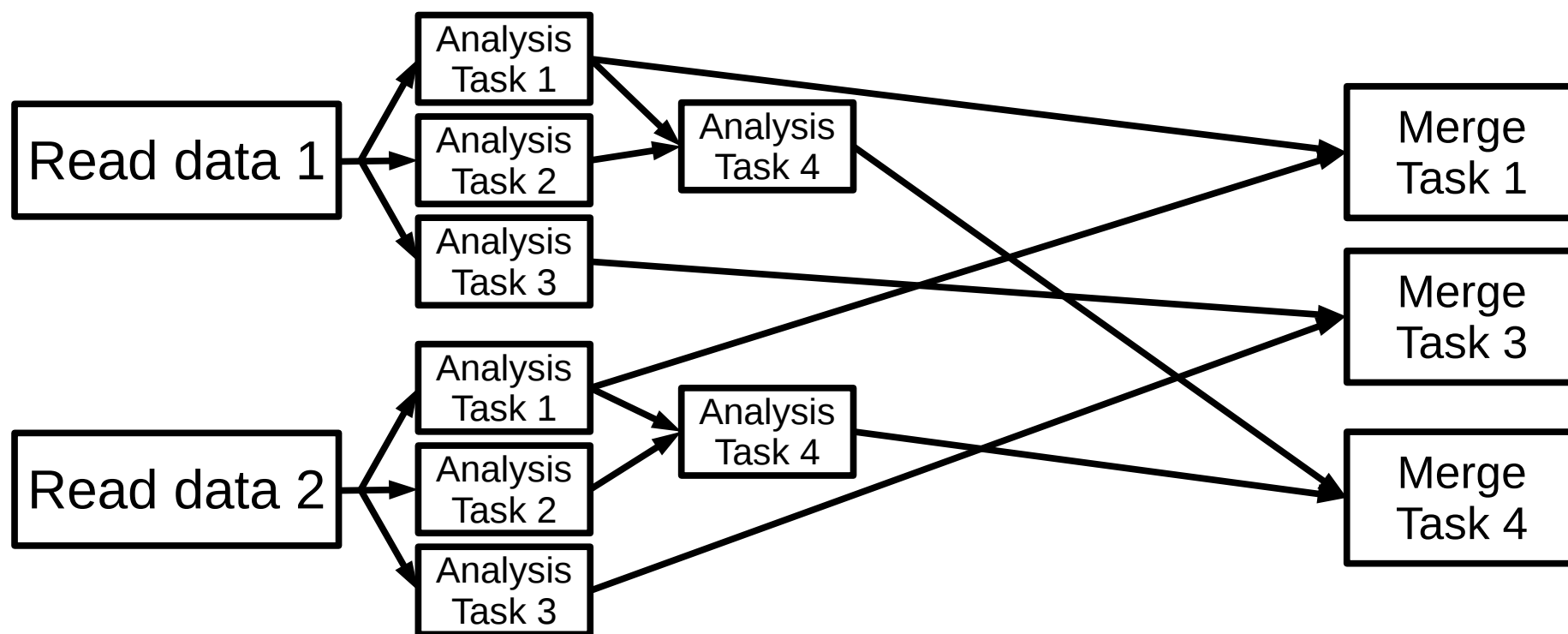


- Struct of Arrays (SoA)
- Advantages
  - Pruning
  - Growing
  - Vectorization
  - Efficiently reading part of the track variables
- Disadvantages
  - Many scattered operations to read all track variables
  - Skimming

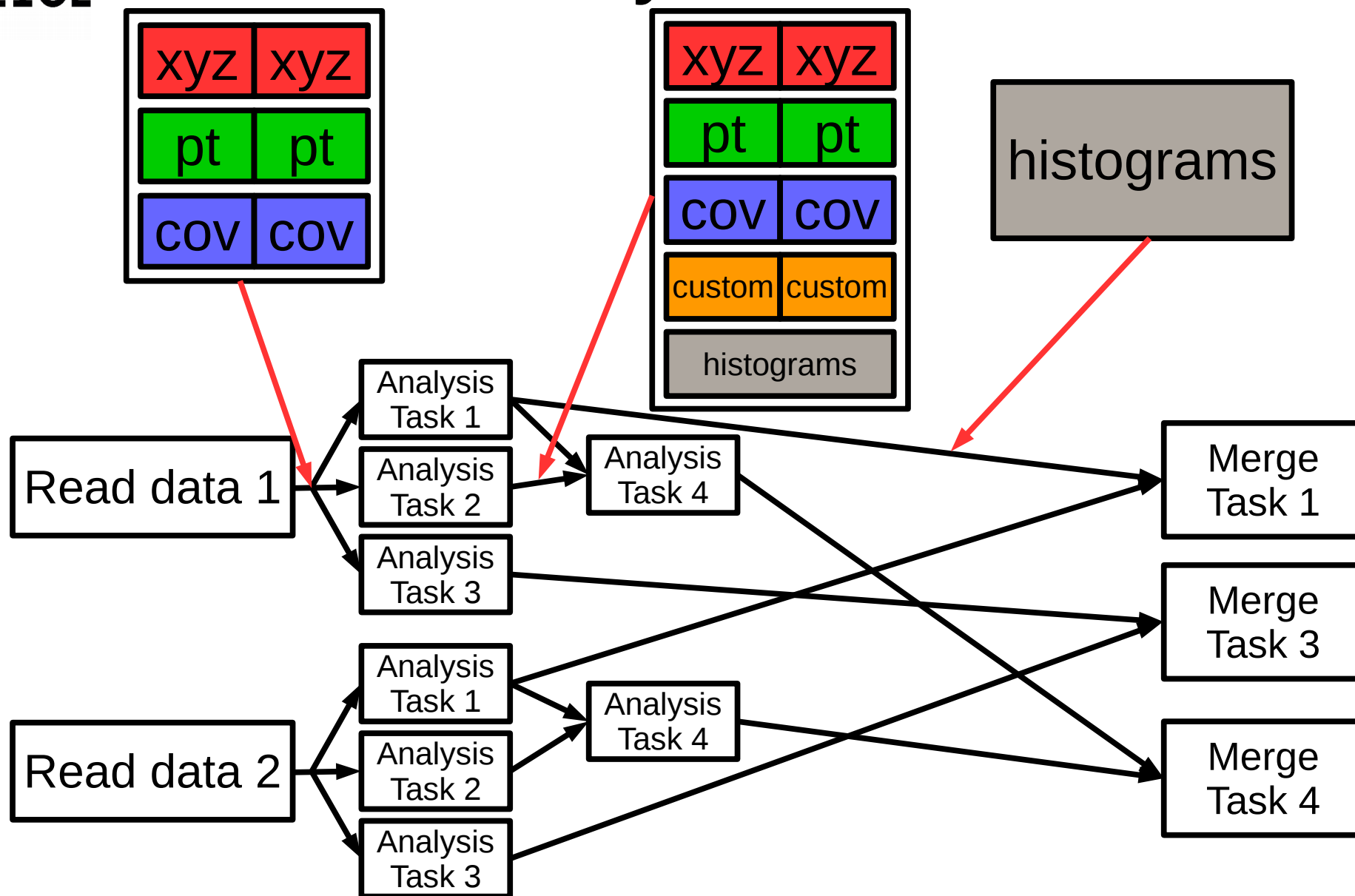


# Analysis Flow

- Data is read and deserialized once and sent to the analysis tasks
- Use in-memory, zero copy messaging system (fairmq) to communicate between analysis processes
- Allow ROOT based analyses
- Output can be sent to other analyses tasks or be merged as final output



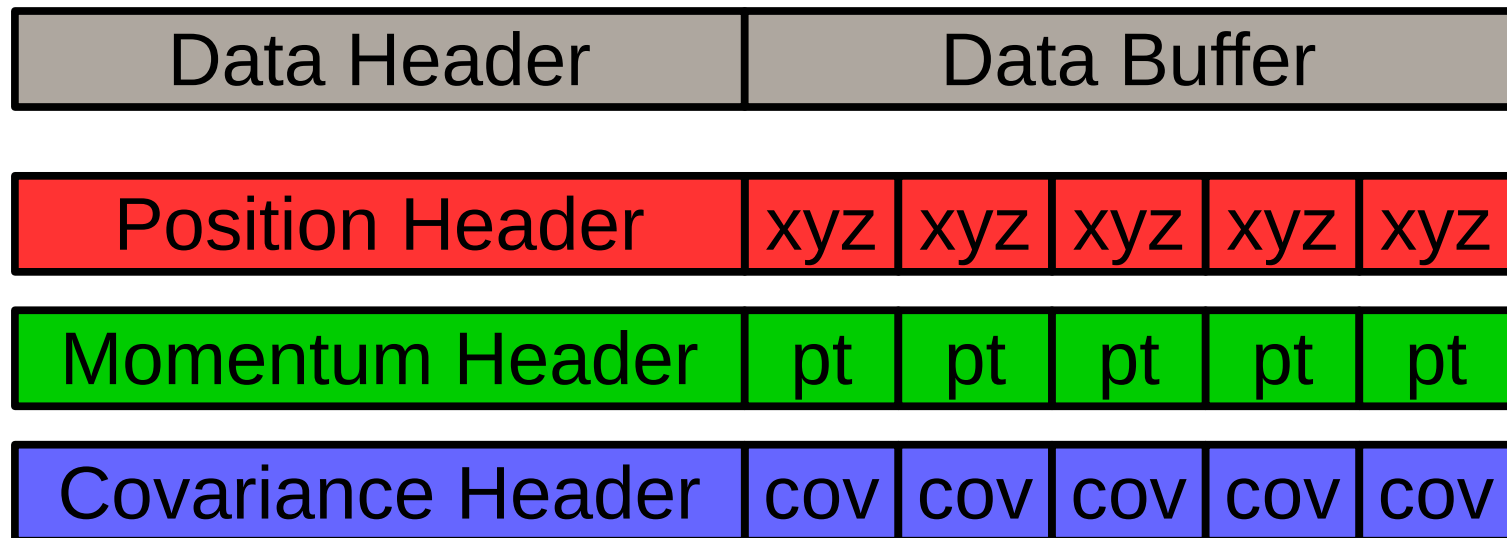
# Analysis Flow





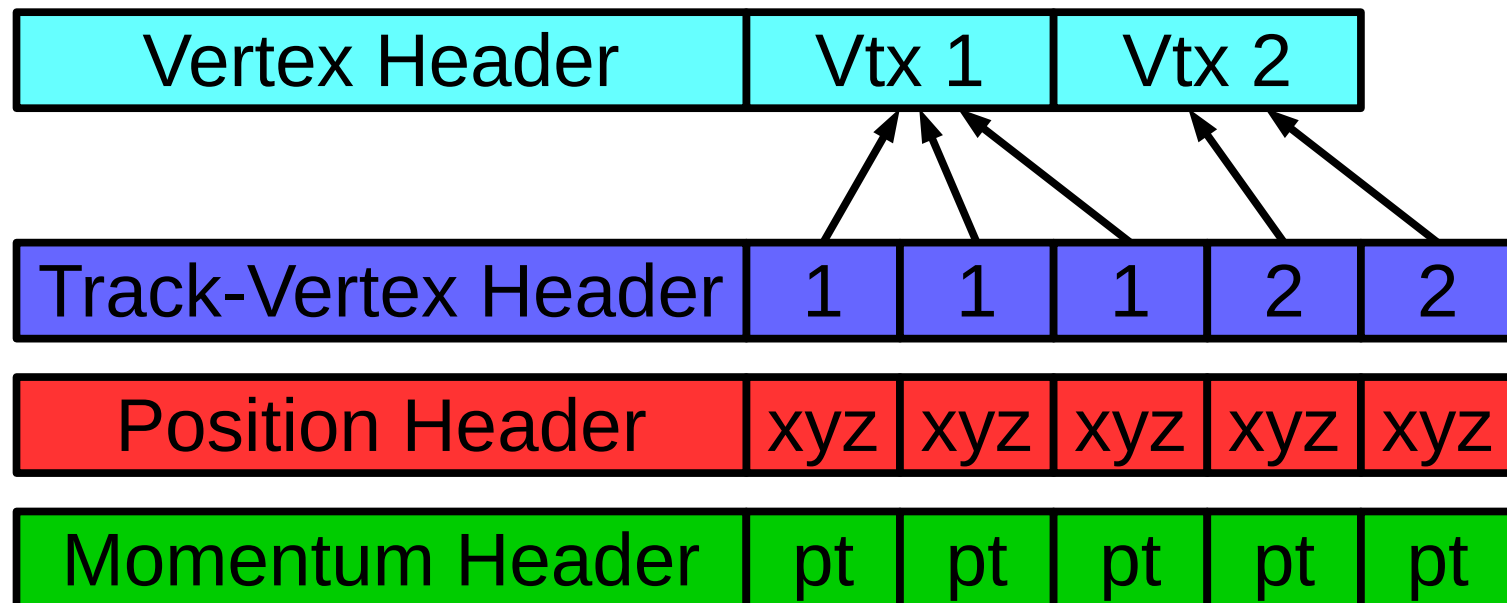
# Data Message

- Use over the network message queuing (=buffers)
- Headers carry metadata used to uniquely identify the message payload
  - Data buffer type
  - Data buffer size
- Data element have to be flat (self-contained)
  - Any data type, arrays, structs, ...
  - No pointers



# References in Data Message

- References between data buffers via the same index
- Separate data buffers can have different entry number
  - Use separate buffer for the reference
  - Track-Vertex contains the index of the vertex related to the track



# Summary

- ALICE will operate in run III in a continuous readout mode
  - Order of  $10^2$  more data
- Change underlying data structure
  - Use Struct of Arrays (SoA) instead of Arrays of Structs (AoS)
  - Store tracks in timeframes instead of events
- Device based analysis framework
  - Read data once and distribute it to the analyses jobs
  - Use self-contained messages without pointers
  - Use specialized analysis facilities (AF) with a lot of memory
- Changes break backwards compatibility of ALICE analyses
  - Backwards compatibility is kept wherever possible
  - Easy porting of old code is foreseen



# BACKUP



# Statistics

- ALICE is running on average 115k analysis jobs on the Grid
  - 82k MC productions
  - 18k organized analysis
  - 10k raw data processing
  - 5k individual user analysis
- One year ago ALICE did run on average 83k
  - 55k MC productions
  - 16k organized analysis
  - 9k raw data processing
  - 4k individual user analysis

# Data Format on Disk

- Close to the data Message format
  - Keep deserialization cost small
- Optimize compression factor
  - Efficient use of available disk space
- Use available Open Source Solutions?
  - Several prototypes are currently tested
  - Use synergies from external developers