Analysis in Run3

G.Eulisse, G.M. Innocenti, P. Hristov

CERN

15.05.2019

Introduction

Challenge and Solutions (Andreas @ ALICE Week, 03.2019)

- Challenge: Cope with 100x larger number of collisions to analyze
- Solutions
  - Only AOD for analysis: reduce the size, avoid redundant information
  - Reduce time spend in I/O
    - fast storage access ➜ dedicated Analysis Facilities
    - deserialisation overhead from complex nested data structures ➜ flat tables
  - Exploit parallelism
    - multiple data processing devices and multiple analysis tasks
    - task parallelism (multiprocessing and shared memory with DPL)
    - columnar data format ➜ vectorisation (RDataFrame)
    - declarative analysis providing automatic automatisation in the background
  - Exploit common data skimming and filtering

Analysis data format: requirements

Dario @ CHEP2018

New data format should reduce as much as possible the cost of deserialization: some generality will be lost for the sake of improved speed

- **Simple, flat**: numbers only (no classes), use tables, cross-reference via numeric indices
- **Columnar**: SoA in-memory structure for better growing/shrinking and vectorization
- **Extensible**: base format is immutable, but easily extensible because it's SoA
- **Chunked**: a single timeframe can be divided in smaller units processable in parallel
- **Zero size for null objects**: filtered-out fields do not use RAM memory
- **Recompute, don't store**: do not store everything because recomputing may be cheaper
- **No data restructuring**: disk → memory → network should use similar representations

Task: Design and implementation of prototype data layout

Period: 07.2019 – 02.2019

- Define minimal universal data set (AOD) for all analyses: Ruben @ WP1, 01.11.2018
- Extract flat Root tree containing the minimal data set and representing the time frame AOD: prototype is ready + analysis task
- Convert the Root representation to Apache Arrow tables and test the functionality: prototype is ready, see later
- Convert the Root representation to SOAContainer data and test the functionality: in progress, expected in 05.2019
- Convert the Root representation to Libflatarray data and test the functionality: not started, for the moment is optional

The Analysis Object Data (AOD) format

Goal:

- minimize the information kept on AOD to save disk space
- **maximize performances with light simple flat data-objects instead of heavier C++ objects**

Current idea is to have flat data tables, initial implementation based on "Ruben's table".

- **Barrel track table = standard helix parameters of each track**
- **Covariance table for barrel tracks**
- **Extra barrel track table = more detailed info like track chi2, number of clusters, PID signal ..**
- **Muon track table**
- **Calorimeter track table**
- **"Vertex" table = info about the collision vertex**
- **Other "small" tables: FIT, ZDC**

Associating objects (e.g. tracks) to vertices

In each timeframe:

- 22 ms of data-taking
- ~1000 PbPb collisions
- ~Millions of tracks per dataframes

→

In AOD tables, each candidate is associated to a given vertex in the vertex table

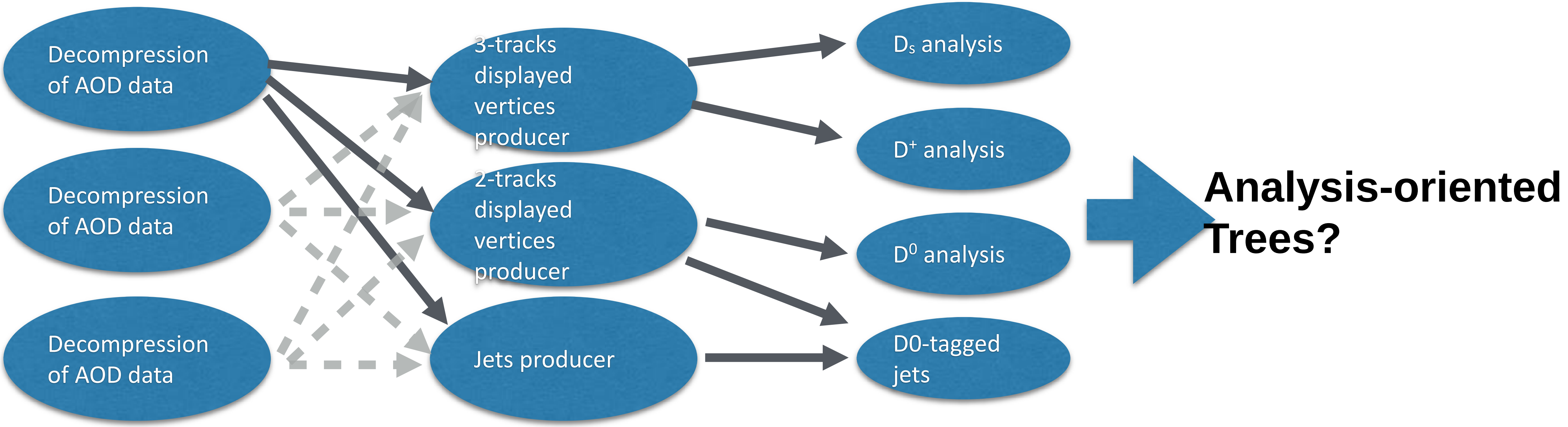
- done in the reconstruction level
- same track can be re-associated to more vertices!

Detailed presentation of Giulio and Gian Michele during EP-AIP meeting, 11.03.2019

A draft of structure for Run3 analysis

Need to develop and optimize a analysis structure that fully exploit the characteristics of the new Framework:

- maximal “sharing" of common processes and optimal use of shared memory



IMPORTANT: should we have an extra analysis layer on skimmed Trees? Yes!

- e.g. skimmed Dataframe analysis-oriented (e.g. HF Tables, Jet …) that can be processed in HPC clusters could dramatically reduce the analysis-cycle
- Need to study an effective compressed format, estimate size and develop optimised analysis software
- Develop a bookkeeping system for storage/skimming

Work in this direction with 2018 data is being carried out on a 32-core server with GPU that ALICE CERN group recently bought with Torino and Utrecht

Multiple possibilities being investigated

Fully declarative

RDataFrame Based solution

A mix of all the previous ones

Traditional analysis

Python Pandas

Vectorised "skin" based solution

Traditional event loop on proxy objects

Traditional way of doing things will always be possible

Multiple possibilities being investigated

Fully declarative

RDataFrame Based solution

A mix of all the previous ones

Python Pandas

Vectorised "skin" based solution

Traditional event loop on proxy objects

Traditional analysis

What follows is a elaboration on how to map the analysis on RDataFrame, following discussions from the presentation of [Danilo Piparo @ AIP meeting].

Single and **nested** loops with RDataFrame

Get an RDataFrame iterating on candidates obtained via O2

```
auto candidates = o2::analysis::doSingleLoopOn(input);
```

Select Good candidates

```
auto filtered = candidates.Filter("cand_type & 1");
```

Fill an histogram

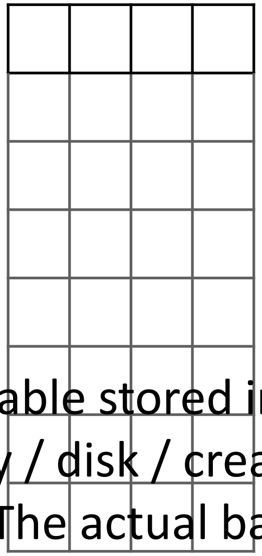
```
auto h1 = filtered.Histo1D("inv_mass");
```

Draw it

```
h1->Draw();
```

⟵ Event loop actually runs here.

RDataFrame internals



Actual table stored in memory / disk / created on the fly. The actual backend will depend on the context.

RDataSource

RDataSource abstracts from the user the act of fetching row from a table. Different implementations depending on the backend.

RDataFrame

RDataFrame abstracts from the user how to iterate on a set of rows and how actions are performed on them.

Role of the analysis framework: provide helpers to construct useful views on the data, using the above building blocks.

Only one of the possibilities...

Fully declarative

RDataFrame Based solution

A mix of all the previous ones

Traditional analysis

Python Pandas

Vectorised "skin" based solution

Traditional event loop on proxy objects

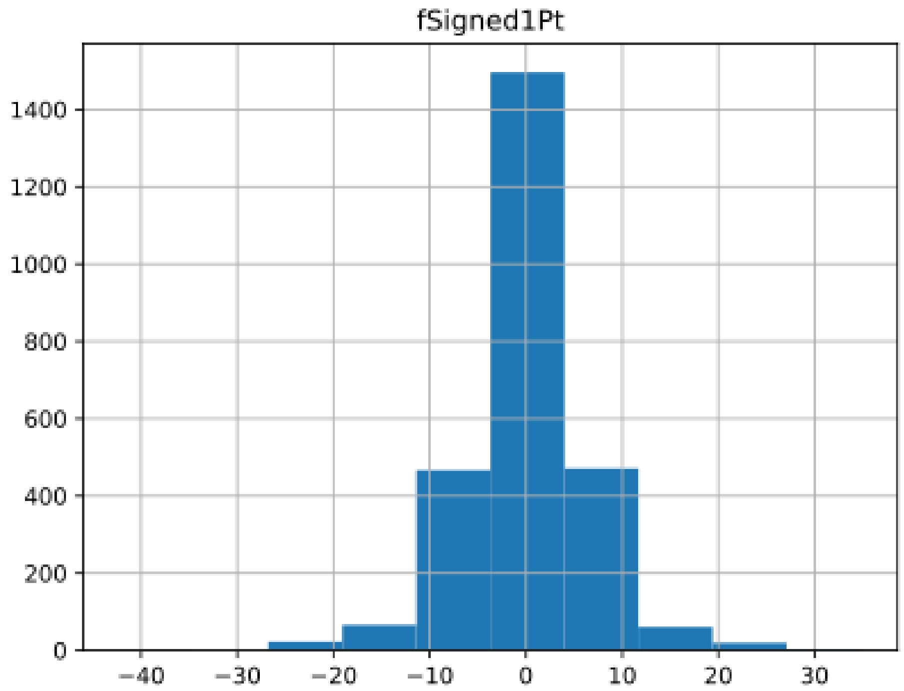
Quick preview of a python based solution in the presentation of [Giulio @ ALICE Computing week]

Python integration preview

```
import pyarrow as pa
import matplotlib.pyplot as plt
import sys

reader = pa.ipc.open_stream(pa.PythonFile(sys.stdin))
t = reader.read_next_batch()
df = t.to_pandas(zero_copy_only=True)
h1 = df.hist(column='fSigned1Pt')
plt.savefig('figure.pdf')
```

If we use Apache Arrow as backing store for the messages being exchanged we get integration with Python Pandas and many others for (almost) for free.



Single Run2 ESD event converted to Run3 AOD.
2607 tracks, 10 bins, not for publication. ;-)

Ongoing Framework level Efforts

Performance optimisation (*mostly ROOT team with our contributions*):

- *Bulk reads (us & ROOT team)*
- *Vectorisation (ROOT team)*
- *GPU support (ROOT team)*
- *Fast path in RCombinedDS for common analysis cases (us)*
- *Profiling of the RDataFrame solutions w.r.t. the other ones.*

Helpers for analysis (us)

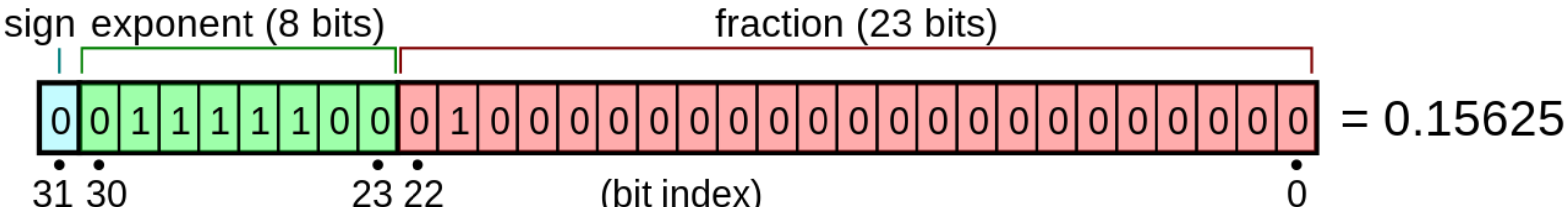
- *Filtered collections*
- *Triple / nth-ple loops*
- *Ability to plug current analysis tasks at the end of RDataFrame processing*

Run2 ESD to Run3 converter: first v ersion done, see the presentation of Giulio

- *No need for intermediate files*
- *Support O2/DPL*
- *Support Python Pandas & Tensorflow*

Storage with reduced precision (truncated fraction)

Used by CMS (discovered by Giulio)



107 PbPb events from 2015, no trigger selection

Type	Size	Truncation	Zip	Fraction, %
Ru1/2 ESD	510222529	no	yes	100.0
O2 AOD	50089236	no	yes	9.8
O2 AOD	47382919	4 bits	yes	9.3
O2 AOD	42018412	8 bits	yes	8.2
O2 AOD	37870154	12 bits	yes	7.4
O2 AOD	30360347	16 bits	yes	6.0

Task: Develop interfaces to access the flat data representations

Period: 02.2019 – 05.2019

- Data manager and reader for Run1/Run2 backward compatibility
- “Skins” for the Apache Arrow data access
- SOAContainer “skins”
- “Skins” for libflatarray (optional)
- On-the-fly calculation of derived quantities (primary and secondary vertex positions and covariance matrices, etc.)
- All these tasks probably will be completed with some delay

Task: Define and reimplement set of reference analyses for benchmarking

Period: 11.2018 – 06.2019

Several candidates are identified:

- Minijet analysis in small systems
- Investigation of longitudinal and azimuthal structure of the near side jet peak in Pb-Pb collisions
- Particle flow analysis
- Open charm analysis
- The conversion depends partially on the previous task

Task: Investigation of RDataFrame-based analysis

Period: 03.2018 – 07.2019

- RDataFrame for skimming and slimming: partially done
- RDataFrame with cartesian product of tables (for nested loops): done, pull request to be merged in Root6
- RDataFrame reimplementation of the analysis examples from the previous task: ongoing
- Performance studies and conclusion on the suitability of RDataFrame for Run3 analysis: not started

Task: Reimplementation of analysis tasks using DPL – analysis devices

Period: 03.2018 – 09.2019

- Prototype of multiple IO devices, multiple analysis devices and data sync to store the results: prototype presented @ CHEP2018
- Reimplementation of the reference analyses from p.3: ongoing
- Reimplementation of simple analysis train: not started
- Performance measurements of DPL-based trains on analysis prototype facility: not started

Task: Development of the Lego train system for Run3

Period: 09.2018 – 05.2020

- Adapt the system to the analysis devices from p.5: not started
- Redesign of the Web interface and data base backend: initial proposal prepared by Markus
- Continuous integration and automatic train testing: not started
- R&D on dynamic reconfiguration of the Lego trains: not started

Task: Development related to the future usage of Machine Learning

Period: 11.2018 – 12.2019

- Direct Python integration: done in Root6, ALICE prototype presented by Gian Michele
- Data exchange via Apache Arrow tables: prototype presented by Giulio
- R&D on using Apache Spark and Pandas for ML analysis: ongoing,
- Many other ML activities, see the presentation of Gian Michele

Task: Development of additional GRID analysis features

Period: 07.2019 – 12.2019

- Efficient use of multicore job queues with analysis devices: not started
- Performance measurements and benchmarking: initial data provided by Costin

Task: Analysis data challenge

Period: 01.2020 – 03.2020

- Large scale tests of the functionality in local, analysis facility and GRID modes
- Performance measurements and benchmarking
- Recommendations for the usage of different resources

Task: Final design and implementation of analysis facilities

Period: 04.2020 – 10.2020

- This task depends on the outcome of the analysis data challenge and the general policy wrt the analysis facilities