RNTuple Workshop 2024: State of Affairs

Jakob Blomer for the ROOT team

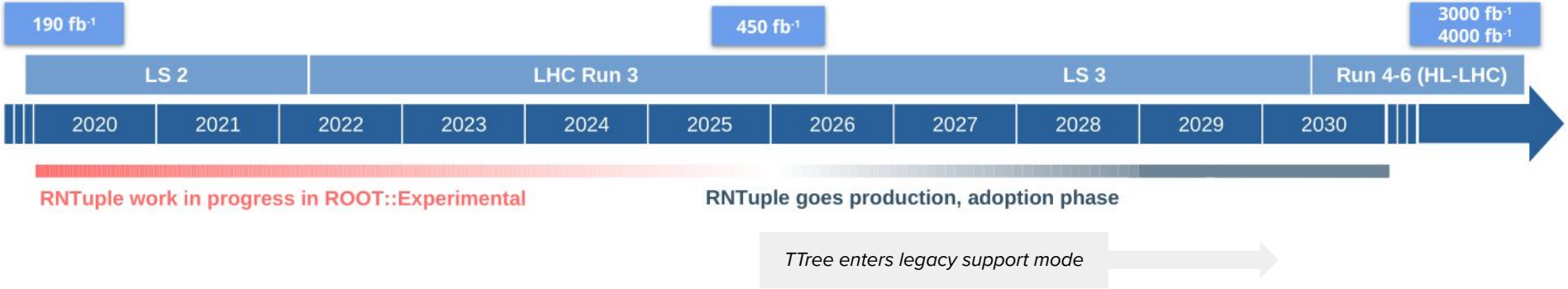
2024-12-02

ROOT

Data Analysis Framework

https://root.cern

Context: ROOT I/O Upgrade for HL-LHC



>2EB (now) → >10EB (end of HL-LHC)
~½ of the currently projected WLCG budget on storage

Major I/O upgrade of the event data file format and access API: TTree ➜ RNTuple

RNTuple Main Results

- **Major I/O upgrade** of the event data file format and access API: **TTree → RNTuple**
  - Less disk and CPU usage for same data content
    - **10-50% smaller** files, **better single-core performance often by factors**
  - Give access to **novel and future storage technologies**
    - Native support for HPC and cloud object stores
    - Async and parallel I/O: fully exploits modern NVMe drives
    - Design prepared for accelerators (e.g., GPUs, compression offloading)
  - Systematic use of checksumming and exceptions to prevent silent I/O errors

- Initial support in **ATLAS, CMS, LHCb software frameworks** (ESD, AOD, derived AODs & ntuples)

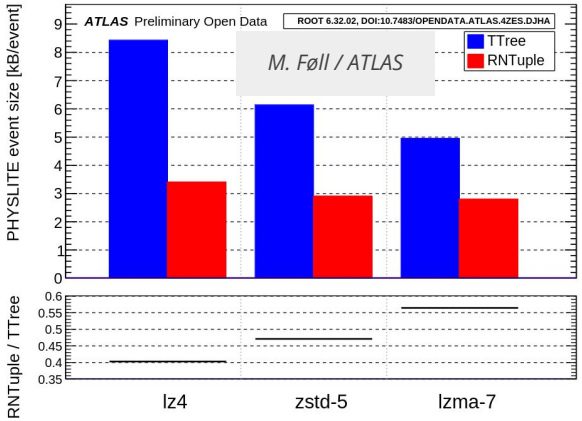
- Large-scale testing with IT storage group
  - 70 nodes, 100GbE EOS connection, 100TB inflated AGC benchmark

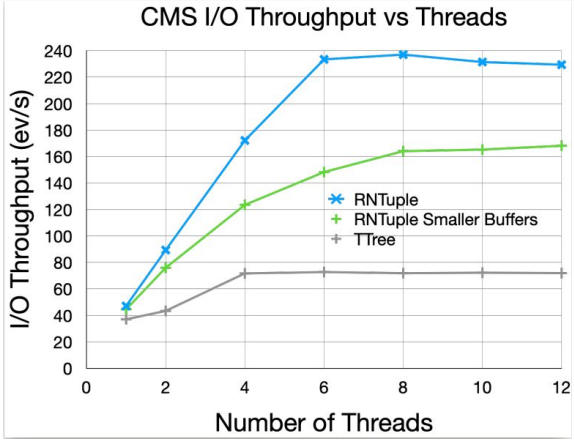
- ROOT 6.34 (Nov 2024): RNTuple stable on-disk format (version 1.0) released
  - Future ROOT versions will read data written with 6.34
  - Planned optional and possibly forward-compatibility breaking changes foreseen

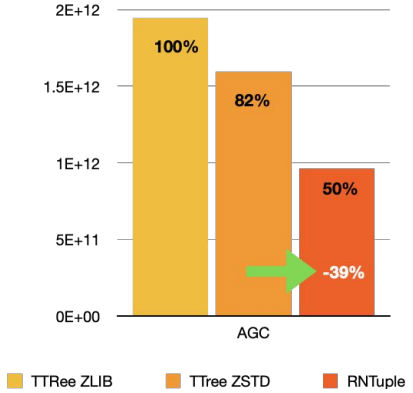
- ROOT v6.36 (planned for Q2/2025): first set of APIs move out of ROOT::Experimental
  - Taking into account the input received by the HEP-CCE review

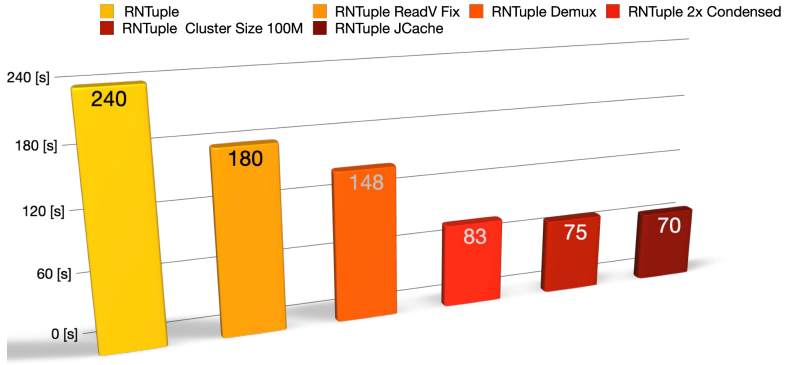
Many results presented at CHEP'24

Highlights from CHEP







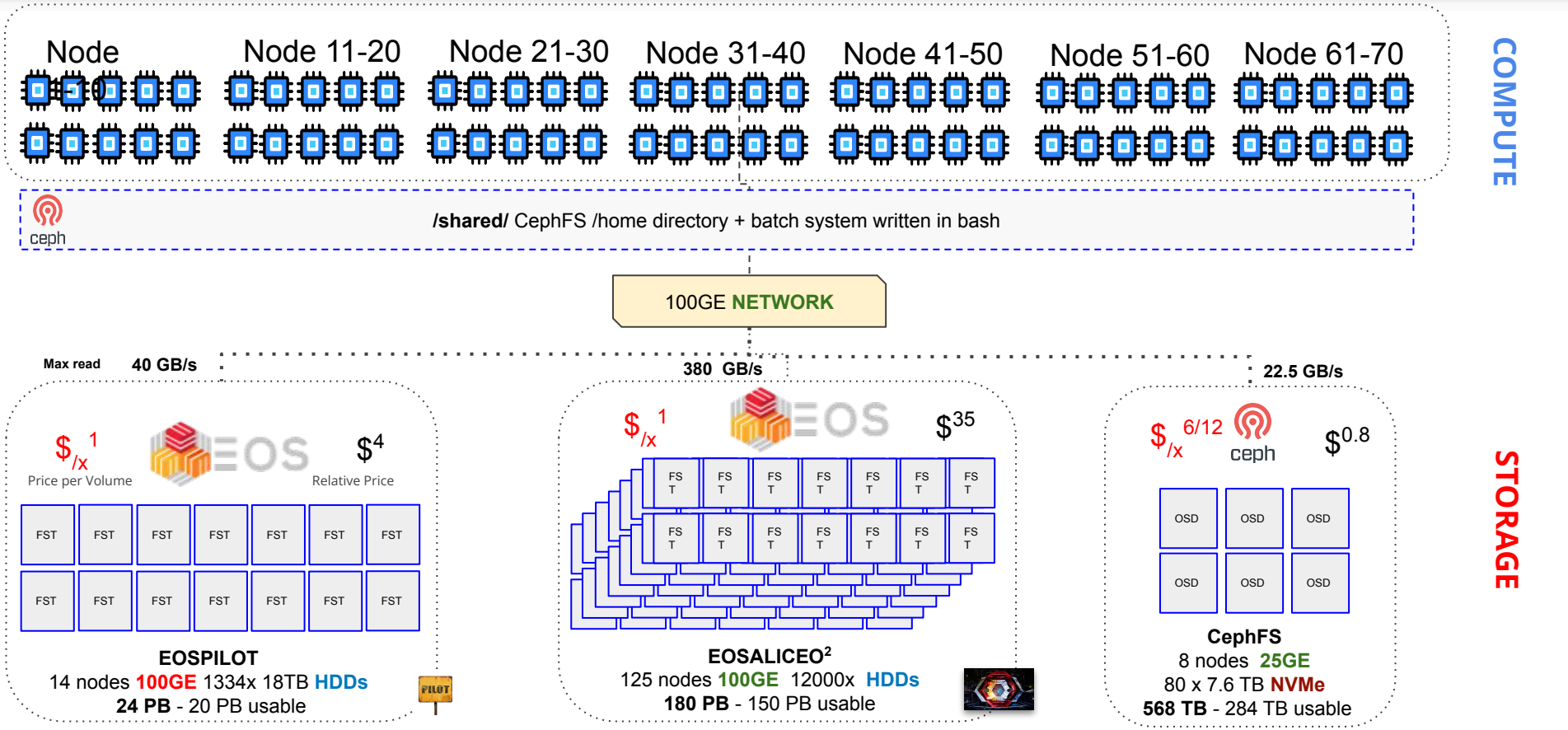


Progress since last year: type system

Type Class	Types	EDM Coverage			RNTuple Status
PoD	`bool, char, std::byte, (u)int[8,16,32,64]_t, float, double`	Flat n-tuple	Reduced AOD	Full AOD / ESD / RECO	Available
Records	Manually built structs of PoDs				
(Nested) vectors	`std::vector, RVec, std::array,` C-style fixed-size arrays				Available
String	`std::string`				Available
User-defined classes	Non-cyclic classes with dictionaries				Available
User-defined enums	Scoped / unscoped enums with dictionaries				Available
User-defined collections	Non-associative collection proxy				Available
stdlib types	`std::pair, std::tuple, std::bitset, std::(unordered_)(multi)set, std::(unordered_)(multi)map`				Available
Alternating types	`std::variant, std::unique_ptr, std::optional`				Available
Streamer I/O	All ROOT streamable objects (stored as byte array)				Available
Low-precision floating points	`Double32_t, f16`	*Optimization benefitting all EDMs*			Available
	Custom precision / range (bfloat16, TensorFloat-32, other AI formats)				Available

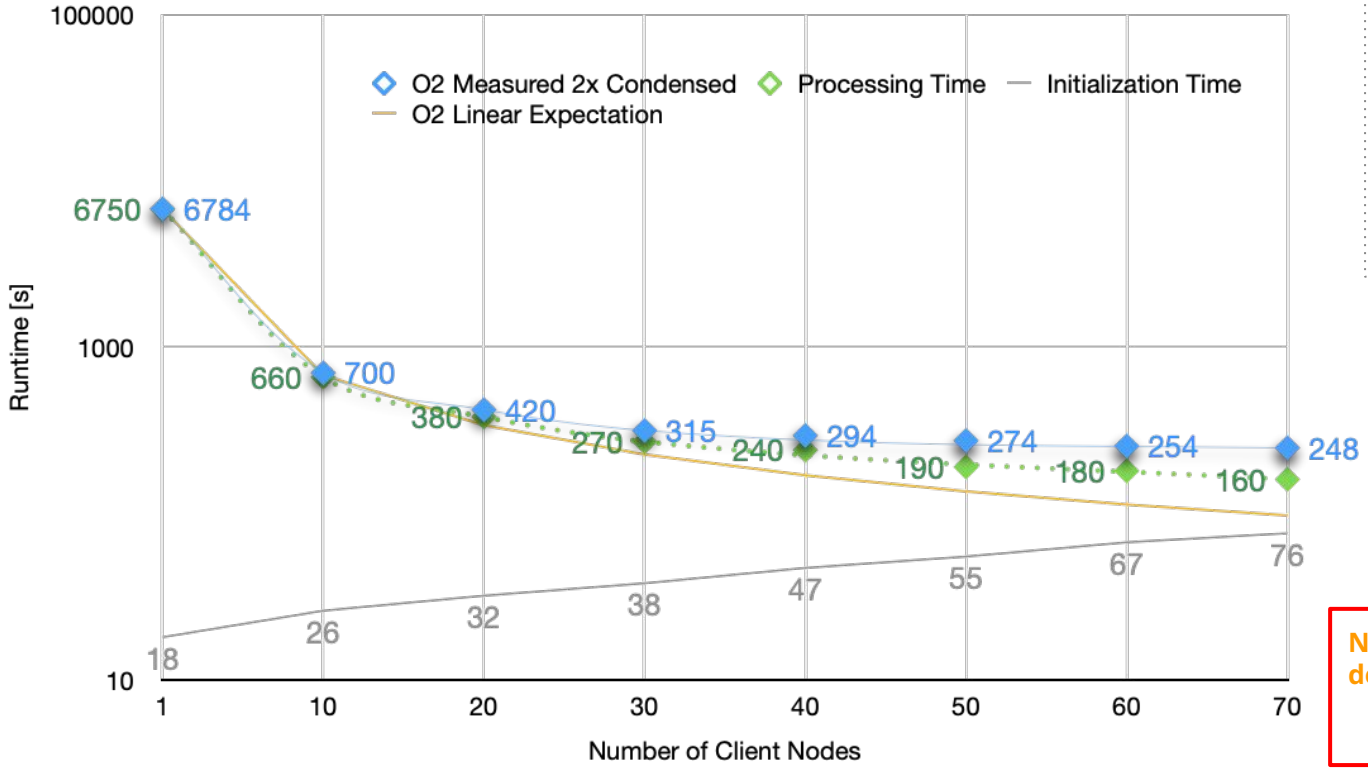
Type cast of PoDs

Progress since last year: AGC Testing I



Progress since last year: AGC Testing II

- Introducing modified RNTuple format for **AGC^{200}** with **EOSALICEO^{2}**



With a 100x inflated **AGC^{200}** dataset we observe that as the number of client nodes increases, the initialization time gets close the processing time, resulting in a breakdown of scalability.

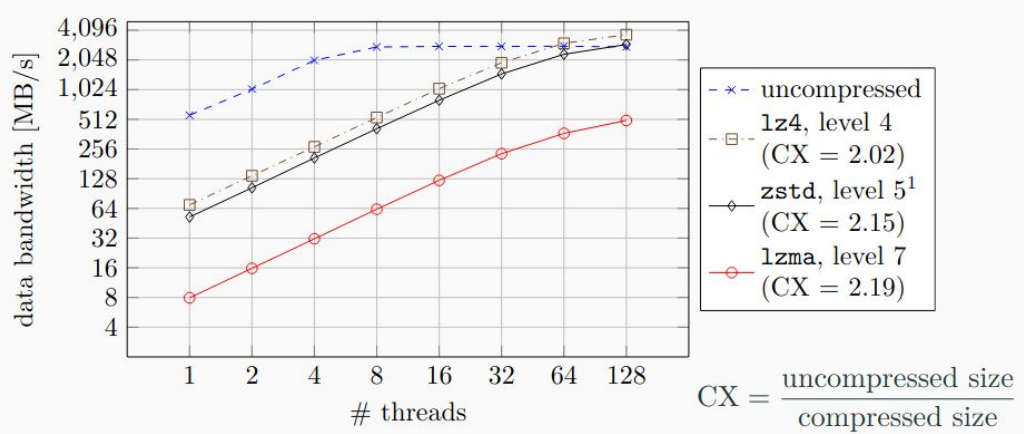
Single Analysis
extremely sparse
reaches avg. INGRES
222 GBit/s
during processing
345 GBit/s

Next step: Reconstruction and/or data derivation benchmark(s)

- → Dense reading and (parallel) writing

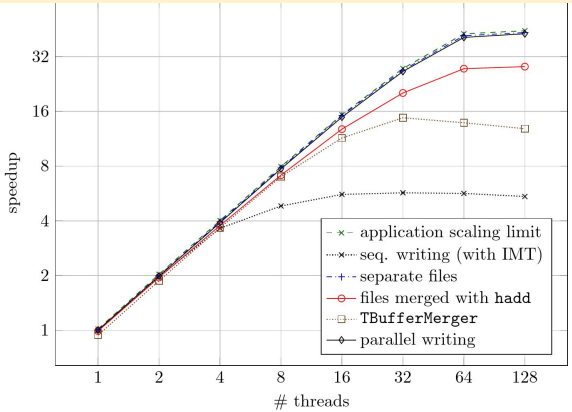
Progress since last year: parallel writing & direct I/O

Reconsider trade-off between write speed and file size



→ https://indico.cern.ch/event/1338689/contributions/6010002

Writing in parallel to one file is as fast as writing 128 files



→ https://arxiv.org/abs/2410.14239

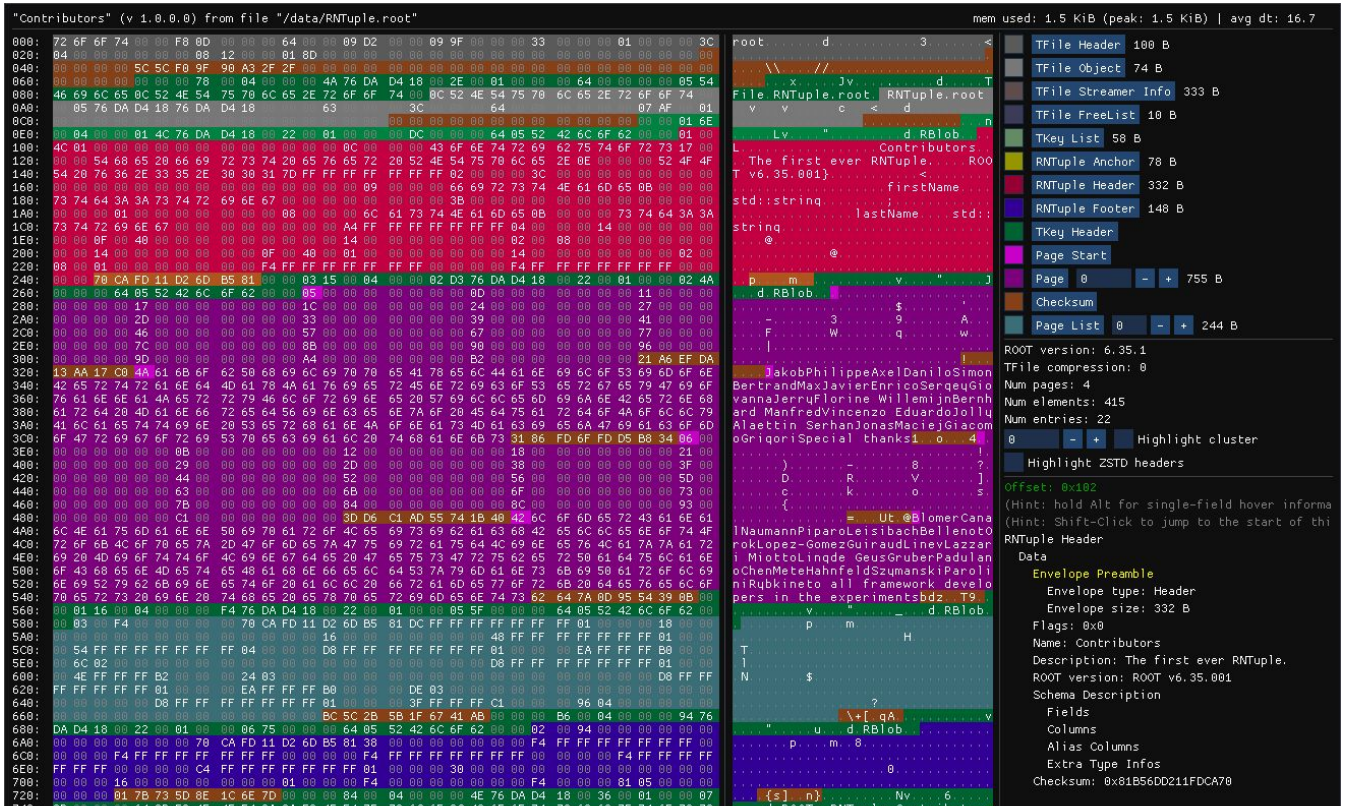
- Truly parallel writing; prototype support for multi-process and MPI support
- Capable of fully exploiting NVMe drives
- Reaching throughput values that allow for meaningful contribution to processing workflow of DUNE supernova event candidates

Progress since last year

- RNTupleProcessor: friends & chains with solid underpinnings
  - https://indico.cern.ch/event/1338689/contributions/6016196
  - See talk by Florine later today: https://indico.cern.ch/event/1468611/#3-rntuple-processor-joins
- Connect RNTuple type description to TFile streamer info (enabling, e.g., MakeProject and manual schema evolution)
- Late model extension in RNTupleMerger (TFileMerger)
- Removal of 1GB TFile limit for RNTuple data (exception: streamer field)
- Tested limits: 100k columns, 100k clusters, 600M elements per page
  - Some factor of 10 larger than largest examples we encounter today (e.g., ~15k columns in CMS AOD)

Tooling: RNTupleViewer

Rich (internal) tooling: RNTupleInspector (presented last year), RNTupleViewer: https://codeberg.org/silverweed/rntviewer

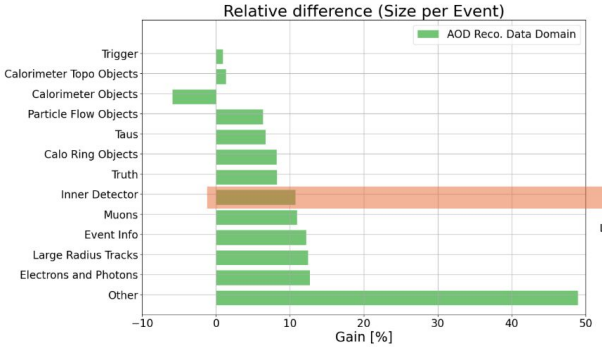


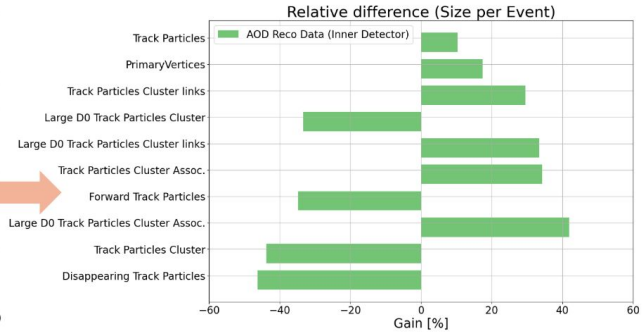
We can imagine a set of power tools, maintained outside the ROOT source tree.

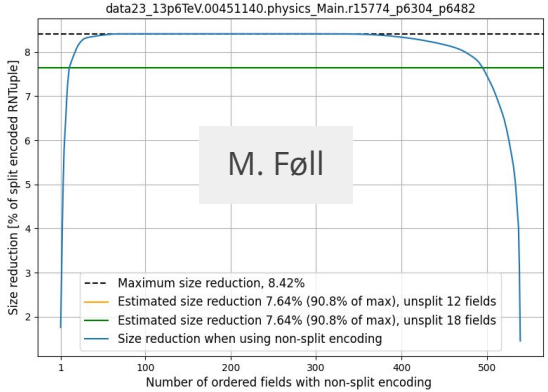
E.g., manual RNTuple descriptor manipulation.

Ongoing Issues

- Tuning (auto tuning?) of column encoding







→https://indico.cern.ch/event/1338689/contributions/6010824/

- Investigation of MiniAOD space savings (~7.5 %, would ideally be > 10% [somewhat arbitrary])
- Framework support: profile & improve writing and reading from frameworks
- Support for vectors with custom allocators (ATLAS)
- Support for writing into directories, bulk reading optimizations (ALICE)
- Validation suite for 3rd party readers
- I/O support for SoA data structures, see talk tomorrow
- Meta-data support, see talk tomorrow

Priorities for 2025

- Define the first set of APIs to move out of ROOT::Experimental
  - Planned for ROOT v6.36, i.e. likely May 2025
  - More or less the classes subject to the HEP-CCE review
  - We can extend the APIs later (e.g. additional ClusterPool tuning), but once in production it will be costly to change existing APIs
  - Not all RNTuple APIs will move out at the same time
- Fully functional schema evolution (basic functionality working for v6.36, full set possibly post v6.36)
- RNTupleProcessor: capability to arbitrarily combine friends and chains
- RNTuple attribute extension prototype (see later), likely leading to v1.1 ondisk format
- Testing and validation on IT testbed with data derivation and/or reconstruction benchmark(s)
- Tuning, support, bug fixes, training: with the transition to production, the support effort begins
- *Lower priority*: S3 backend, intra-event links, checkpoints during writing, sharded clusters and horizontal merge

Round table: questions to experiments

- Round table discussion starters:
  - From your point of view, are we missing anything important?
  - Is the parallel writer of interest to you?
  - How can we facilitate RNTuple adoption?
  - Can you provide us a benchmark (code + data) for a reconstruction or data derivation task for the IT testbed
    - Ideally: test also the parallel writer
    - Large data set would allow for validating possible automatic tuning of column encoding
  - Extra: we will (re)start development on RFile next year. What are your wishes (e.g. ownership model, concurrency, etc.)