Status and Roadmap of ROOT's RNTuple and DAOS

Javier López-Gómez⋆ – EP R&D fellow, CERN
Vincenzo Eduardo Padulano⋆⋆ – CERN

DUG'21, 2021-11-19

ROOT project, EP-SFT, CERN
`http://root.cern/`

⋆ `<javier.lopez.gomez@cern.ch>`

⋆⋆ `<vincenzo.eduardo.padulanocern.ch>`

About me

- PhD in Computer Science and Technology
- Strong focus in low-level: electronics/embedded systems, kernel development, compilers, RE, etc.

Currently at CERN's ROOT project working in:

- Improvements and bug fixes to the `cling` C++ interpreter

```
[cling]$ std::cout << "Hello DUG'21!" << std::endl;
Hello DUG'21!
```

- Improvements to RNTuple, the ROOT's new columnar I/O system
- Maintainer of the RNTuple DAOS backend

Contents

1 Introduction

2 RNTuple and DAOS

3 Evaluation

4 RNTuple caching & DistRDF on DAOS

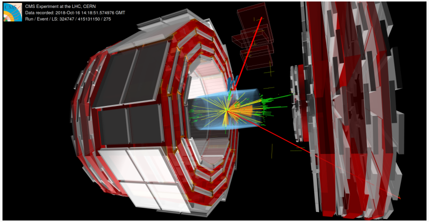
5 Conclusion

Introduction

What we do at CERN

- High-energy physics studies laws governing our universe at the smallest scale.
- CERN experiments observe particle interactions by colliding particles.
- LHC collides protons that move in opposite directions.

- Detectors are similar to a 100 MP camera taking a picture every 25 ns.
- 10^9 collisions/sec generating $\sim$ **10 TB/s**.
- Processing:
  - Online: filtering step. Part of the detector read-out.
  - Offline: distributed; disk storage at different LHC compute centers around the globe.

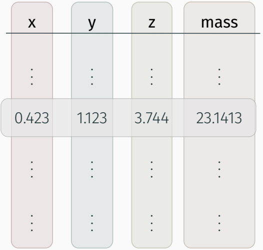

CMS
CMS Experiment at the LHC, CERN
Data recorded: 2018-Oct-16 14:18:51.574976 GMT
Run / Event / LS: 324747 / 415131150 / 275

ROOT's TTree and RNTuple

HEP analyses typically require access to a subset of the columns: column-wise storage[1].

- **ROOT's TTree**: used for 25 years (**1+ EB** stored in ROOT files!).
- However, not designed to fully exploit modern hardware.

- **RNTuple**: R&D project to evolve the TTree I/O.
- Object stores are first-class citizens.

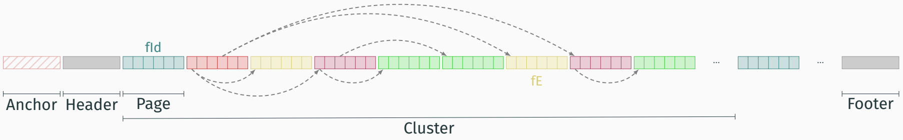


x	y	z	mass
⋮	⋮	⋮	⋮
0.423	1.123	3.744	23.1413
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

[1] See also: Apache Arrow/Parquet

RNTuple and DAOS

RNTuple: on-disk file format



```
struct Event {
  int fId;
  vector<Particle> fPtcls;
};
struct Particle {
  float fE;
  vector<int> fIds;
};
```

Pages: Array of fundamental types (maybe compressed); ∼tens of kB (tunable at write time).

Cluster: comprises pages for a certain range of rows, e.g. 1–1000.

Page group: pages on a given cluster that contain instances of the same data member.

Alternatives for mapping clusters/pages to DAOS

- One page per OID in a single *akey*. Constant *dkey*.
- One cluster per OID and one *akey* per page in the cluster. Constant *dkey*.
- One cluster per OID (2). Same as above, but varying *dkey* (e.g. one *dkey* per page group).

RNTupleReader: file vs. DAOS

Only requires the replacement of the file path

```
auto ntuple = RNTupleReader::Open("DecayTree",
            "./B2HHH~zstd.ntuple");
```

to a `daos://` URI

```
auto ntuple = RNTupleReader::Open("DecayTree",
            "daos://e6f8e503-e409-4b08-8eeb-7e4d77cce6bb/b4f6d9fc-e081-41d4-91ae-
                 41adf800b537");
```

Evaluation

Evaluation

Test environments

- **CERN openlab:** 3 servers, 2 clients. Intel Omni-Path.
- **HPE Delphi:** 2 servers, 9 clients. Mellanox InfiniBand.

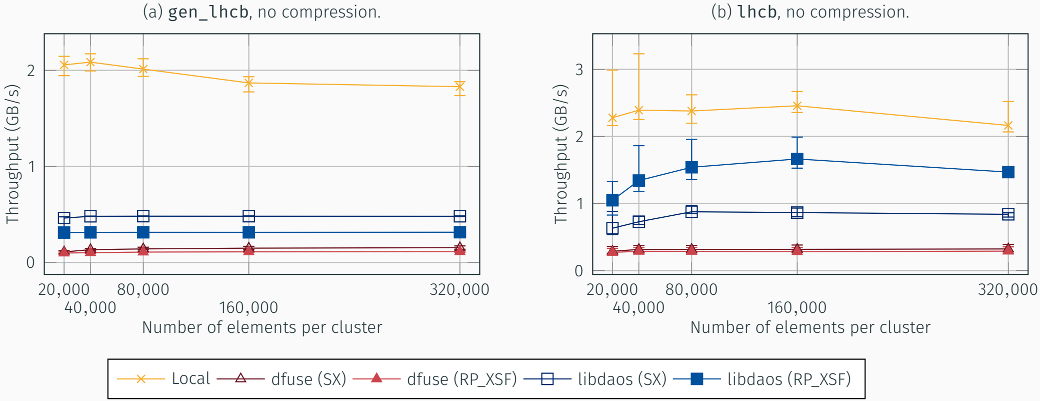
Test cases

Steps: (a) move data into DAOS, and
(b) run analysis using imported data.

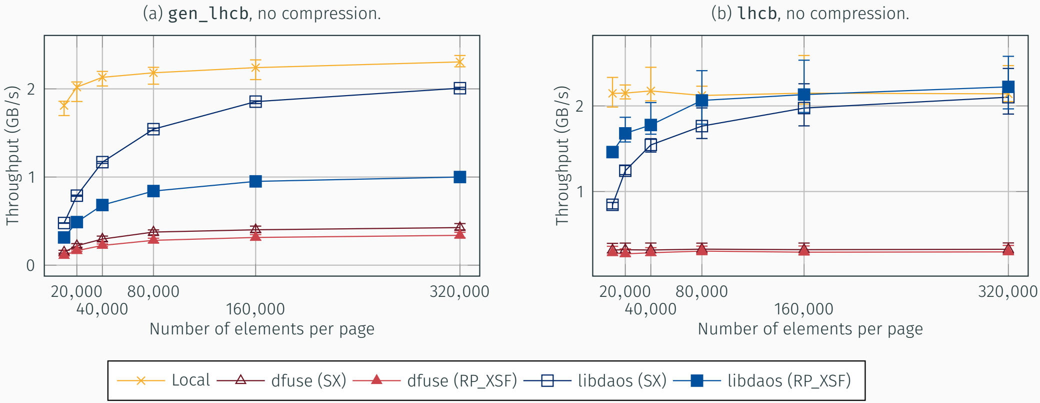
Conditions:

1. **Constant page size, increasing cluster size.** Observe the effect of queuing many small read operations.
2. **Increasing page size, constant cluster size.** Impact of the I/O request size on the throughput.

CERN openlab: one page per OID, constant page size, increasing cluster size



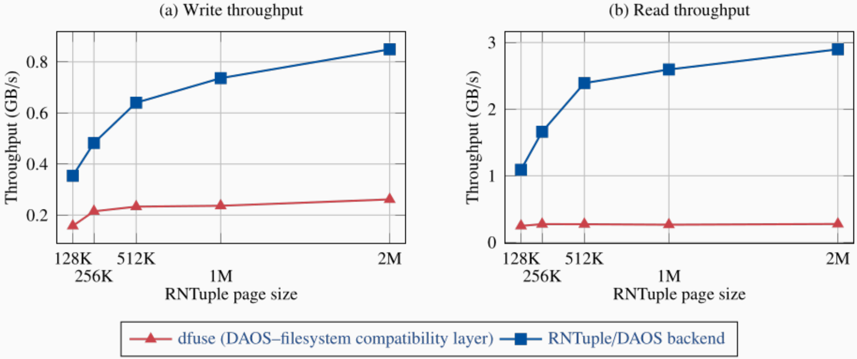
CERN openlab: one page per OID, increasing page size, constant cluster size



HPE Delphi: one page per OID, increasing page size, constant cluster size

- Preliminary tests: poor performance of ∼550MB/s.
- Wrong use of event queues: EQ created (destroyed) before (after) each bulk read.

Single-process, single-thread results after patching:

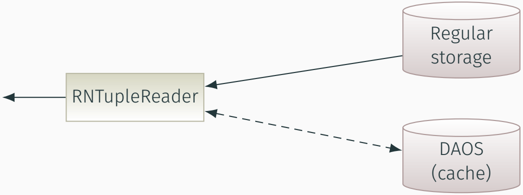


Summary

- Higher read throughput with large pages (larger transfer size).
- RNTuple libdaos-based backend outperforms `dfuse` in our tests.
- Room for improvement, e.g.
  - combine EQ with `daos_obj_fetch()` for multiple akeys (implies using a “One cluster per OID” mapping)
  - multiple, maybe per-thread, event queues (?)

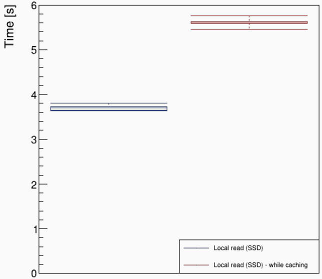
RNTuple caching & DistRDF on DAOS

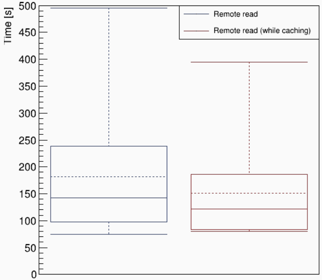
RNTuple caching on DAOS

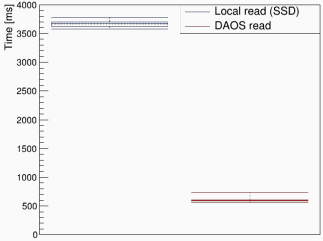

RNTupleReader

Regular storage

DAOS (cache)







ROOT's RDataFrame and DistRDF

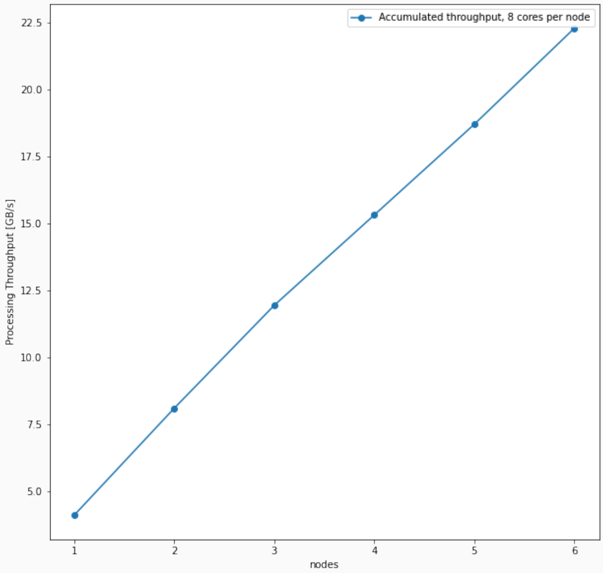
- RDataFrame provides a convenient, declarative interface for HEP analyses, e.g.

```
from ROOT import RDataFrame
df = RDataFrame(dataset)
df2 = df.Filter("x > 0")
        .Define("r2", "x*x + y*y");
rHist = df2.Histo1D("r2");
```

- Multi-threaded, but only works in a single node

- DistRDF extends RDataFrame to run distributed computation using Spark or Dask
- In our experiment: data was stored in DAOS

DistRDF on DAOS: evaluation



Conclusion

Conclusion

- RNTuple architecture decouples storage from serialization/representation.
- Object stores are first-class: we expect DAOS to have an important role in HPC centers.
- RNTuple DAOS backend already merged into ROOT's '`master`' branch.

Work in progress

1. Ongoing efforts to improve the read throughput.
2. Data mover: importing large amounts of existing HEP data into DAOS.

Status and Roadmap of ROOT's RNTuple and DAOS

Javier López-Gómez* – EP R&D fellow, CERN
Vincenzo Eduardo Padulano** – CERN

DUG'21, 2021-11-19

ROOT project, EP-SFT, CERN
`http://root.cern/`

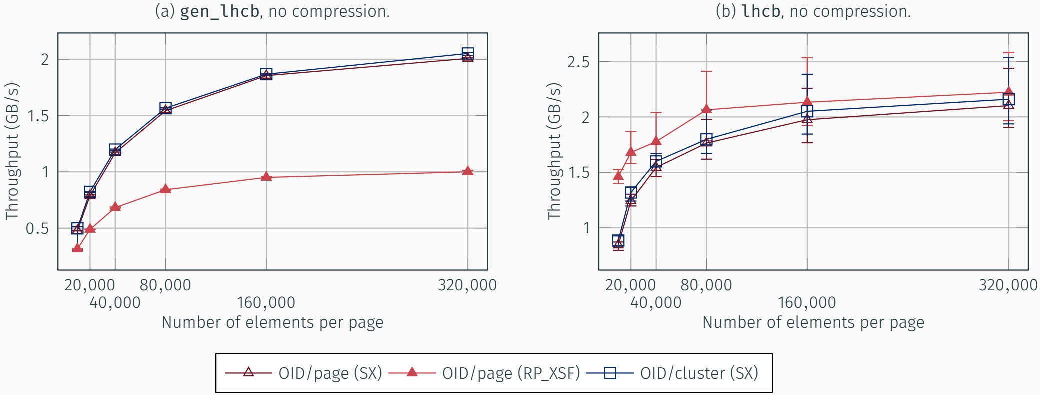
* <javier.lopez.gomez@cern.ch>

** <vincenzo.eduardo.padulanocern.ch>

Why invest in tailor-made I/O sub system (TTree / RNTuple)

- Capable of storing the HENP event data model: nested, inter-dependent collections of data points
- Performance-tuned for HENP analysis workflow (columnar binary layout, custom compression etc.)
- Automatic schema generation and evolution for C++ (via cling) and Python (via cling + PyROOT)
- Integration with federated data management tools (XRootD etc.)
- Long-term maintenance and support

CERN openlab: one cluster per OID, increasing page size, constant cluster size


(a) `gen_lhcb`, no compression.

(b) `lhcb`, no compression.