RNTuple on DAOS

EP R&D Software, 7.4—Efficient Analysis

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

EP R&D Software Working Group meeting, 2021-07-14

EP-SFT, CERN, Geneva, Switzerland

http://root.cern/

Contents

RNTuple Goals

Reminder: HENP Event Data I/O

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

RNTuple Goals

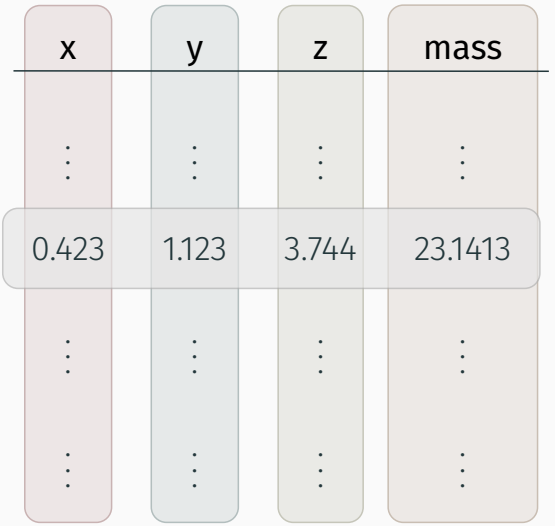
- Less disk and CPU usage for same data content
  - 25% smaller files, $\times$2-5 better single-core performance
  - 10GB/s per box and 1GB/s per core sustained end-to-end throughput (compressed data to histograms)
- Native support for object stores (targeting HPC)
- Lossy compression
- Systematic use of exceptions to prevent silent I/O errors

- Getting ready for a new hardware landscape: architectural heterogeneity, parallelism on all levels, blurring between device classes (e.g. active storage, NV-DIMMs)

Introduction

ROOT TTree and RNTuple

- HEP analyses typically require access to many events, but only a subset of their properties.
- **TTree** has been in use for 25 years (**1+ EB** stored in ROOT files!).
- However, not designed to fully exploit modern hardware.
- **RNTuple** is the R&D project to evolve the TTree I/O.
- Object stores are first-class citizens.



Object Stores: Motivation

Issues with traditional storage stack

- Designed for spinning disks (few IOPS): I/O coalescing, buffering, etc., became less relevant for modern devices.
- POSIX I/O (strong consistency), has been acknowledged as a major problem for parallel filesystem scalability.

Modern object stores overcome these limitations.

- GET and PUT primitives; objects accessed via a unique object identifier (OID).
- Object stores may have an important role in next-generation data centers.

Why Intel DAOS?

- Modern fault-tolerant object store optimized for high bandwidth, low latency, and high IOPS. Foundation of the Intel exascale storage stack.
- Optimal use of Intel Optane DC persistent memory and NVMe SSDs (access time in the order of μs).
- Argonne's Aurora[1] I/O system will be based on DAOS.
- Experience acquired supporting this in RNTuple can be reused for other object stores.

[1] https://alcf.anl.gov/aurora

RNTuple Overview

RNTuple: Architecture Overview

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

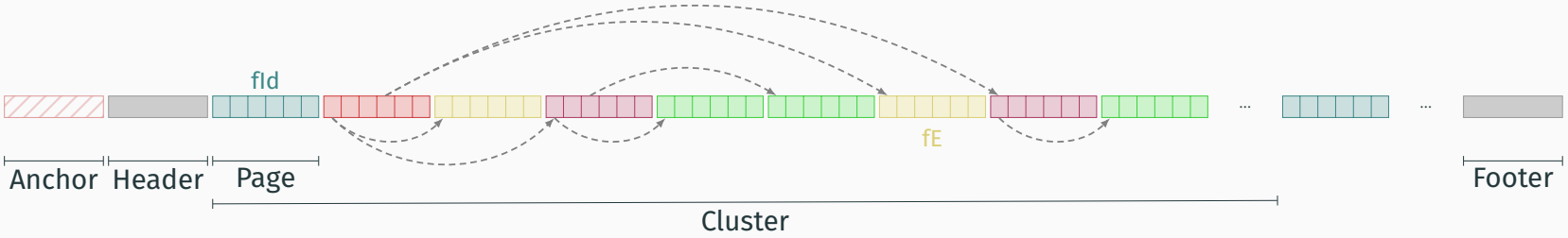
(`RPageSourceXxx`, `RPageSinkXxx`)

POSIX files, **object stores**, ...

Approximate equivalent of TTree and RNTuple classes:

TTree	$\approx$	RNTupleReader
		RNTupleWriter
TTreeReader	$\approx$	RNTupleView
TBranch	$\approx$	RField
TBasket	$\approx$	RPage
TTreeCache	$\approx$	RClusterPool

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); order of $\sim$ tens of KiB, but tunable at write time.

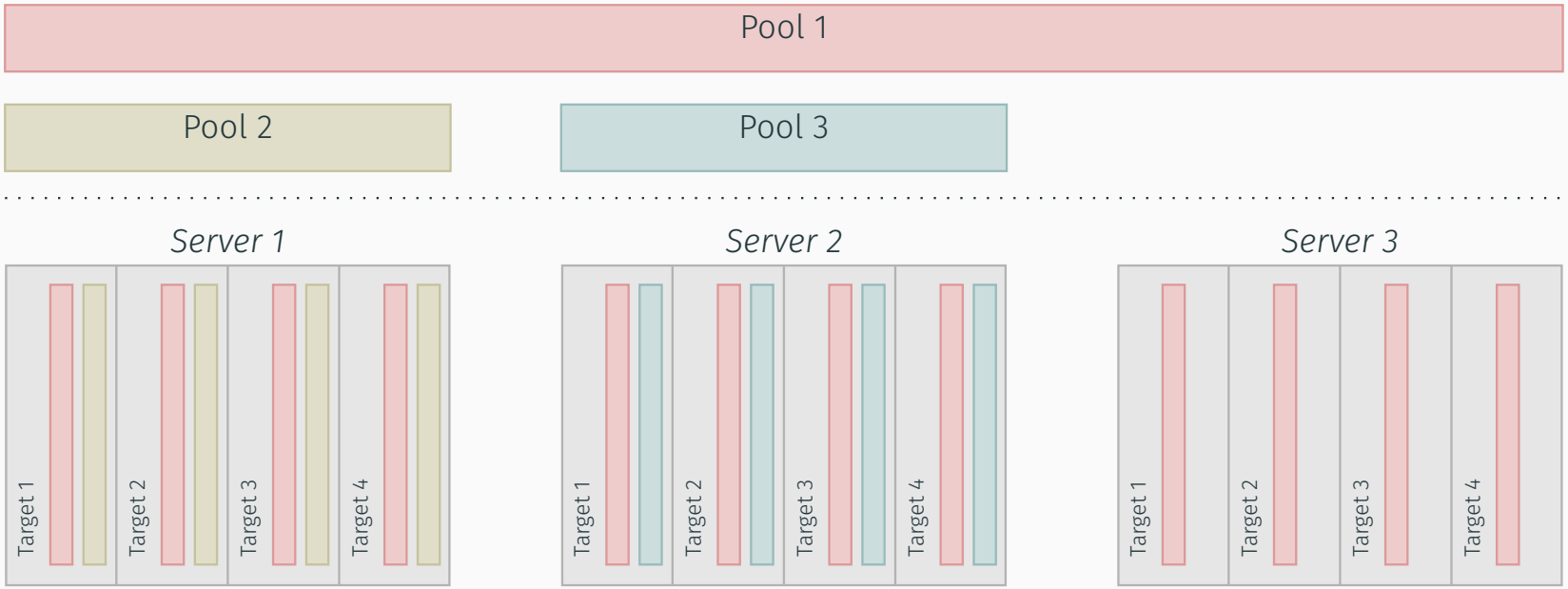
Cluster: Collection of pages for a certain range of events, e.g. 1–1000.

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

Anchor/Header/Footer: Schema information + location of pages/clusters.

RNTuple on DAOS

DAOS Overview: Architecture

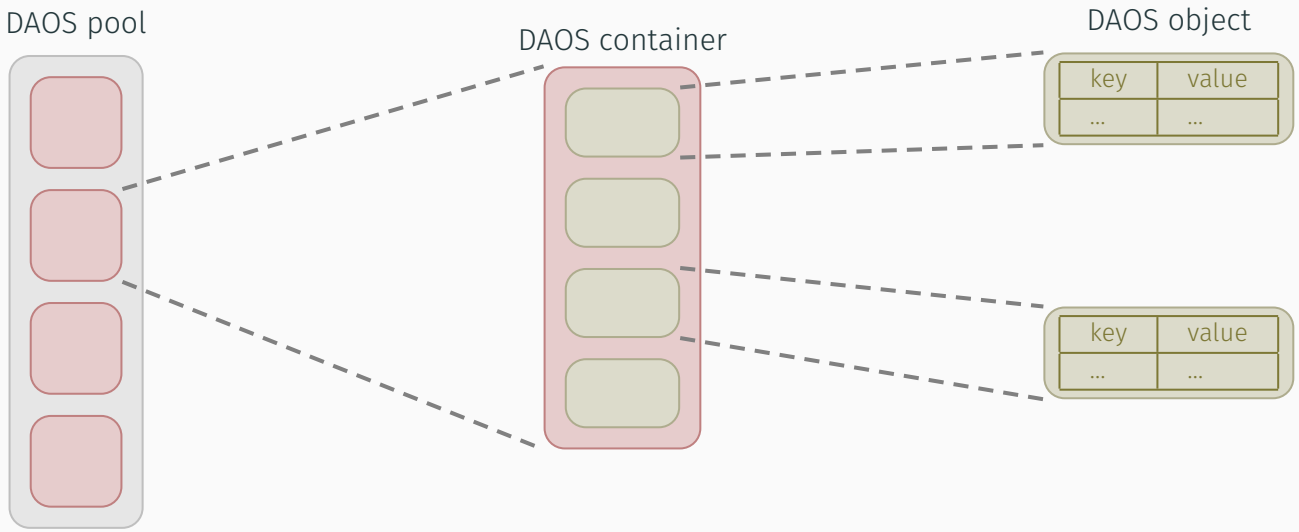


System: a set of DAOS servers connected to the same fabric.

Server: Linux daemon that exports locally-attached NVM storage. Listens on a management interface and 1+ fabric endpoints.

Target: static partition of storage resources (host controller, etc.). Avoids contention, as each target has its private storage that can be directly addressed over the fabric.

DAOS Overview: Pools, Containers and Objects



- **Object:** a Key–Value store with locality.
  - The key is split into **dkey** (distribution key) and **akey** (attribute key).
  - dkey affects data locality: DAOS guarantees that same dkey maps to same target.
- **Object class:** determines redundancy (replication/erasure code).

DAOS Overview: Compatibility Layer

Legacy software can still use DAOS through its compatibility layer, i.e.

- **POSIX filesystem (libdfs).** Can be used either through `libioil` (I/O call iterception) or `dfuse` (FUSE filesystem).
- **MPI–IO.** Provides DAOS support through a ROMIO driver (MPICH and Intel MPI).
- **HDF5, Apache Spark, ...**

...although throughput may not be on par to direct use of libdaos.

Back to RNTuple...

Alternatives for mapping Clusters/Pages to Objects[2]

OID-per-page. A sequential OID is assigned for each committed page; constant *dkey* and *akey*.

OID-per-cluster. *clusterindex* $\mapsto$ *OID*, *dkey* addresses individual pages in the cluster; constant *akey*.

Improved OID-per-cluster. *clusterindex* $\mapsto$ *OID*, *column* $\mapsto$ *dkey*, *akey* addresses individual pages.

[2] For the implementation, see: `RDaos.cxx`, `RDaos.hxx`, `RPageStorageDaos.cxx`, and `RPageStorageDaos.hxx`.

User (rather: file catalog) Interface: file vs. DAOS

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

auto    x = ntuple->GetView<double>("x");
auto    y = ntuple->GetView<double>("y");
auto    z = ntuple->GetView<double>("z");
auto mass = ntuple->GetView<double>("mass");

for (auto i : ntuple->GetEntryRange()) {
   //...
}
```

[3] UUIDs are not meaningful to users (common problem in object stores).

User (rather: file catalog) Interface: file vs. DAOS

```
auto ntuple = RNTupleReader::Open("DecayTree",
            "daos://e6f8e503-e409-4b08-8eeb-7e4d77cce6bb:1/b4f6d9fc-e081-
                41d4-91ae-41adf800b537");[3]

auto    x = ntuple->GetView<double>("x");
auto    y = ntuple->GetView<double>("y");
auto    z = ntuple->GetView<double>("z");
auto mass = ntuple->GetView<double>("mass");

for (auto i : ntuple->GetEntryRange()) {
   //...
}
```

[3] UUIDs are not meaningful to users (common problem in object stores).

Evaluation

Evaluation

Environment

Experiments ran on the CERN Openlab DAOS testbed:

- 3 DAOS servers, 1 client node
- HFI: Omni-Path Silicon 100 + Omni-Path Edge 100 24-port switch

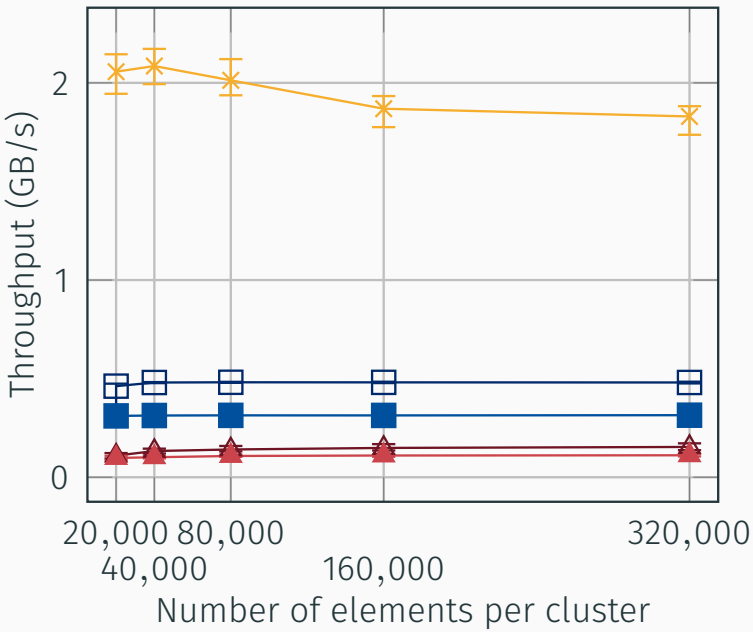
Test cases

Store/retrieve a 1.5 GB LHCb dataset into RNTuple, in a variety of conditions:

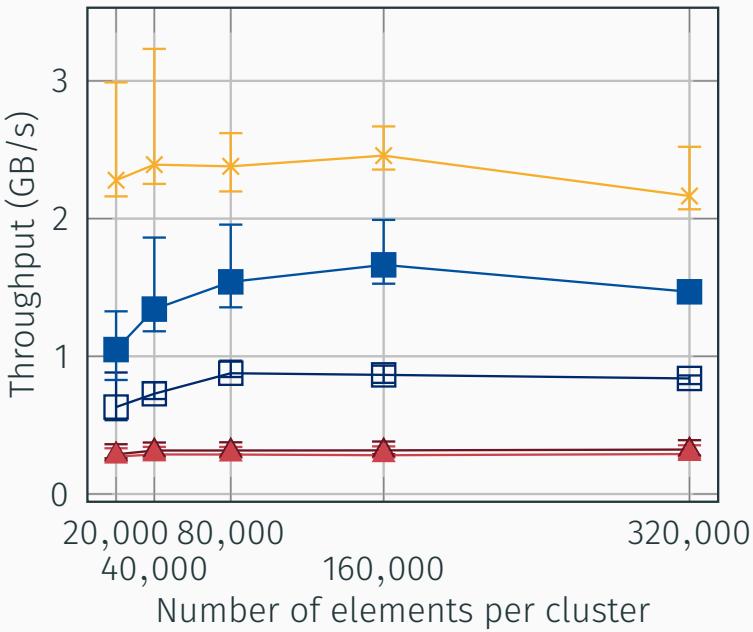
1. **Constant page size (10 000), increasing cluster size.** Observe the effect of queuing many small read operations.
2. **Increasing page size, constant cluster size (320 000).** Impact of the I/O request size on the throughput.

OID-per-page, constant page size, increasing cluster size

(a) `gen_lhcb`, no compression.

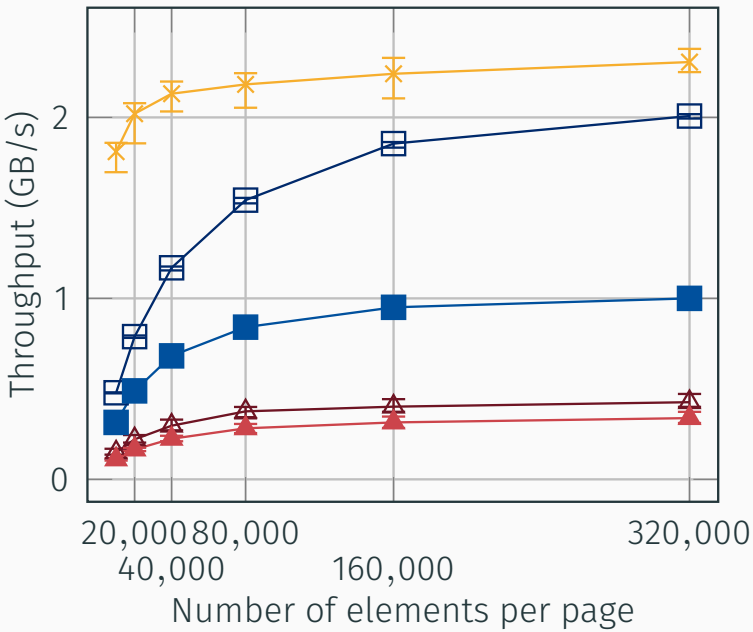


(b) `lhcb`, no compression.

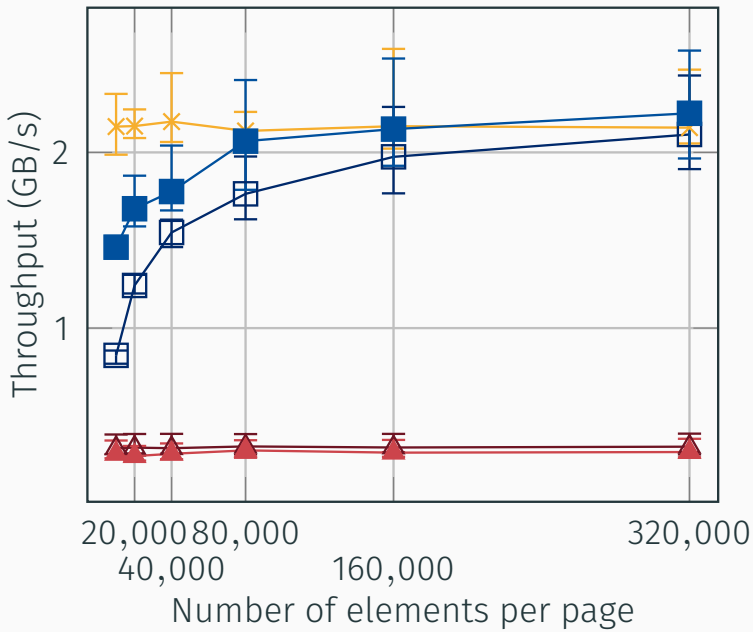


OID-per-page, increasing page size, constant cluster size

(a) `gen_lhcb`, no compression.

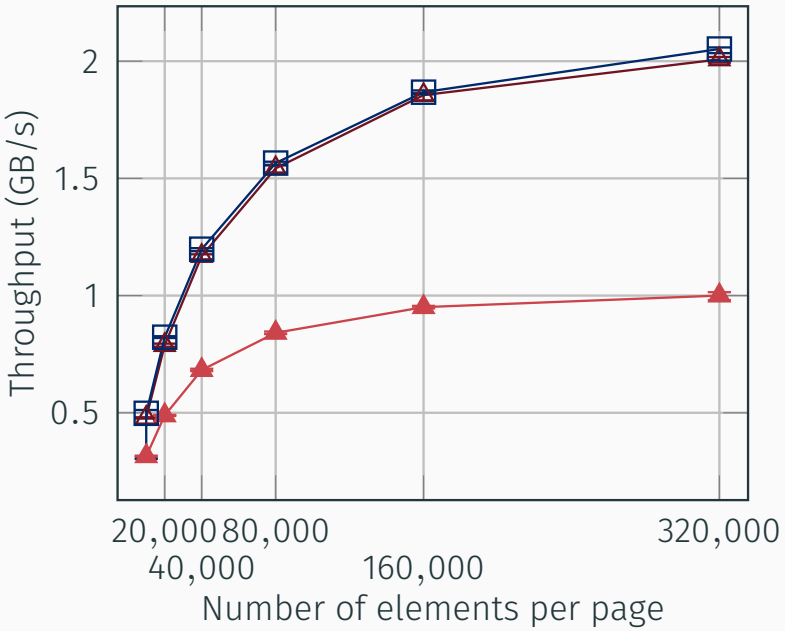


(b) `lhcb`, no compression.

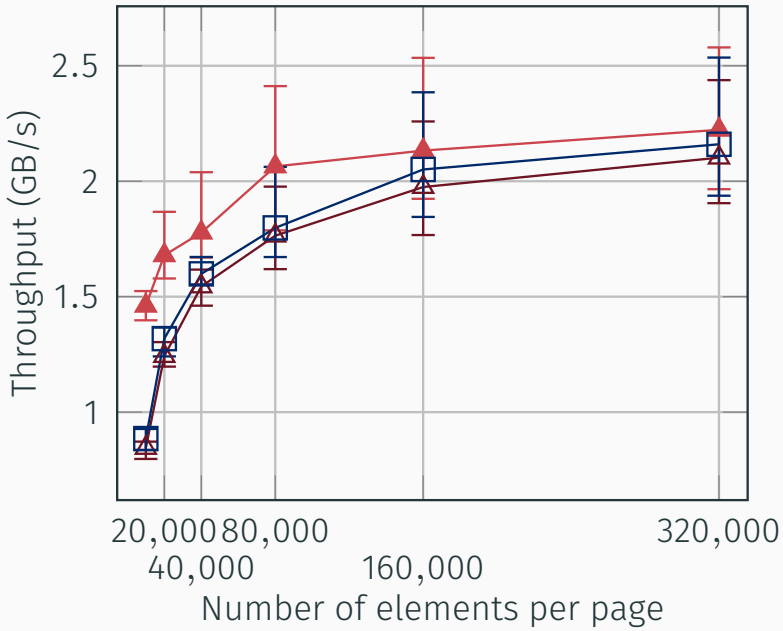


OID-per-page vs. OID-per-cluster

(a) `gen_lhcb`, no compression.



(b) `lhcb`, no compression.



Summary

- Issuing many small I/O requests has a negative impact on the read throughput.
- DAOS performs better with large page sizes, where it outperforms local SSDs.
- RNTuple native DAOS backend outperforms `dfuse` in all cases.

- IOR measured 12.3 GB/s: number of processes/threads seems to be a limiting factor.
- RNTuple-to-DAOS mapping seems to have a performance impact.

Conclusion

Conclusion

- We expect object stores to have an important role in next-generation data centers.
- RNTuple architecture decouples storage from serialization/representation. Object stores are first-class.
- First prototype implementation of an Intel DAOS backend merged into ROOT's '`master`' branch.
- Regular contact with DAOS development team; ongoing efforts to test the backend in other cluster platforms, e.g. HPE DAOS testbed.

Next Questions

1. Further investigate reads not saturating the data link.
2. WIP: Optimize moving large amounts of existing HEP data to a DAOS-based data center.
3. Implement and test Improved OID-per-cluster mapping.

RNTuple Checkpoint

RNTuple Status and Plans

In the last 6 months

- Merged DAOS backend
- Experimental support for S3
- CMSSW nanoAOD output module (Max' IRIS-HEP project)
- Support for (aligned) friends
- RBrowser support
- Type casting (e.g. read float in double)
- Parallel page compression during writing
- LHCC review input

Upcoming steps

- Merge version 1 binary format
- Schema evolution proof-of-concept
- Finish up chains and merging
- RVec support (important for RDF)
- Double32 support
- New HDF5 comparison benchmarks

RNTuple on DAOS

EP R&D Software, 7.4—Efficient Analysis

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

EP R&D Software Working Group meeting, 2021-07-14

EP-SFT, CERN, Geneva, Switzerland

http://root.cern/

Evaluation – Hardware and Software Environment

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 1: Server nodes

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 2: Client node

- DAOS 0.9.4, libfabric 1.7.2, libpsm2 11.2.78, and ROOT `1f1e9b8`.
- PSM2 transport (`ofi+psm2`); flow control disabled.