RNTuple Performance: Status and Outlook

Javier Lopez-Gomez ⋆ EP–SFT, CERN

Jakob Blomer ⋆⋆ EP–SFT, CERN

EP-R&D Programme on Technologies for Future Experiments

ROOT Data Analysis Framework

Storage of HENP Data: Outlook

Upcoming HENP experiments are expected to increase the volume of generated data by a factor of **10**:

- I/O performance is critical for many HENP workflows.
- Technology landscape changed since TTree: low-latency high-bandwidth NVMe devices, distributed object stores, concurrency, development best practices...

Learning from TTree and Apache Arrow/Parquet, RNTuple comes with new API and data format (Fig. 1) to improve: performance, robustness, reliability, error handling, type safety, and concurrent/asynchronous behavior.

In this contribution, we compare the throughput delivered by RNTuple to popular I/O libraries outside HENP: HDF5 and Apache Parquet.

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

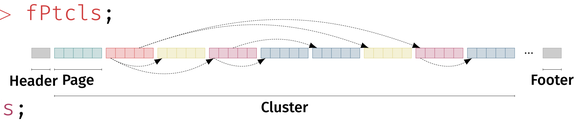


Fig. 1: RNTuple on-disk format; pages store column data

Evaluation Procedure

In order to make a fair comparison, all the test programs were coded in C++. Tunable values (e.g. row group and page sizes, compression level) were set to match in all the tests.

For Apache Parquet, we used the Parquet–Arrow API which allows convenient use of complex nested structs/lists.

In HDF5, however, columnar access of nested/heterogeneous data is not trivial. Parts of the code in our benchmarks [1] bridge this gap. We tested two data layouts:

- **Row-wise:** uses compound and variable-length HDF5 types.
- **Column-wise:** emulated columnar layout as described in [2]. This layout requires custom adjustments to read data event-by-event.

Performance Evaluation

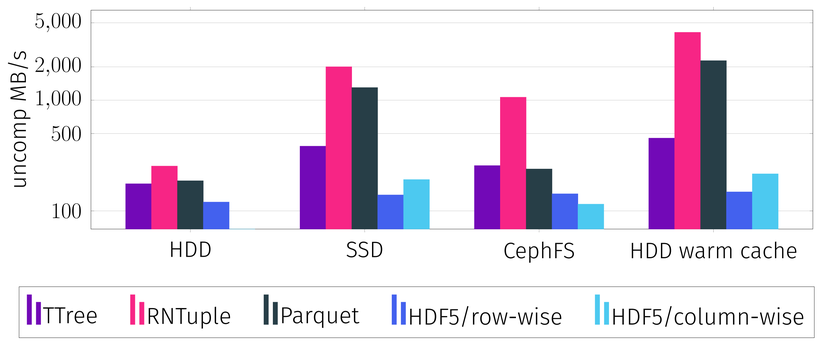
Datasets:

- **LHCb OpenData B2HHH:** 8.5 million events; no nested collections. 26 branches.
- **CMS OpenData Higgs→4 leptons:** 300k events [×16]; NanoAOD-like. 84 branches.

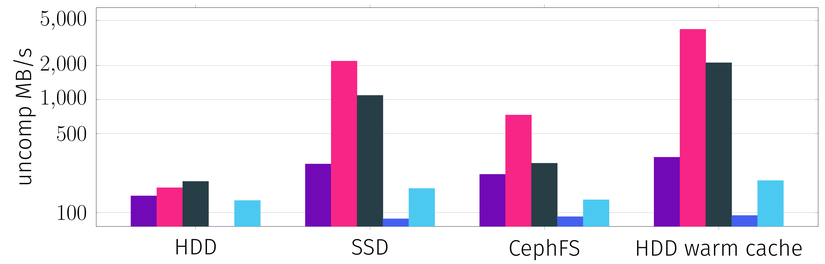
We measured:

- End-to-end analysis throughput (i.e. from storage to histogram), in uncompressed MB/s for LHCb.
- Read rate (uncompressed MB/s) for LHCb and CMS.
- Using zstd compression (gzip for HDF5).

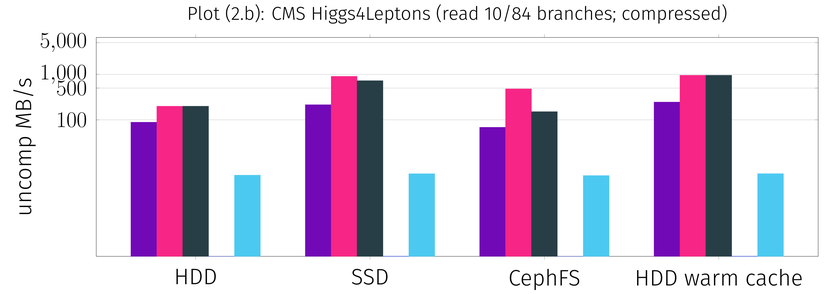
Plot (1): LHCb B2HHH analysis throughput (18/26 branches; compressed)



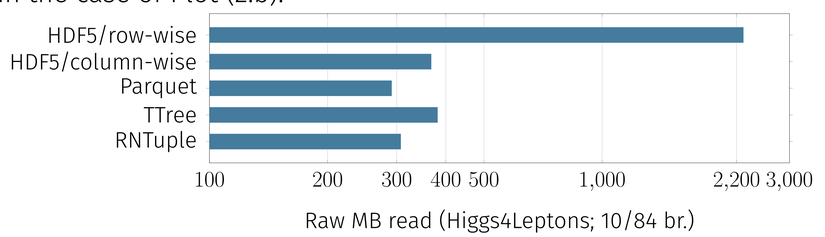
Plot (2.a): LHCb B2HHH (read 10/26 branches; compressed)



Plot (2.b): CMS Higgs4Leptons (read 10/84 branches; compressed)



Also, we used `vmtouch` [3] to measure the number of raw bytes read in the case of Plot (2.b):



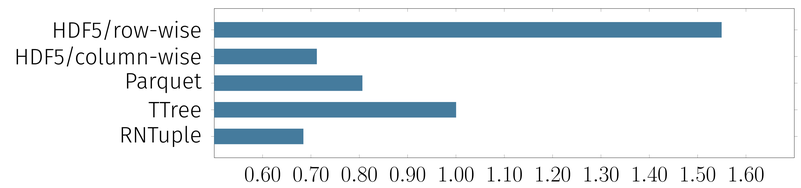
Raw MB read (Higgs4Leptons; 10/84 br.)

Qualitative Evaluation

For each candidate, we compared the support level of different features that are considered important in a HENP I/O layer.

	HDF5	Parquet	TTree	RNTuple
Transparent compression	(1)	●	●	●
Columnar access	(2)	●	●	●
Merging without uncompressing data			●	○
Vertical/horizontal data combinations		?	●	○
C++ and Python support	/	●	●	●
Support for structs/nested collections	?	●	●	●
Architecture-independent encoding	●	●	●	○
Schema evolution			●	○
Support for application-defined metadata	●			○
Fully checksummed	●	●		○
Multi-threading friendly	○	●	●	●
Native object-store support	●	●		●
XRootD support			●	●

● Supported ○ Planned / Under development
/ Partial / Incomplete ? Unclear
(1) Only for chunked datasets (2) Via emulated columnar



Relative increase in file size w.r.t. TTree (Higgs4Leptons; compressed)

Future Work

Our results show the benefits of using RNTuple in HENP workflows. RNTuple delivers improved performance thanks to its **performance-tuned implementation**, and **parallel I/O scheduling and decompression**.

This evaluation will be extended by:

- Comparing object store performance for RNTuple and HDF5
- Adding more benchmark samples

RNTuple is scheduled to become production grade in 2024. We appreciate the first experiments implementing RNTuple writers in their workflows, providing feedback on features and performance.

References

[1] https://github.com/jalopezg-r00t/iotools/tree/ACAT21/compare
[2] https://inspirehep.net/files/e048b0cd122919dc9a009793983a81e0
[3] https://github.com/hoytech/vmtouch/

⋆ <javier.lopez.gomez@cern.ch> ⋆⋆ <jblomer@cern.ch>