Evil Storage Layer for RNTuple

*Victoria Porter
EP-SFT, CERN

1 Motivation

ROOT [1] is a data analysis framework used in High Energy Physics (HEP) experiments:

- More than **1 exabyte** of data is stored in ROOT files.
- Upcoming HEP experiments at the High-Luminosity LHC are expected to generate **at least 10×** as much event data.

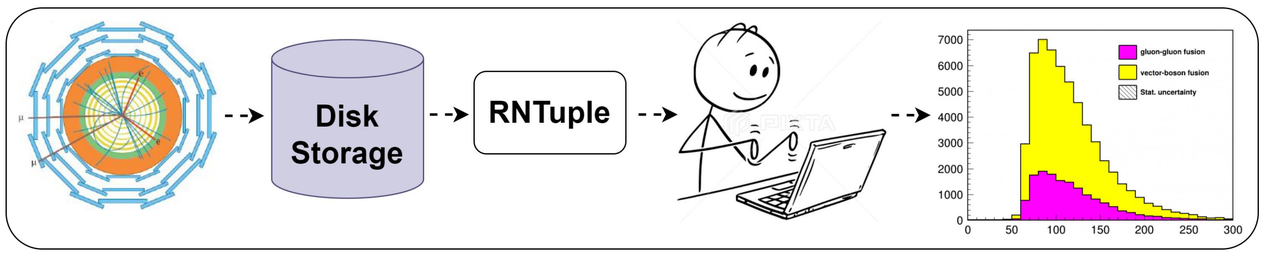


Fig. 1: HEP event data flow from detector to analysis.

Why do we care?

Reliability	Robustness	Reproducibility

Objectives:

- Implement an additional **testing layer** for RNTuple.
- Inject failures that emulate conditions such as **hardware faults** and **network glitches**.

2 Introduction

ROOT currently supports columnar storage through TTree:

- **Not optimised** for modern hardware and storage systems.
- Projected data increase calls for a **re-engineering of TTree**.

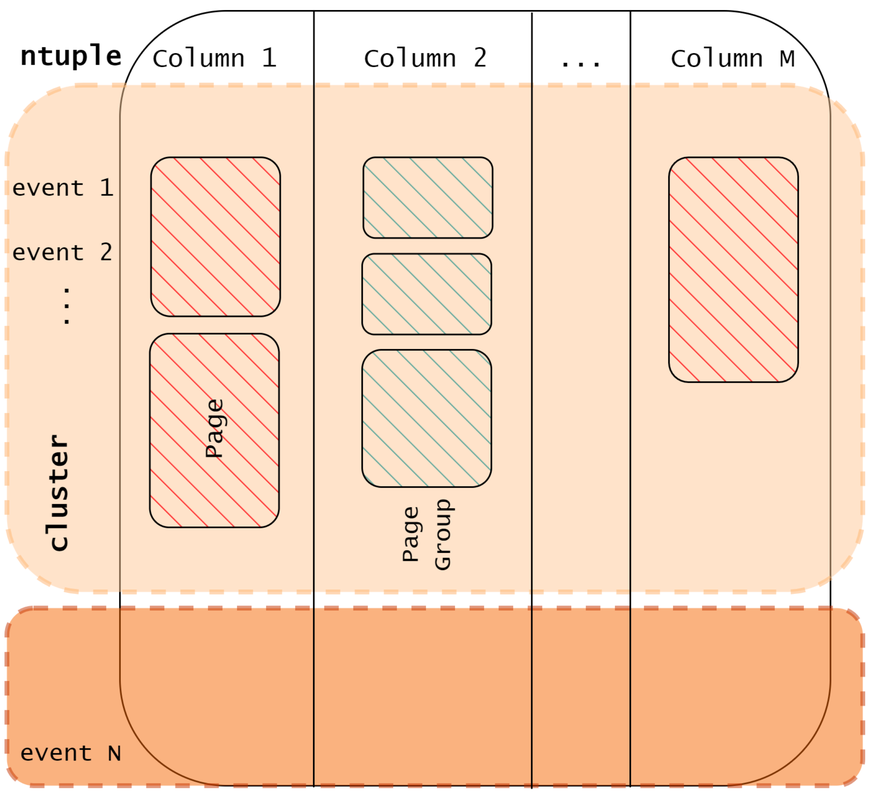


Fig. 2: RNTuple on-disk format [2].

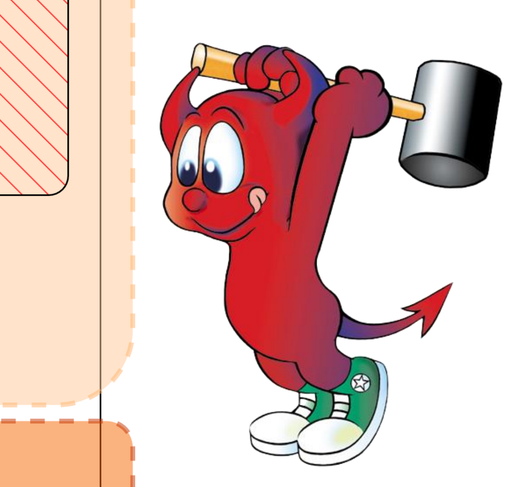
What is RNTuple?

- A **columnar I/O subsystem** for storing HEP event data.
- The experimental evolution of ROOT's TTree component.
- Designed to address **performance** and **scalability** issues.

What failures are injected?

- Bit flips
- Short reads
- Disk full
- Corrupted writes

Read / Write



3 Evil Storage Layer

RNTuple brings significant performance and usability improvements including **avoiding silent I/O errors** and **detecting corrupted data**.

What is an "evil" storage layer?

- A modified version of the main storage layer that **alters the behaviour** of I/O methods.

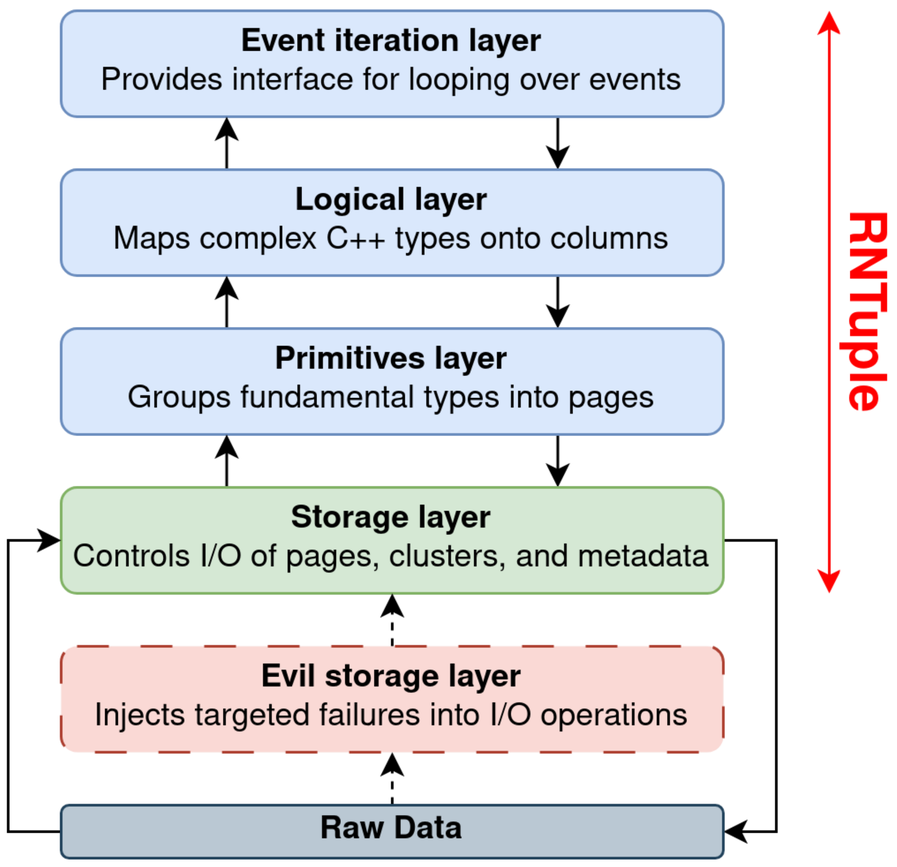


Fig. 2: RNTuple columnar I/O subsystem.

How does it work?

- The evil layer is activated by a **preprocessor macro**.
- On activation a **target byte range** will be intercepted and modified.
- Failures are triggered in the form of **unit tests**.

4 Discussion

What we hope to see:

✓ Error detection
✓ Tidy error messages
✓ Graceful handling

What we might get:

× Undetected errors
× Untidy error messages
× Nasty crashes

Why is this important?

- Data preservation.
- Hardware and Network failures ≠ software bugs.
- **Continuous Integration** for arbitrary failures.

5 Future Work

Continuous integration testing framework for **READ** and **WRITE** operations.

References

[1] https://github.com/root-project/root
[2] https://tinyurl.com/2h3vjpwu

*victoria.porter@cern.ch