Object Stores for CMS data

**Nick Smith**, Bo Jayatilaka, David Mason, Oliver Gutsche, Alison Peisker, Robert Illingworth, Chris Jones (FNAL)
CHEP 2023
9 May 2023
Our inference pipeline

30k CPU-y → 30k CPU-y → 30k CPU-y → 3k CPU-y

RAW 100 PB

Centrally managed 100 TB-10 PB

Order of magnitude for CMS Run-II (2016-18)

Processing time (~100B events)

Data volume on disk
Centrally managed data

Primary dataset

Abstract, “what kind of events.”

e.g. hard scatter process for simulation, trigger filter for data

Data tiers

AOD

Data columns pertaining to low-level reconstruction

- .root
- 1e9/file

MiniAOD

Calibrated physics objects
Pathicle-flow candidates

- .root
- 1e9/file

...
Primary dataset

Abstract, “what kind of events.”
e.g. hard scatter process for simulation, trigger filter for data

AOD
Data columns pertaining to low-level reconstruction
- Data tiers
  - .root
  - 1e9/file

MiniAOD
Calibrated physics objects
Particle-flow candidates
- Data tiers
  - .root
  - 1e9/file

Data volume
order of magnitude [bytes]
Primary dataset

Abstract, "what kind of events.

e.g. hard scatter process for simulation, trigger

AOD

Data columns pertaining to
low-level reconstruction

MiniAOD

Calibrated physics objects

Particle-flow candidates

Root file...root

1e9/file

1e5/event

1e4/event

Data tiers

Data volume

order of magnitude [bytes]

Centrally managed data

Accessed

Not accessed
Primary dataset

Abstract, “what kind of events.”

e.g. hard scatter process, trigger filter for data

AOD

Data columns pertaining to low-level reconstruction

MiniAOD

Calibrated physics objects

Particle-flow candidates

Data tiers

Data volume order of magnitude [bytes]

Centrally managed data

Accessed

Not accessed

Copied forward each data tier
Projected usage

https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults
CMS-NOTE-2022-008
Projected usage

CMSPublic
Total Disk HL-LHC (2031/No R&D improvements) fractions
2022 Estimates

CMSPublic
Total Tape usage HL-LHC (2031/No R&D improvements) fractions
2022 Estimates

900 PB

2400 PB

https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults
CMS-NOTE-2022-008
**Strawman**

- What if we stored batches of events for each data product individually?
  - No more merge jobs!
- Most content does not change with re-processing
  - Even for UltraLegacy, already two MiniAOD versions
  - Keeping only new products would save a lot of disk

<table>
<thead>
<tr>
<th>Data-tier scheme</th>
<th>Column scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiniAOD Data product</td>
<td><strong>KB per event</strong></td>
</tr>
<tr>
<td></td>
<td>v1</td>
</tr>
<tr>
<td>packed+pruned genParticles</td>
<td>5.7</td>
</tr>
<tr>
<td>slimmedElectrons</td>
<td>1.3</td>
</tr>
<tr>
<td>Others</td>
<td>48.7</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>55.7</strong></td>
</tr>
</tbody>
</table>

Numbers sourced from a CMS UL17 TTBar simulation file
Object store vs. filesystem

- Traditional data storage technology: distributed filesystem
  - e.g. NFS, EOS, dCache, Lustre, HDFS*, ...
  - Often with remote access protocol (xrootd)
  - Files are concurrently read/writeable

- Popular new-ish technology: object store
  - Native remote access (http)
  - Objects are immutable (overwrite possible)
Breaking down the ROOT file

• Essentially storing (+ moving) smaller units
  - This is usually a bad thing
Breaking down the ROOT file

• Essentially storing (+ moving) smaller units
  - This is usually a bad thing
• Calculated placement
  - Like a hash, client-side
  - Downside: cluster state change causes reshuffle
    • Consistent hashing to minimize movement
Object data format

Infrequently used column stripes could be concatenated into a file and offloaded to tape systems.
Test cluster

- Ceph pilot cluster setup at FNAL
  - 9 retired dCache machines
  - Total 2 PB HDD, circa 2014-2018
  - 288 OSDs
- Two servers for metadata
  - 20TB NVMe (32 OSDs)

- Edge machines for:
  - xrootd door to CephFS
  - Ceph management daemons
  - RadosGW
    - Implements S3 protocol
    - Auth: pre-shared key or OIDC token
- Obviously not production-grade
  - Good for us: experience with failures!
Client design

- Framework to evaluate alternative I/O strategies ([github](https://github.com)): Mimics CMS event processor design: TBB thread pool + tasks
  - Easy to add new output modules, simulate event processing, and test I/O
  - Serialization of data products: ROOT TBufferFile

- Developed S3 source and output module in framework
  - Using [libs3](https://github.com) + libcurl for protocol, async event loop separate from thread pool
  - Key features:
    - Parallel stream compression
    - Asynchronous I/O
    - Row-wise to column-wise pivot

- In following slides: stress testing the RadosGW server
  - Using many clients in parallel
Storage efficiency

- Input: 80k event MiniAOD file
  - LZMA compression
- Various S3 output configurations tested
- For erasure-coded Ceph pools, minimum object granularity of k*4kiB
  - Implies wasted space (vs. overhead for data resiliency)
  - Wasted space for EC4+2 in % listed below

<table>
<thead>
<tr>
<th>Format</th>
<th>KB per event</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiniAOD input</td>
<td>55.7</td>
</tr>
<tr>
<td>Objects:</td>
<td></td>
</tr>
<tr>
<td>- event batch size 720</td>
<td></td>
</tr>
<tr>
<td>- target stripe size 128kiB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>71.4 + 6.5%</td>
</tr>
<tr>
<td>Objects:</td>
<td></td>
</tr>
<tr>
<td>- event batch size 720</td>
<td></td>
</tr>
<tr>
<td>- target stripe size 512kiB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>70.6 + 3.5%</td>
</tr>
<tr>
<td>Objects (LZMA):</td>
<td></td>
</tr>
<tr>
<td>- event batch size 720</td>
<td></td>
</tr>
<tr>
<td>- target stripe size 512kiB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>61.8 + 3.7%</td>
</tr>
<tr>
<td>Objects (+product groups):</td>
<td></td>
</tr>
<tr>
<td>- event batch size 720</td>
<td></td>
</tr>
<tr>
<td>- target stripe size 512kiB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>70.6 + 1.4%</td>
</tr>
</tbody>
</table>
Write-only stress test

- Submit 1-core condor jobs
  - Read MiniAOD from FNAL dCache, write to S3
  - Handling up to 500 PUT/s
    - Past experience: can do ~1500 for smaller objects
  - Wrote 4.5 TB, 7.4 Mobj total
- Saturation at ~400 clients, ~400 MB/s*
Read-only stress test

- Submit 4-core condor jobs
  - Read from S3, decompress, deserialize
- Unable to reach saturation
  - Poor condor queue priority
  - Performance in line with single-machine scaling
    • (As shown at ACAT22)
Client performance considerations

- Client application CPU inefficiency driven by I/O latency: either waiting for inputs or to flush output

- By pre-fetching input stripes and using a “fire-and-forget” output technique, CPU efficiency improves substantially

- When server is saturated, client CPU efficiency degrades significantly
Next steps

- Demonstrate use case: job 2 reads job 1 and input products concurrently
  - Best example of advantage for column-level storage?
Summary

- **Object data formats** provide new data management capabilities
  - Compared to current tier-based EDM file model
  - Reduce disk storage requirements for re-processing
  - Obviate the need to define data tiers
- In a **prototype framework** accessing a Ceph S3 service
  - On-disk data and metadata volume is as expected
  - Service scaling is promising: one RadosGW can serve ~400 client threads
- To fully utilize, more software development will be needed
Backup
**S3Outputer design**

Each box is a TBB task
Color = task group

Product 1 has stripes written every 4 events

Product 2 has stripes written every 2 events

Label convention:
\( E^* \) = Event number
\( P^* \) = Product (column) index
\( i^* \) = Global index
\( S^* \) = Stripe starting at global index
S3Source design

Each box is a TBB task
Color = task group

Product 1 has stripes
read every 4 events

Product 2 has stripes
read every 2 events

Label convention:
E* = Event number
P* = Product (column) index
i* = Global index
S* = Stripe starting at global index
S3 vs. other Source/Outputers

- **ROOT source similar to a CMSSW grid job**
  - Read file via xrootd, server has CephFS (same cluster) mounted
  - No ROOT outputer due to bug
- **PDS source: write whole events sequentially**
  - Very good thread scaling (last data point = all cores on machine)
  - Writing to local file rather than remote server
Bandwidth inconsistencies

Ceph cluster prometheus metrics under-report the RadosGW bandwidth compared to server IP traffic & client aggregate bandwidth (recorded by application)

Server reported

Ceph reported

Clients reported