Storing High-Energy Physics data in DAOS

Javier López Gómez – **CERN fellow**
<javier.lopez.gomez@cern.ch>

DUG '20, 19th November 2020

ROOT project,
EP-SFT (SoFTware Development for Experiments),
CERN

http://root.cern/

Contents

1 Introduction

2 RNTuple 101

3 RNTuple DAOS backend

4 First evaluation

5 Conclusions

Introduction

High-Energy Physics (HEP)

- High-Energy Physics studies laws governing our universe at the smallest scale: fundamental particles, forces and its carriers, mass, etc. The "Standard model" describes these particles/interactions.
- CERN experiments observe particle interactions (typically by colliding particles at high-energies).
- HEP data = detector observations.

Large Hadron Collider (LHC)

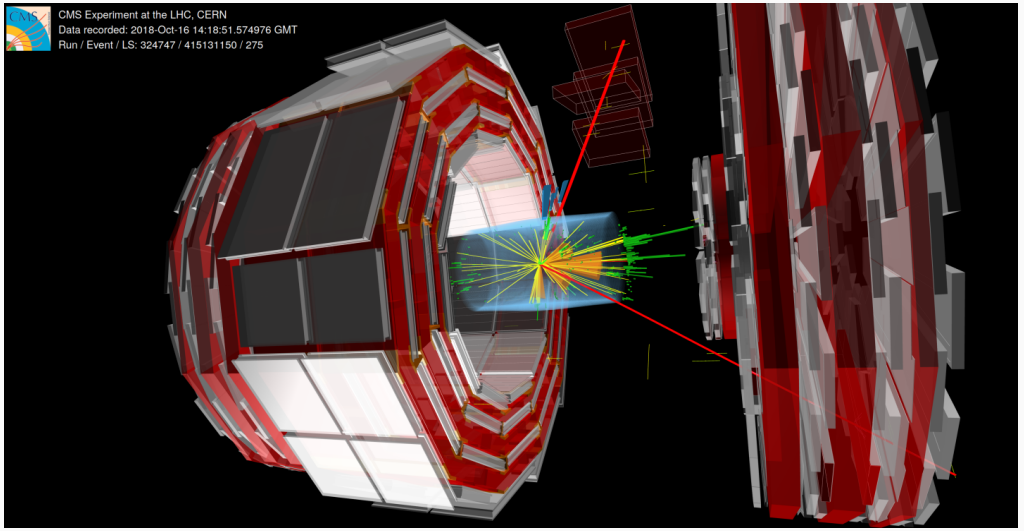


Figure 1: Graphical representation of a CMS event.[1]

- 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.

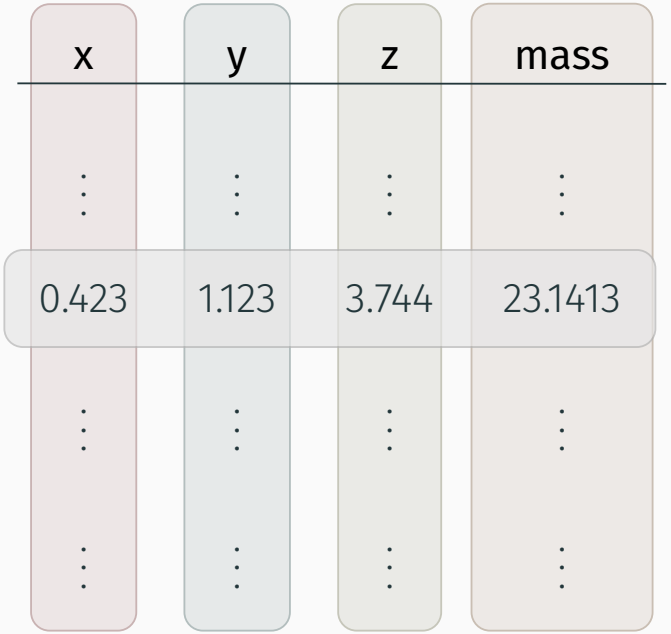
[1] `http://opendata.cern.ch/visualise/events/cms`

ROOT project

- ROOT: open-source data analysis framework written in C++. Provides C++ interpretation, **object serialization** (I/O), statistics, graphics, and much more.
- PyROOT provides dynamic C++ $\leftrightarrow$ Python bindings.
- **ROOT I/O:** row-wise/column-wise storage of C++ objects.

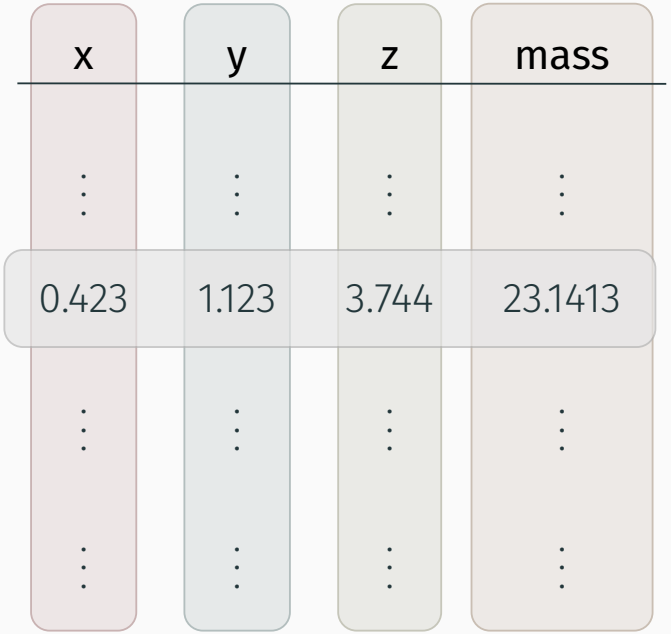
TTree and RNTuple

- HEP data analysis often only requires access to a subset of the properties of each event.
- Row-wise storage is inefficient. TTree organizes the dataset in columns that contain any type of C++ object.
- **1+ EB** of HEP data stored in TTree ROOT files.
- TTree has been there for 25 years. RNTuple is the R&D project to replace TTree for the next 30 years.
- Object stores are first-class.



TTree and RNTuple

- HEP data analysis often only requires access to a subset of the properties of each event.
- Row-wise storage is inefficient. TTree organizes the dataset in columns that contain any type of C++ object.
- **1+ EB** of HEP data stored in TTree ROOT files.
- TTree has been there for 25 years. RNTuple is the R&D project to replace TTree for the next 30 years.
- Object stores are first-class.



RNTuple 101

RNTuple architecture

Event iteration

Looping over events for reading/writing

Logical layer / C++ objects

Mapping of C++ types onto columns, e.g.
`std::vector<float>` $\mapsto$ index column and a value column

Primitives layer / simple types

"Columns" containing elements of fundamental types (`float`, `int`, ...) grouped into (compressed) pages and clusters

Storage layer / byte ranges

POSIX files, object stores, ...

Storage layer: access to the header (= schema), the pages, and the footer (= location of pages).

File backend: on-disk format


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

To put it simple...

Anchor: specifies the offset and size of the header and footer sections.

Header: schema information.[2]

Footer: location of pages and clusters.[2]

Pages: little-endian fundamental types (possibly packed, e.g. bit-fields) —typically in the order of tens of KiB.[2]

[2] This element may be compressed or not.

RNTuple DAOS backend

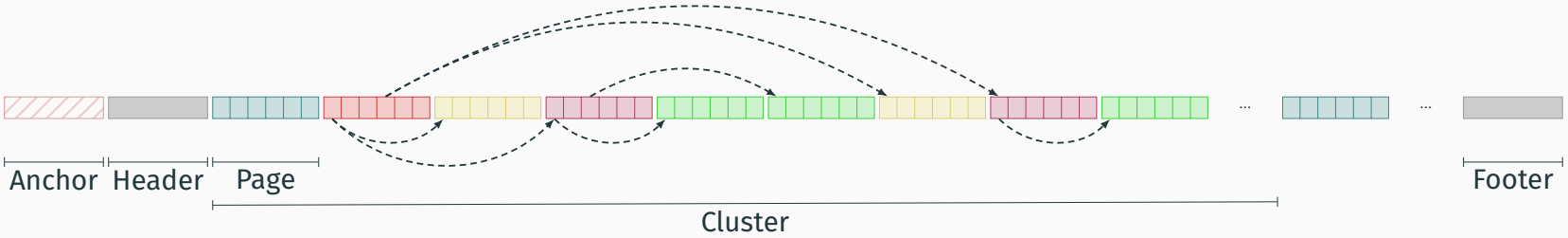
libdaos C++ interface classes

To simplify resource management, we wrote C++ wrappers for part of libdaos functionality.

```
auto pool = std::make_shared<RDaosPool>(
    "e6f8e503-e409-4b08-8eeb-7e4d77cce6bb", "1");
RDaosContainer cont(pool, "b4f6d9fc-e081-41d4-91ae-41adf800b537");

std::string s("foo bar baz");
cont.WriteObject(daos_obj_id_t{0xcafe4a11deadbeef, 0}, s.data(), s.size()
    , /*dkey =*/ 0, /*akey =*/ 0);
```

DAOS backend: mapping things to objects




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

- Each RNTuple page is stored in a separate object. The UUID is sequential starting from `00000000-0000-0000-0000-000000000000`.
- **Header**, **Footer**, and **Anchor** are stored in three different objects with reserved UUIDs.

Usage: RNTuple/file vs. RNTuple/DAOS

From the user's perspective...

```
auto model = RNTupleModel::Create();
auto ntuple = RNTupleReader::Open(std::move(model),
            "DecayTree",
            "./B2HHH~zstd.ntuple");

auto viewH1IsMuon = ntuple->GetView<int>("H1_isMuon");
auto viewH2IsMuon = ntuple->GetView<int>("H2_isMuon");
auto viewH3IsMuon = ntuple->GetView<int>("H3_isMuon");
```

Usage: RNTuple/file vs. RNTuple/DAOS

From the user's perspective...

```
auto model = RNTupleModel::Create();
auto ntuple = RNTupleReader::Open(std::move(model),
            "b4f6d9fc-e081-41d4-91ae-41adf800b537",
            "daos://e6f8e503-e409-4b08-8eeb-7e4d77cce6bb/1");

auto viewH1IsMuon = ntuple->GetView<int>("H1_isMuon");
auto viewH2IsMuon = ntuple->GetView<int>("H2_isMuon");
auto viewH3IsMuon = ntuple->GetView<int>("H3_isMuon");
```

First evaluation

Test environment

Our evaluation ran on CERN OpenLab DAOS test machines:

- 3 DAOS servers, 1 DAOS head node.
- interconnected by an Omni-Path Edge Switch 100 Series | 24 ports.

System specifications	
CPU	Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz
CPU per node	24 cores/socket, 2 sockets, 2 threads/core (HT enabled)
Core frequency	Base: 1.0 GHz Range: 1.0GHz - 3.9GHz
Numa nodes	node0: 0-23,48-71 node1: 24-47,72-95
System Memory	12x 32GB DDR4 rank DIMMs
Optane DCPMM	12x 128GB DDR4 rank DIMMs
Optane FW version	01.02.00.5395
BIOS	version: SE5C620.86B.02.01.0011.032620200659 date: 03/26/2020
Storage	4x 1 TB NVMe INTEL SSDPE2KX010T8
HFI	1x Intel Corporation Omni-Path HFI Silicon 100 Series.
HFI Firmware	Termal Management Module: `10.9.0.0.208`; Driver: `1.9.2.0.0`

Figure 2: Server nodes HW (`olcsl-*`)

System specifications	
CPU	Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz
CPU per node	24 cores/socket, 2 sockets, 2 threads/core (HT enabled)
Core frequency	Base: 1.0 GHz Range: 1.0GHz - 3.9GHz
Numa nodes	node0: 0-23,48-71 node1: 24-47,72-95
System Memory	12x 16GB DDR4 rank DIMMs
BIOS	version: SE5C620.86B.02.01.0011.032620200659 date: 03/26/2020
HFI	1x Intel Corporation Omni-Path HFI Silicon 100 Series.
HFI Firmware	Termal Management Module: `10.9.0.0.208`; Driver: `1.9.2.0.0`

Figure 3: Client node HW (`olsky-03`)

`dfuse` simple benchmark (ofi+sockets)

⚠ These results are preliminary and might not be reliable.

	Block size					
	4K	8K	16K	512K	1M	4M
Seq. Write	**7.62**	14.42	27.44	189.21	205.10	**225.62**
Seq. Read	2.62	5.04	9.21	116.86	147.7 9	188.90
Random Write	7.30	**14.67**	**27.63**	**199.68**	**209.17**	211.40
Random Read	2.16	4.20	7.92	120.91	162.7 0	211.12

Table 1: `dfuse` read/write benchmark (**in MiB/s**)

- Far from the 34.2 Gbits/sec (4.275 GiB/s) achieved by `iperf`.
- Path lookup not bad; around **700+** `open()`/`creat()` calls/s.

RNTuple (ofi+sockets)

⚠ These results are preliminary and might not be reliable.

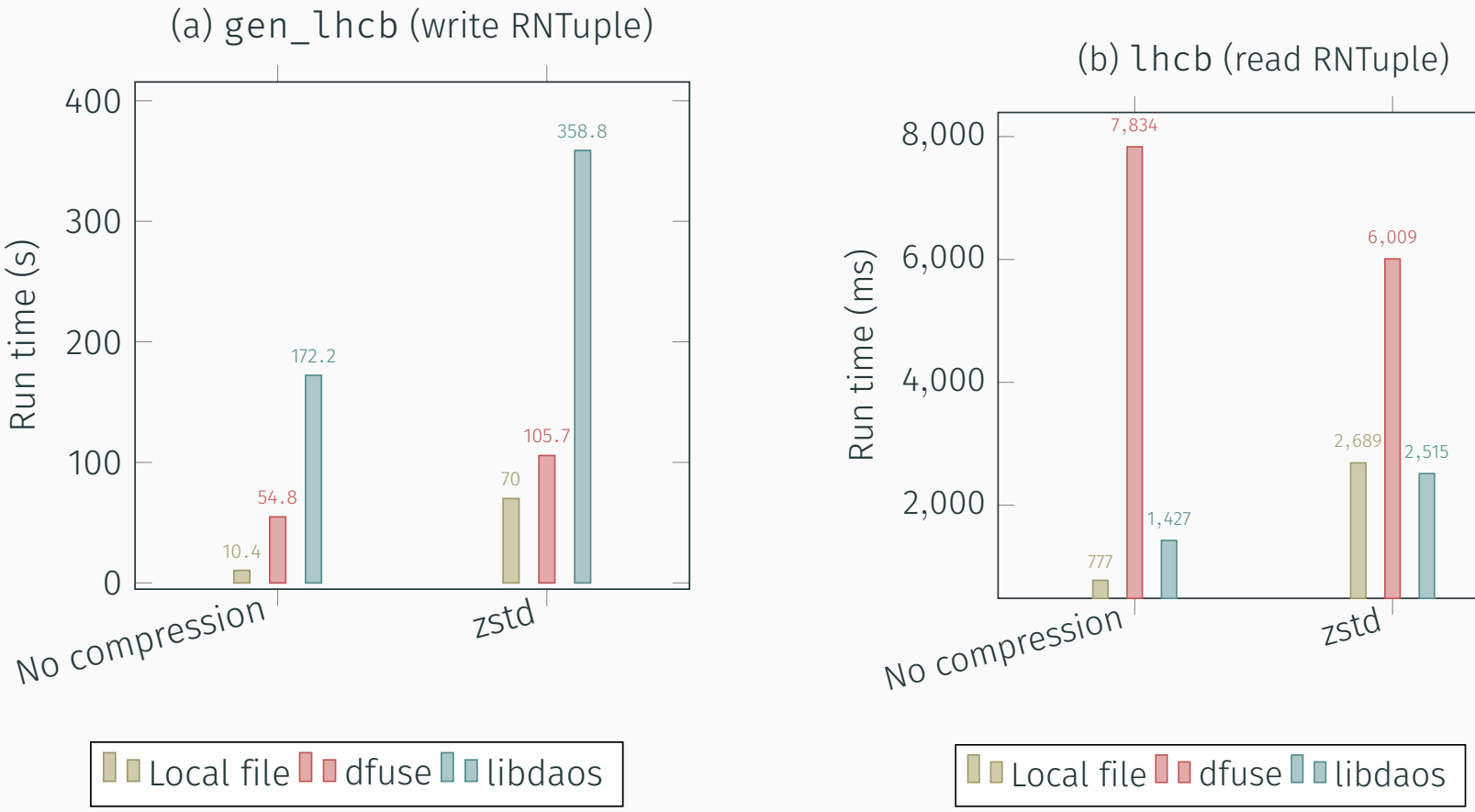


Figure 4: RNTuple benchmark on LHCb data (ofi+sockets).[2][3]

[2] Input data size: 1.5 GiB (uncompressed) / 1007 MiB (zstd).
[3] `https://github.com/jblomer/iotools`

RNTuple (ofi+psm2)

⚠ These results are preliminary and might not be reliable.

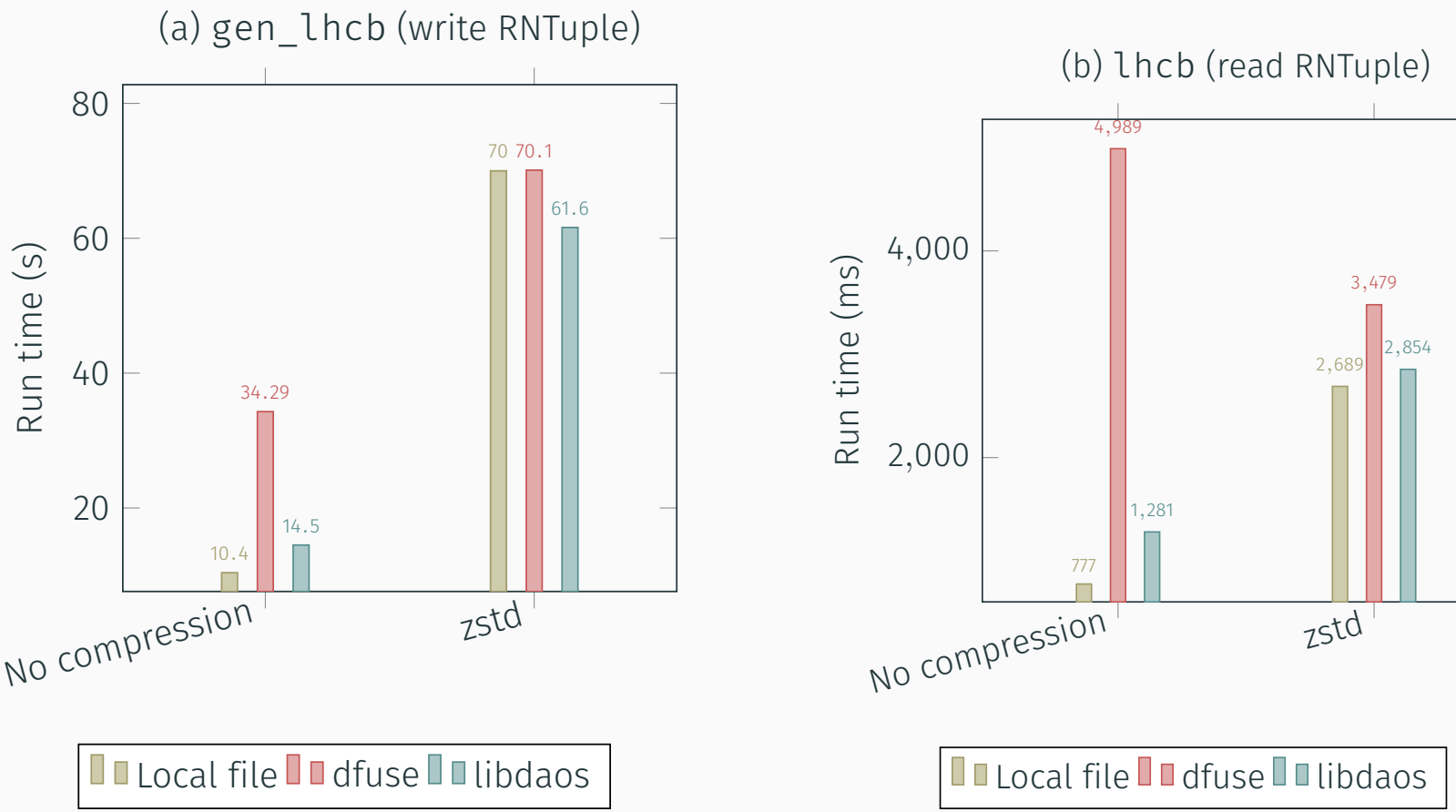


Figure 5: RNTuple benchmark on LHCb data (ofi+PSM2).[4][5]

[4] Input data size: 1.5 GiB (uncompressed) / 1007 MiB (zstd).
[5] `https://github.com/jblomer/iotools`

Conclusions

Conclusions

- 1+ EB of HEP data in ROOT files (TTree). RNTuples replaces TTree columnar storage for the next 30 years.
- RNTuple architecture decouples storage from serialization/representation. Object stores are first-class.
- First prototype implementation of an Intel DAOS backend. Currently "1 Page == 1 Object" + constant dkey. Still some performance issues.

Next Questions:

1. How to maximize throughput (bulk reading/writing of pages)?
2. How to distribute pages appropriately, e.g. put together pages corresponding to the same data member?

Storing High-Energy Physics data in DAOS

Javier López Gómez – **CERN fellow**
<javier.lopez.gomez@cern.ch>

DUG '20, 19th November 2020

ROOT project,
EP-SFT (SoFTware Development for Experiments),
CERN

http://root.cern/