Efficient Analysis Facilities

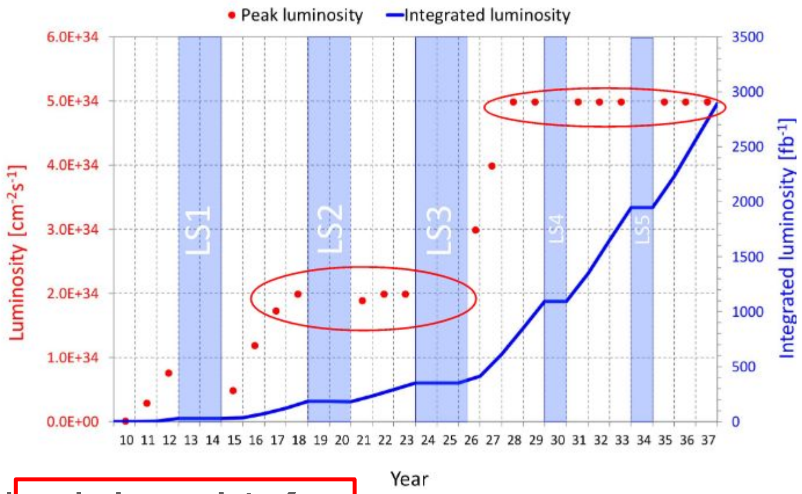
EP R&D - Software WP

Jakob Blomer

2020-04-08

Analysis Challenge

- HL-LHC challenge: first major milestone on the way towards future accelerators and detectors
    - From 300fb^{-1} in run 1-3 to 3000fb^{-1} in run 4-6
    - 10B events/year to 100B events/year
    - Real analysis challenge depends on several factors: number of events, analysis complexity, number of reruns, etc.
        - **As a starting point, let's prepare for ten times the current demand**
- Developments in our favour
    - Experiment R&D on central, compact AODs, e.g. CMS nanoAOD, ATLAS DAOD_PHYSLITE 1kB - 10kB per event
    - R&D on ROOT I/O throughput
        - Currently 100kB - 10MB/s per core
        - In the lab: 100MB/s per core
            - Google: 200MB/s per core
- Faster storage devices: SSD, NV-RAM (~10x faster)
    - Too expensive for the grid, but not for HPCs and dedicated analysis facilities
- **Calls for major R&D and engineering on I/O subsystem and analysis user interface**

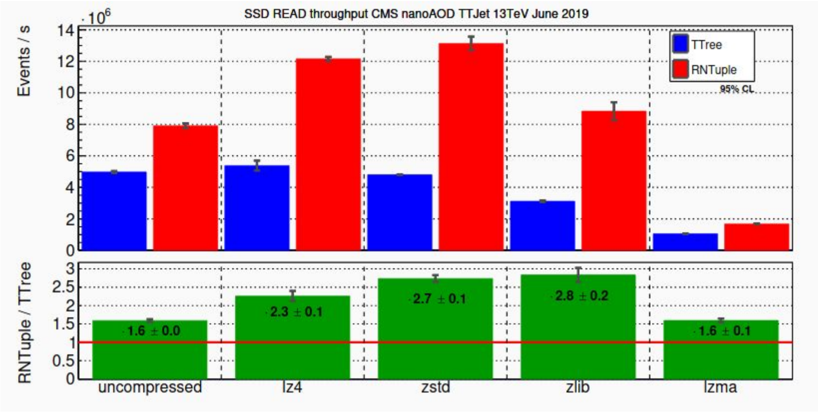


Increase data throughput

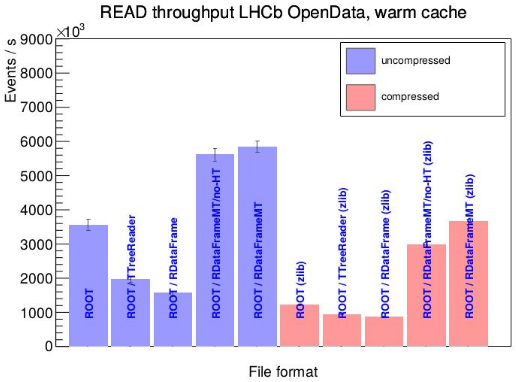
Increase analyst's throughput

R&D Programme on Efficient Analysis Facilities

1) Increase data throughput
   - Data format R&D
   - ROOT RNTuple is a research prototype for next-generation event I/O
   - Expect ~25% smaller files,
     x2-5 better single-core throughput on SSD
   - Promising first results, full exploitation subject of the R&D programme



2) User interface R&D
   - RDataFrame introduced as ROOT's declarative analysis toolkit
   - Two major R&D challenges
     i) Optimal translation to low-level I/O routines
     ii) Distributed execution engine: how to run the analysis on my laptop on O(1000) cores



Non-exhaustive list of R&D topics

- Direct access to object stores, e.g. Intel DAOS
- Integration of data decompression tasks into (experiment's) MT frameworks
  - Big speed-up on laptops: all cores decompress
- Optimal data pipeline from RNTuple to numpy
  - Otherwise we risk wasting enormous resources on data transformation
- Lossy data compression: automatic choice of floating point precision (lead by FNAL and BNL)
- Automatically detect storage class for optimal access method (particularly difficult for remote I/O)
- Active exploitation of NV-RAM, e.g. for dynamically created index branches
- Understanding of throughput results in the lab vs. throughput results in production clusters
- Exploiting bulk I/O and vectorization with RDataFrame
- Application-assisted caching in analysis facilities (e.g. certain hot branches)
- Explore expressiveness of RDataFrame: which analysis building blocks can be described (e.g. model building, data set management)

Background task: engineering work on RNTuple and RDF
E.g. visualization (RBrowser etc), RNTuple meta-data management, RDF connector to cluster manager, format conversion utilities, testing I/O error cases and many more

Resources in 2020

- Vincenzo Padulano (PhD student)
  - Started in February, supervised by Enric Tejedor
  - Investigation on distributed data caching for analyses expressed in RDataFrame
    - pyRDF and Spark as technology basis

- New fellow
  - To be selected for Q2/2020, supervised by me
  - Expected milestone: prototype integration of RNTuple with Intel DAOS
    - Object store system for HPCs
    - ~230PB planned for Argonne HPC from 2021

- R&D hardware: Fast I/O development machine
  - To be purchased for Q2/2020
  - Access to latest SSDs and NV-RAM devices
  - For the time being, we use temporary hardware made available by openlab

RNTuple Integration with Object Stores: API

Event iteration
Reading and writing in event loops and through `RDataFrame`
`RNTupleDataSource, RNTupleView, RNTupleReader/Writer`

Logical layer / C++ objects
Mapping of C++ types onto columns
e.g. `std::vector<float>` ↦ 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, RColumnElement, RPage`

Storage layer / byte ranges
`RPageStorage, RCluster, RNTupleDescriptor`

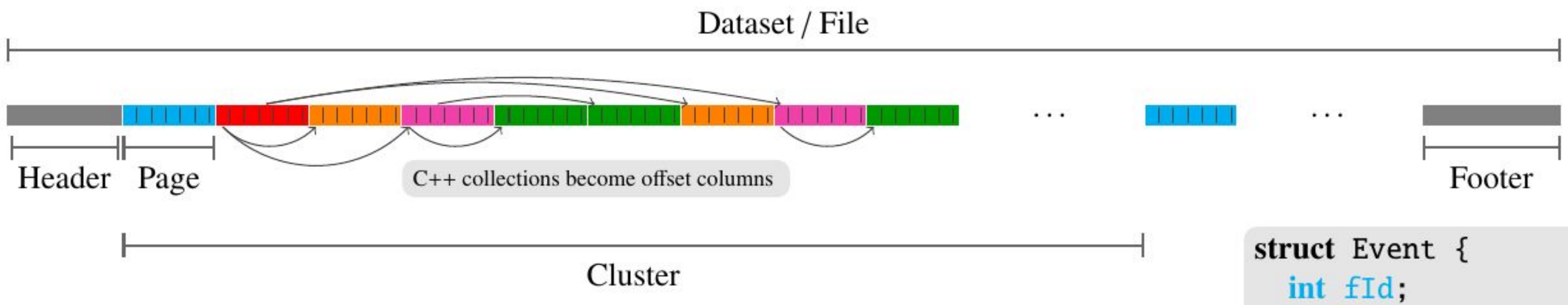
Modular storage layer that support files as data containers but also file-less systems (object stores)

Currently in touch with Intel DAOS engineers on RNTuple integration

Approximate translation between TTree and RNTuple classes:

TTree	≈	RNTupleReader RNTupleWriter
TTreeReader	≈	RNTupleView
TBranch	≈	RField
TBasket	≈	RPage
TTreeCache	≈	RClusterPool

RNTuple Integration with Object Stores: Data Layout



Approximate translation between TTree and RNTuple concepts:

Basket	≈	Page
Leaf	≈	Column
Cluster	≈	Cluster

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

Cluster:

- Block of consecutive complete events
- Unit of thread parallelization (read & write)
- Typically tens of megabytes

Page:

- Unit of memory mapping or (de)compression
- Typically tens of kilobytes
- Naturally representable by an object, e.g. in the DAOS object store (under investigation)

Distributed data caching in an RDataFrame analysis

Cache input data that is being read during a distributed RDF analysis to the most granular degree possible

Cache only what it's actually read during the analysis:

- Cache only the processed branches
- Cache only the TTree clusters read by worker tasks

Goal: Speedup of a physics analysis, that is repeatedly run on the same data with slightly different parameters each round.

How to cache TTree data?

Currently using PyRDF + Spark backend to create a set of tests for comparing different tools/ways in which we could cache data:

1. TFilePrefetch: stores in a file the TBuffers that are read.
2. XRootD: ProxyPlugin feature for orchestrating a cache.
3. RDataFrame: Snapshot in parallel to the analysis.
4. Something more...?

Granularity	Branch, Cluster
Granularity	File
Granularity	Not present

First Steps: TFilePrefetch

Toy example:

- Small Spark cluster (CERN OpenStack VMs): 1 master and 3 workers.
- Simple analysis run with PyRDF

Input file from EOS is cached on one worker.

Issues:

- Only one worker caches if running analysis on multiple workers
- Doesn’t cache data ranges outside of the first cluster

Possible bug in TFilePrefetch under investigation.

Github repo