Exploring Object Stores for High-Energy Physics Data Storage

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

vCHEP2021, 2021-05-19

EP-SFT, CERN, Geneva, Switzerland

http://root.cern/

Contents

Introduction

Object Stores: Motivation

- Traditional storage stack designed for high-latency rotating disks that could handle few IOPS.
- I/O coalescing, buffering, etc. is less relevant for modern devices, such as NVMe SSDs.
- POSIX I/O consistency model is a major problem in parallel filesystem scalability.

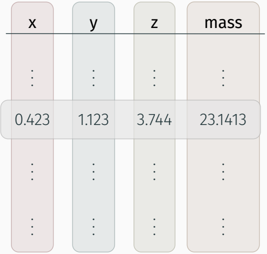
- Object stores, e.g. Intel DAOS, provide a fault-tolerant object store optimized for high bandwidth, low latency, and high IOPS.
- At least the GET and PUT primitives; objects accessed via a unique object identifier (OID).

Object Stores: Motivation

- Traditional storage stack designed for high-latency rotating disks that could handle few IOPS.
- I/O coalescing, buffering, etc. is less relevant for modern devices, such as NVMe SSDs.
- POSIX I/O consistency model is a major problem in parallel filesystem scalability.

- Object stores, e.g. Intel DAOS, provide a fault-tolerant object store optimized for high bandwidth, low latency, and high IOPS.
- At least the GET and PUT primitives; objects accessed via a unique object identifier (OID).

- Most analyses in HEP require access to many events, but only a subset of their properties: *columnar storage*.
- TTreehas been in use for 25 years. **1+ EB** of HEP data stored in ROOT files.
- However, it was not designed to fully exploit modern hardware.
- RNTuple is the R&D project to replace TTree for the next 30 years.
- Object stores are first-class.



RNTuple: Architecture Overview

Event iteration
Looping over events for reading/writing

`RNTupleView, RNTupleReader/Writer`

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

`RField, RNTupleModel, REntry`

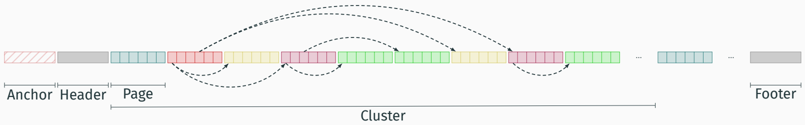
Primitives layer / simple types
"Columns" containing elements of fundamental types (`float`, `int`, ...) grouped into (compressed) pages and clusters

`RColumn, RPage, …`

Storage layer / byte ranges
POSIX files, object stores, ...

`RPageStorage, RCluster, …`

RNTuple: On-disk File Format




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

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

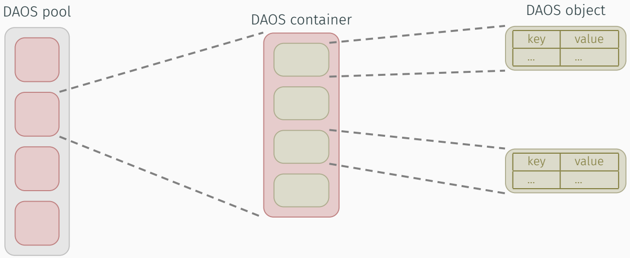
Header: schema information.[2]

Footer: location of pages and clusters.[2]

Pages: little-endian fundamental types (possibly packed, e.g. bit-fields) $\sim$ tens of KiB.[2]

[2] This element may be compressed or not.

Intel DAOS: Pools, Containers and Objects



- **Object:** a Key–Value store with locality.
  - The key is split into **dkey** (distribution key) and **akey** (attribute key). $dkey_i \mapsto target_k$.
- **Object class:** determines redundancy (replication/erasure code).

The RNTuple DAOS backend

Overview


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

Mapping RNTuple Clusters and Pages to Objects

Two possible mappings for pages and clusters:

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

One OID per cluster. OID = cluster index; *dkey* is used for addressing individual pages in the cluster; constant *akey*

User Interface: RNTuple/file vs. RNTuple/DAOS

Changes required to the user code...[1]...

```
auto ntuple = RNTupleReader::Open("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");
```

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

User Interface: RNTuple/file vs. RNTuple/DAOS

Changes required to the user code...[1]...

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

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

[1] Issue: UUIDs are not meaninful to users (common problem in object stores).

Evaluation

Hardware and Software Environment

Experiments ran on the CERN Openlab DAOS testbed:

- 3 DAOS servers, 1 head node
- interconnected by an Omni-Path Edge 100 Series 24-port switch.

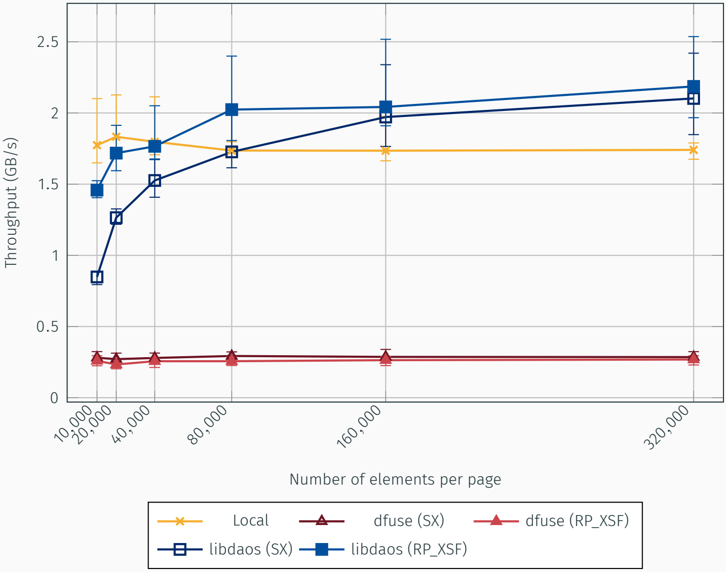
System specifications	
CPU	Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz
CPU per node	24 cores/socket, 2 sockets, 2 threads/core (HT enabled)
Core frequency	Base: 1.0 GHz Range: 1.0GHz - 3.9GHz
Numa nodes	node0: 0-23,48-71 node1: 24-47,72-95
System Memory	12x 32GB DDR4 rank DIMMs
Optane DCPMM	12x 128GB 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 HW

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 HW

Performance Analysis: fixed cluster size, increasing page size



Conclusion

Conclusion

- 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.

Next Questions:

1. Investigate why reads are not saturating the data link.
2. Data movement: how to quickly move large amounts of data from HEP storage to a DAOS data center?

Exploring Object Stores for High-Energy Physics Data Storage

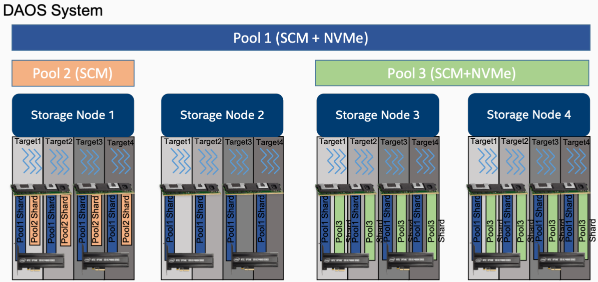
Javier López-Gómez – **CERN**
<javier.lopez.gomez@cern.ch>

vCHEP2021, 2021-05-19

EP-SFT, CERN, Geneva, Switzerland

http://root.cern/

BACKUP – DAOS: Overview



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

Storage resources are partitioned into targets that can be accessed independently (avoids contention).

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

Pool: storage partition that may expand over many servers (and is distributed among the available targets). Identified by and UUID.

BACKUP – DAOS: compatibility layer

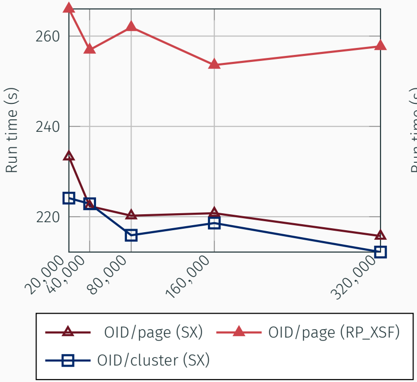
Existing software can use DAOS[2,3] through:

- **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, ...**

[2] https://daos-stack.github.io/
[3] https://github.com/daos-stack/daos/

BACKUP – Comparing OID-per-page to OID-per-cluster

(a) `gen_lhcb`, no compression.



(b) `lhcb`, no compression.