WAYNE STATE
UNIVERSITY

ALICE

Using derived data in analysis

O2 Analysis tutorial 3.0, 7th November 2023

A few things already said

How to produce derived data

Creating your own tables

or: the moment when things get really interesting

myTable.h

```
#include "Framework/ASoA.h"
#include "Framework/AnalysisDataModel.h"
namespace o2::aod {
namespace myTable {
DECLARE_SOA_COLUMN(MyValue, myValue, float, "myValue");
} //end myTable namespace
DECLARE_SOA_TABLE(MyTable, "AOD", "MYTABLE", myTable::MyValue);
}
```

```
struct ATask {
    Produces<aod::MyTable> thisTableHere;
    (…)
    process(o2::aod::Collision const& collision, soa::Join<Tracks, TracksExtras> const& myTracks) {
        registry.fill(HIST("hCandidateCounter"), 0.5);
        for (auto track : myTracks) {
            registry.fill(HIST("phi"), track.phi()); //property in Tracks
            registry.fill(HIST("length"), track.length()); //property in TrackExtras
            thisTableHere(  track.phi() + M_PI ); //this fills our new table! (very simple example…)
        }
    };
};
```

This operation is flexible! We can then use the extra table for filtering (ultra fast), manipulating, etc and be very modular!
In this case, this new table can be joined with tracks (same size)

Why you would need derived data
A use case

- **Say you need to run over tracks**
  - In a two nested loops
  - Need particle identification
  - Need some filtering
  - Constraints on execution time
- **You define/use two tasks**
  - First one classifies the tracks – **the classifier**
  - Second one processes the classified tracks – **the consumer**
- **Tracks classification in a new table**
  - Just one single column
  - Produced by the classifier
  - Joined to the `Tracks` table in the consumer `process...` subscription

A use case

```
#include "Framework/ASoA.h"
#include "Framework/AnalysisDataModel.h"
namespace o2::aod {
namespace myTable {
DECLARE_SOA_COLUMN(TrackCode, trackCode, int, "trackCode");
} //end myTable namespace
DECLARE_SOA_TABLE(MyTable, "AOD", "MYTABLE", myTable::TrackCode);
} //end o2::aod namespace
```

```
struct producer {
  Produces<aod::MyTable> thisTableHere;
  ...
  process(o2::aod::Collision const& collision, soa::Join<Tracks, TracksExtras> const& myTracks) {
    for (auto track : myTracks) {
      ...
      thisTableHere(thetrackcode); //this fills our new table! (very simple example...)
    }
  }
};
```

```
struct consumer {
  ...
  process(o2::aod::Collision const& collision, soa::Join<Tracks, TracksExtras, MyTable> const& myTracks) {
    for (auto track1 : myTracks) {
      for (auto track2 : myTracks) {
        ...
      }
    }
  }
};
```

Are these derived data?
The described use case

- **Actually, yes**
  - You produce a table from the processing of other tables
- **You benefit from the SOA approach**
  - Faster access
  - Bulk processing
  - Zero copy
- **But we will not refer to them as derived data**
  - You process them on the fly
  - You don't store them
  - You shouldn't / cannot store them
  - You should use them as much as you can!!!

Storing and using derived data

Derived table handling

- **Writing tables to disk**
- Any table that is accessible by its type can be written to disk at the end of processing by using:
  - `--aod-writer-keep` command line option (See docs for more options)
- This is mainly useful for storing skims and ML training data
- Tables are stored as ROOT trees

Using tables in processing

- Any table that is accessible by its type and has been created by means of `Produces<>` , `Spawns<>` or `Builds<>` can be subscribed by other tasks in the workflow
- It behaves exactly as the tables that were read from AOD file and can be subjected to the same operations
- A typical usage is joining the data tables with those produced by helper tasks (e.g. track DCA, PID, track and event selection)

Saving and retrieving derived data

- **Saving tables to a file**
  - `OutputDirector` configuration file with `--aod-writer-json`
  - https://aliceo2group.github.io/analysis-framework/docs/basics-usage/SavingTablesToFile.html
- **Reading tables from files**
  - `InputDirector` configuration file with `--aod-reader-json`
  - https://aliceo2group.github.io/analysis-framework/docs/basics-usage/ReadingTablesFromFile.html

But that is for your local tests

How to do it

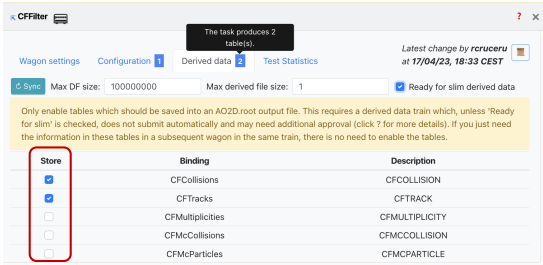
```
namespace cfskim
{
DECLARE_SOA_COLUMN(CFCollisionFlags, selflags, uint64_t);
DECLARE_SOA_INDEX_COLUMN(CFCollision, cfcollision);
DECLARE_SOA_COLUMN(CFTrackFlags, trackflags, uint64_t);
DECLARE_SOA_COLUMN(CFPidFlags, pidflags, uint64_t);
DECLARE_SOA_COLUMN(Pt, pt, float);
DECLARE_SOA_COLUMN(Eta, eta, float);
DECLARE_SOA_COLUMN(Phi, phi, float);
DECLARE_SOA_DYNAMIC_COLUMN(Sign, sign,
                           [](uint64_t mask) -> int8_t
                           { return ((mask & 0x1L) == 0x1L) ? 1 :
                                ((mask & 0x2L) == 0x2L) ? -1
} // namespace cfskim
DECLARE_SOA_TABLE(CFCollisions, "AOD", "CFCOLLISION",
                  o2::soa::Index<>,
                  collision::PosZ,
                  bc::RunNumber,
                  timestamp::Timestamp,
                  cfskim::CFCollisionFlags);
DECLARE_SOA_TABLE(CFTracks, "AOD", "CFTRACK",
                  o2::soa::Index<>,
                  cfskim::CFCollisionId,
                  cfskim::CFTrackFlags,
                  cfskim::Pt,
                  cfskim::Eta,
                  cfskim::Phi,
                  cfskim::Sign<cfskim::CFTrackFlags>);
DECLARE_SOA_TABLE(CFTrackPIDs, "AOD", "CFTRACKPID",
                  cfskim::CFPidFlags);
```

```
{
  "OutputDirector": {
    "debugmode": false,
    "resfile": "AnalysisResults_trees",
    "resfilemode": "RECREATE",
    "ntfmerge": 1,
    "OutputDescriptors": [
      {
        "table": "AOD/CFCOLLISION/0",
        "treename": "O2cfcollision",
        "columns": [
          "fPosZ",
          "fRunNumber",
          "fTimestamp",
          "fCFCollisionFlags",
          "fCFCollisionCentMult"
        ]
      },
      {
        "table": "AOD/CFTRACK/0",
        "treename": "O2cftrack",
        "columns": [
          "fIndexCFCollisions",
          "fCFTrackFlags",
          "fPt",
          "fEta",
          "fPhi"
        ]
      },
      {
        "table": "AOD/CFTRACKPID/0",
        "treename": "O2cftrackpid",
        "columns": [
          "fCFPidFlags"
        ]
      }
```

On hyperloop it is easier

Derived data settings

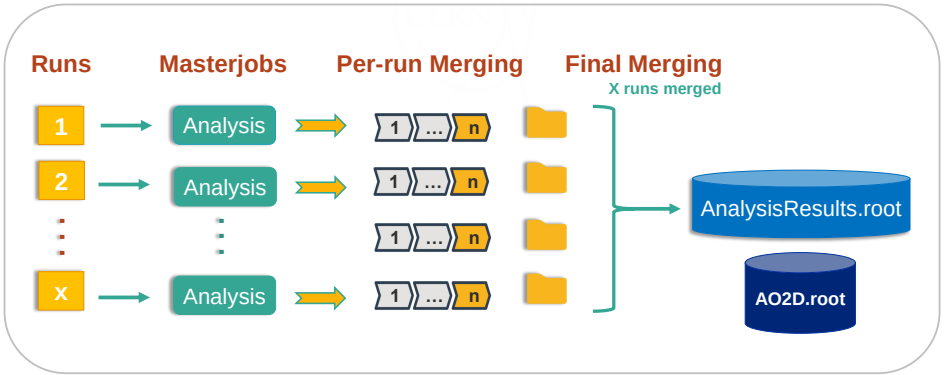
- Displays the tables which are produced by the task
- Here you can enable tables which should be saved into an **AO2D.root output file**
- ***This requires a derived data train which, unless 'Ready for slim' is checked, does not submit automatically and may need additional approval***
- ***If you just need the information in these tables in a subsequent wagon in the same train, there is no need to enable the tables***
- ***For derived data of small output size, you can enable the slim derived data option***



- In order to ***update*** the derived data configuration with the latest O2Physics version of the workflow, click on the **↻sync button**
- By synchronizing the derived data, the tables which no longer belong to the workflow will be removed, and the values of the tables will be updated

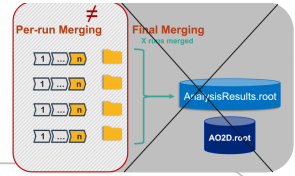
But a more varied zoo

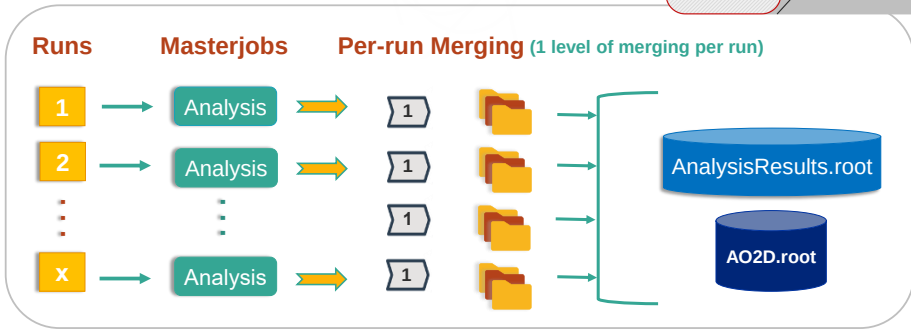
Slim Derived Data Train



But a more varied zoo

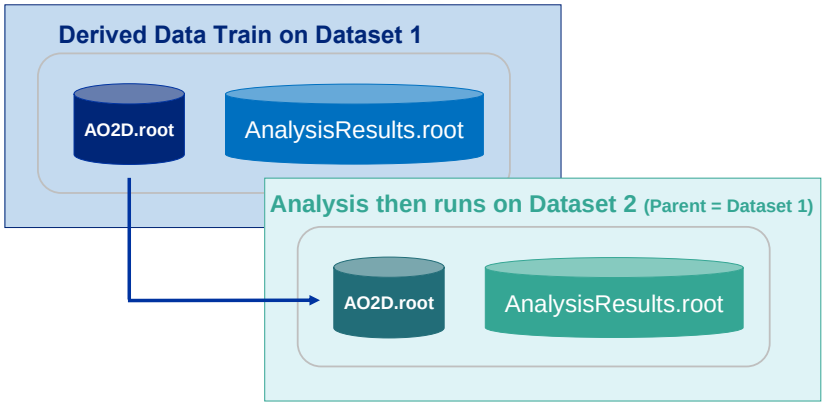
Derived Data Train





But a more varied zoo

Linked Derived Data Train



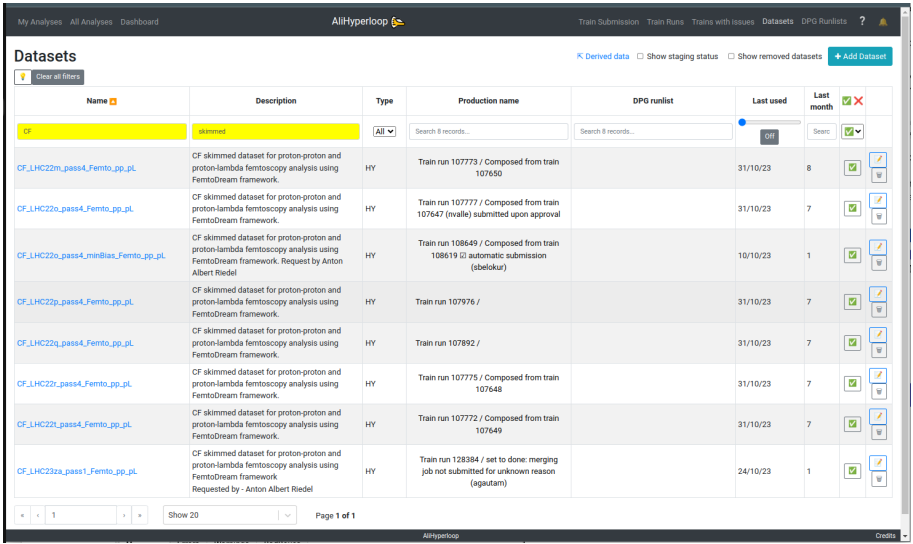
Ask the train operators

Train runs

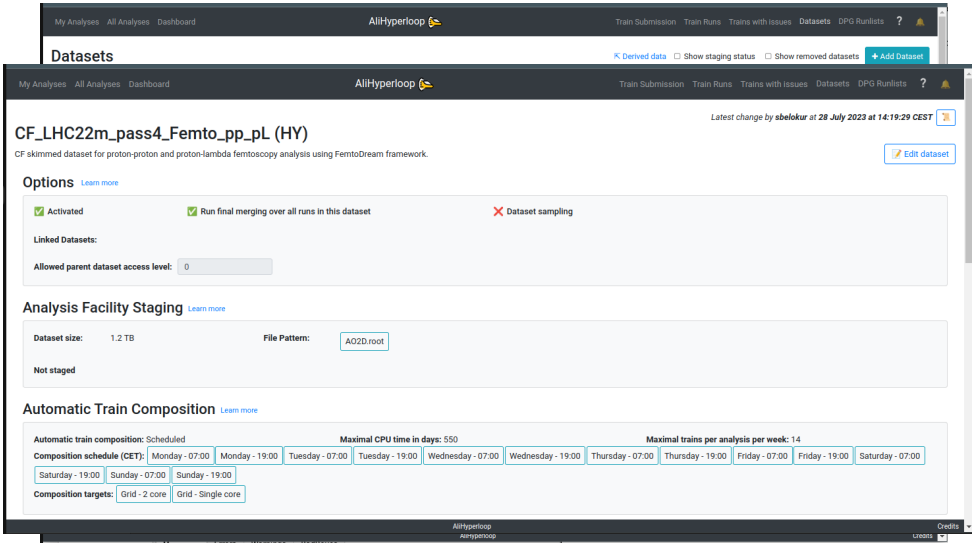
The train type is decided by operators at composition in the Train Submission page

1. **Analysis Train** - is a standard analysis train and no derived data will be produced
2. **Slim Derived Data** - reserved for derived data of **small output size**
   - Similarly to the standard derived data case, this train **will produce derived data** to be used for further analysis
   - The **results will be merged across runs** and are **not available to use in future train runs**
   - The data will be **automatically deleted** after a pre-set period of time
3. **Standard Derived data** - **will produce derived data** to be used for further analysis
   - The **results will not be merged across runs** and can be used **as input for future train runs**
4. **Linked Derived data** - this option is for **derived data which needs to access its parent file when it is processed**
   - The derived data file produced will remember its parent files, inheriting also their storage location
   - The **results will not be merged across runs** and can be **used as input for future train runs**
   - Datasets composed from this train need to have parent access level activated

Productified derived data



Productified derived data



Now we are talking!

In Run 3 you cannot walk alone

But that's why we are a collaboration

Huge amount of collected data

New datasets available from 2022

more information: https://twiki.cern.ch/twiki/bin/view/ALICE/AliDPGRun3DataSets

Period	System	reco†	Size (full)
22m	pp 13.6 TeV	pass3	341 TB
22o	pp 13.6 TeV	pass3	2.1 PB
22p	pp 13.6 TeV	pass3	95 TB
22q	pp 13.6 TeV	pass3	9.5 TB
22r	pp 13.6 TeV	pass3	299 TB
22t	pp 13.6 TeV	pass3	67.7 TB
22s	Pb-Pb 5.36 TeV	pass4	680 GB
Full pp datasets			**3.0 PB**
Full Pb-Pb dataset			**680 GB**

† please note: rapidly changing conditions!

DPG ALICE • Data Preparation Group

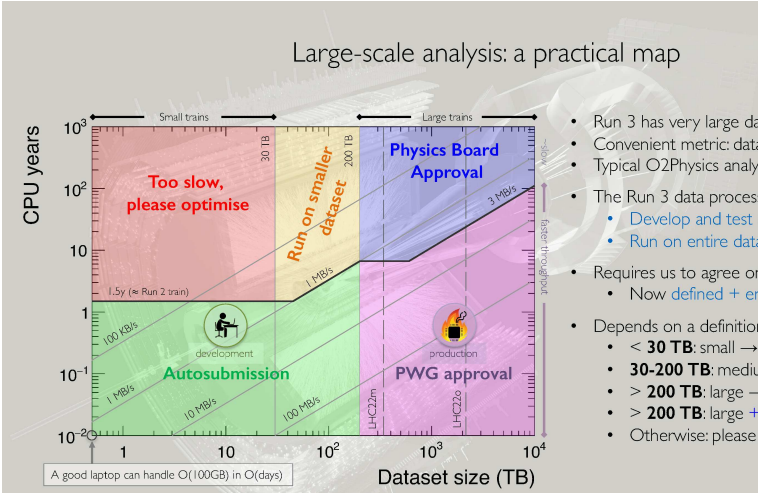
- Large effort in reconstruction software and calibrations ongoing
  - Situation still in flux!
- Software triggering processing: select interesting data!
  - The Run 3 modus operandi
- Analysing this data is a challenge!
  - Hyperloop: next-generation train system
  - Organized analysis is required
  - Organized how?

→ some rules are required

UNICAMP

ALICE

Limited processing capacity



- Run 3 has very large datasets!
- Convenient metric: data processed per unit time
- Typical O2Physics analysis task speed: 1-10 MB/s
- The Run 3 data processing logic:
  - Develop and test on small datasets (freely)
  - Run on entire dataset when ready (organized)
- Requires us to agree on **computing rules**
  - Now defined + endorsed by Physics Board
- Depends on a definition of dataset size scale
  - < **30 TB**: small → **autosubmission**
  - **30-200 TB**: medium → **autosubmission**
  - > **200 TB**: large → **PWG approval**
  - > **200 TB**: large + slow → **PB approval**
  - Otherwise: please work in **dev mode**

Relaying on derived/skimmed data

Projections for Train Running

- Run 2: ~1200 LEGO trains/month (mix of datasets) with on average 40k jobs
- Run 3 assumptions (indicative examples)
  - AFs: 5k cores | Grid: 50k cores for analysis
  - Throughput 10 MB/s (including some expected improvements)
  - AOD of 1 month PbPb: 5 PB
- Run 3 projections
  - Full AOD: 270 trains/month → exceeds what storage can provide by factor 2
  - Needs derived data (here some illustrative examples)
    - Correlations: CPU ⇓2 | Storage ⇓90 → 540 trains/month
    - Nuclei: CPU ⇓9 | Storage ⇓1000 → 3200 trains/month
    - UPC: Storage ⇓90 000 → off the Grid

Despite significant increase in taken data, comparable number of trains can be run

Derived/skimmed data

- **Statistics demanding analyses**
  - Will only be able to be run on derived data
  - Derived data stored and productified as actual data
  - Amount of stored derived data limited at PWG level
- **Only golden periods will be available for analyses**
  - Derived data concept able to be used
  - Derived data will not be stored (size on pair of actual data)

Derived/skimmed data

- **First rule: don't create stored derived data**
- **Second rule: don't create stored derived data**
- **Present your needs in your PAG**
- **Be ready to discuss them in your PWG**
- **Familiarize with the derived data data model**
- **Analysis tasks on derived data**
  - Are developed exactly in the same way
  - The appropriate tables have to be selected
- **In your hands-on session you will touch derived data**

Derived/skimmed data

- **First rule: don't create stored derived data**
- **Second rule: don't create stored derived data**
- **Present your needs in your PAG**
- **Be ready to discuss them in your PWG**
- **Familiarize with the derived data data model**
- **Analysis tasks on derived data**
  - Are developed exactly in the same way
  - The appropriate tables have to be selected
- **In your hands-on session you will touch derived data**

– THANK YOU –