WAYNE STATE
UNIVERSITY

Using derived data in analysis

O2 Analysis tutorial 4.0, 15th October 2024

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**
  - You want to extract two-particle correlations
    - You need two nested loops
  - Need particle identification
  - Need some filtering
  - Constraints on execution time

Why you would need derived data
A use case - implementation

- **You define/use two tasks**
  - First one classifies the tracks – **the classifier**
  - Second one processes the classified tracks – **the consumer**
    - and extracts the two-particle correlations
- **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 – Table declaration

```
#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
```

A use case – The producer

```
DECLARE_SOA_COLUMN(TrackCode, trackCode, int, "trackCode");
DECLARE_SOA_TABLE(MyTable, "AOD", "MYTABLE", myTable::TrackCode);
```

```
struct producer {
  Produces<aod::MyTable> thisTableHere;
  ...
  process(soa::Join<Tracks, TracksExtras> const& myTracks) {
    for (auto track : myTracks) {
      int thetrackcode = -1;
      ...
      thisTableHere(thetrackcode); //this fills our new table!
    }
  }
};
```

A use case – The consumer

```
DECLARE_SOA_COLUMN(TrackCode, trackCode, int, "trackCode");
DECLARE_SOA_TABLE(MyTable, "AOD", "MYTABLE", myTable::TrackCode);
```

```
struct consumer {
  ...
  process(o2::aod::Collision const& collision,
          soa::Filtered<soa::Join<Tracks, TracksExtras, MyTable>> const& myTracks) {
    ...
    for (auto track1 : myTracks) {
      for (auto track2 : myTracks) {
        ...
        myHist[track1.trackCode][track2.trackCode]->Fill(getDeltaPhi(track1,track2));
      }
    }
  }
};
```

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

Are these derived data?
The described use case

- **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

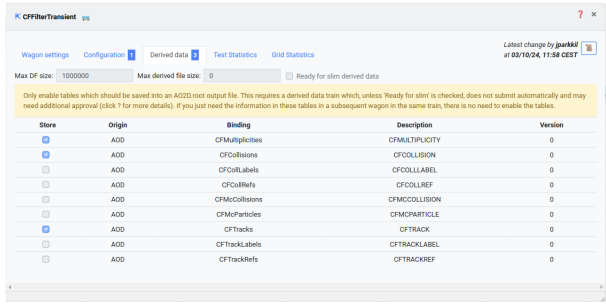
```
namespace cfskim
{
DECLARE_SOA_COLUMN(CFCollisionFlags, selflags, uint64_t);
DECLARE_SOA_INDEX_COLUMN(CFCollision, cfcollision);
DECLARE_SOA_COLUMN(CFTrackFlags, trackflags, uint64_t);
DECLARE_SOA_COLUMN(CFPidFlags, pidflags, uint64_t);
DECLARE_SOA_COLUMN(Pt, pt, float);
DECLARE_SOA_COLUMN(Eta, eta, 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

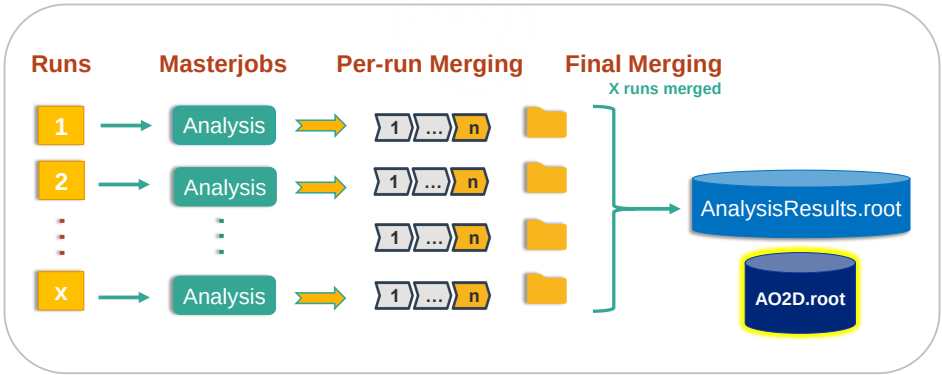
- Displays the tables which are produced by the task
- Here you can enable tables which should be saved into an **AO2D.root output file**
- ***This requires a derived data train which, unless 'Ready for slim' is checked, does not submit automatically and may need additional approval***
- ***If you just 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

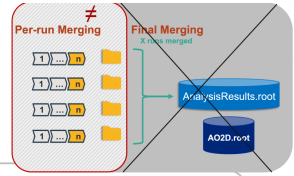


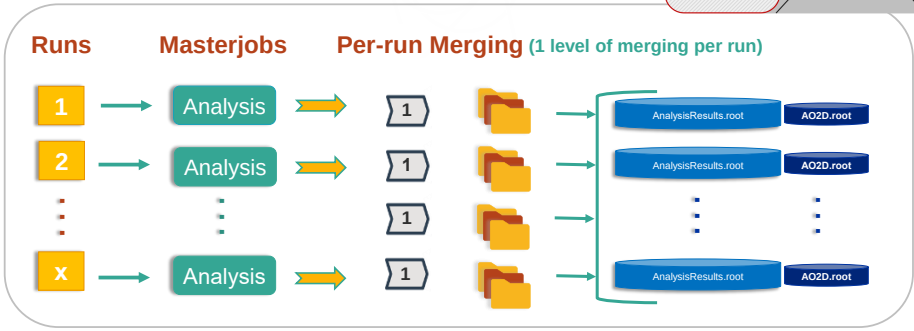
Slim derived trains provide an AO2D.root to be used locally. **Only possible when output < 4GB.**

But a more varied zoo

Derived Data Train

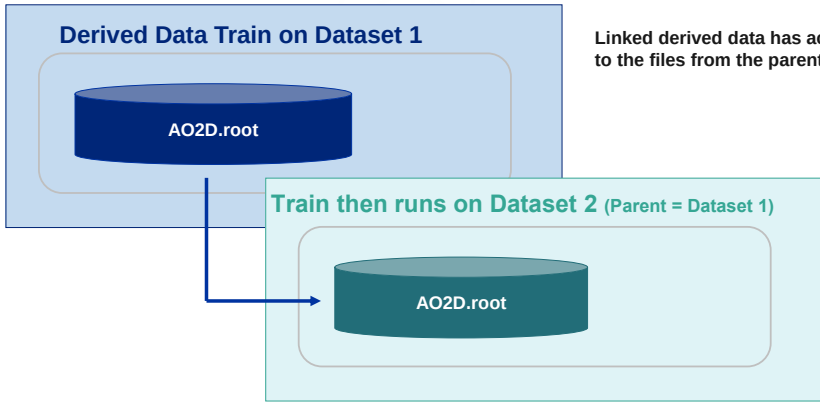
To be used as input in future train runs.





But a more varied zoo

Linked Derived Data Train



Linked derived data has access to the files from the parent.

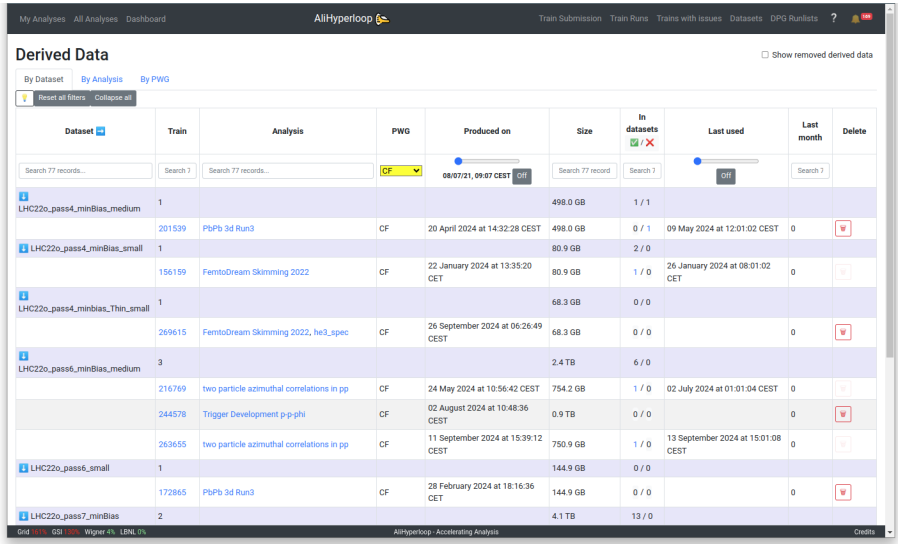
Ask the train operators

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



Now we are talking!

In Run 3 you cannot walk alone

But that's why we are a collaboration

Huge amount of collected data

Disk usage — Trigger report. July AW

- Total disk buffer size: 150 PB
- Currently on disk (05.07.24): 112 PB
  - 47 PB PbPb 2023 raw CTF + 65 PB pp 2024 raw CTFs
- Accumulating ~9PB of pp data per week
- Expected size of HI 2024: 68 PB = 14 PB of pp-ref(4.5/pb) + 54 PB of Pb-Pb (1.9/nb)

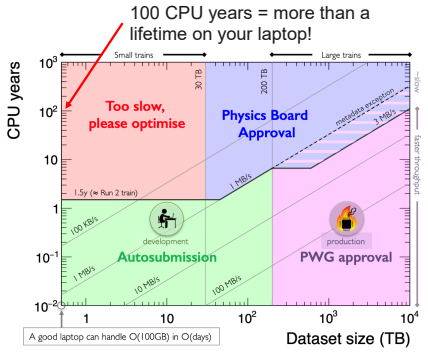
8

Limited processing capacity

Fair usage policy

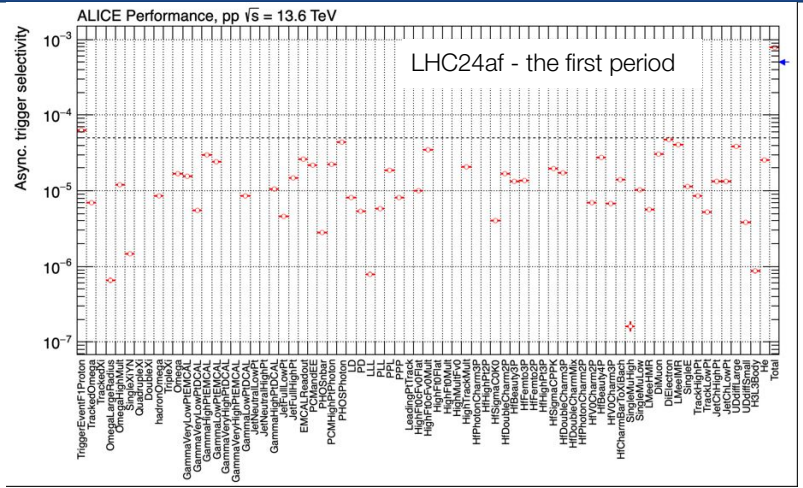
- Operators follow guidelines prepared by analysis coordination and approved by physics board (current policy documented here)
  - **Operators cannot grant exceptions, even if justified**
- Aim of guidelines
  - allow efficient analysis by everyone
  - share resources fairly
  - avoid excessive use; identify room for optimization

Dataset size	CPU limit	Trains / week	Automatic schedule
Small datasets			
< 30 TB	1.5 CPU year (550)	14	twice per day
Medium datasets			
< 100 TB	3 CPU years (1095)	6	once a day
< 200 TB	6 CPU years (2190)		twice per week
Large datasets			
< 300 TB	6 CPU years (2190)	2	none (PWG / PB approval)
< 400 TB	6 CPU years (2190)		



Relaying on derived/skimmed data

Trigger menu for 2024 **Trigger report. July AW**



11

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 –