Introducing RTreeMap & Extending RNTupleInspector

Patryk Tymoteusz Pilichowski, Summer Student 2025
Supervisors: Giacomo Parolini, Florine De Geus
University of Antwerp, Belgium

Abstract
ROOT is a widely used data analysis framework in high-energy physics, offering tools for efficient processing and scientific analysis of large datasets. Among these, RNTuple provides a modern data storage format that achieves superior throughput and compression compared to existing solutions. With its planned adoption as the main storage technology for the HL-LHC Run 4, ensuring that interfacing with an RNTuple is user-friendly has become crucial. This report presents the development of tools that visualize on-disk information of RNTuples, introducing the `RTreeMap` component and an enhancement to the `RNTupleInspector`.

Keywords
ROOT; JSROOT; RNTuple; RTreeMap; C++.

1 Introduction

1.1 ROOT and RNTuple

ROOT is a data analysis framework widely used in high energy physics, providing tools that enable processing and scientific analysis of large datasets. Among which is RNTuple, a columnar data storage format succeeding TTree in terms of speed, storage optimization, and interface robustness.

Internally, an RNTuple can be described as a tree of fields, where a field represents a serialized, logical data type. A single field is mapped to zero or more deserialized, primitive C++ data types, called columns.

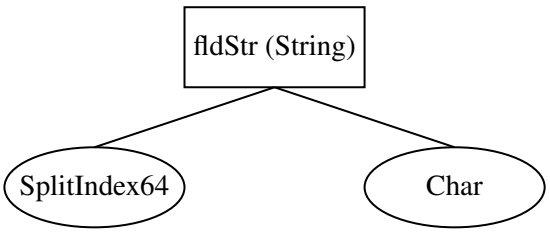


Figure 1: Example of a string field comprised of columns

1.2 Treemapping

When working with RNTuples, it can be useful to visualize the relative sizes on-disk of all fields and columns. Treemapping provides a way to address this challenge by visualizing tree-structured data. It uses nested rectangles whose area is proportional to the size of the associated data. Coloring the rectangles gives information about which category the data elements belong to. This makes treemapping a powerful visualization tool as it can simultaneously provide insights on hierarchy, proportion, and category of data, allowing for pattern discovery in large datasets.

There are different ways rectangles can be divided into sub-rectangles, known as tiling algorithms or tessellations. Disregarding the aspect ratio of the rectangles while tessellating a large dataset, often results in a convoluted visualization.

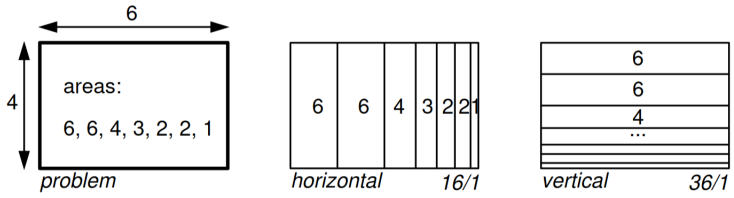


Figure 2: Example of a treemap with slice-and-dice tiling [1]

An approach to mitigate such convolution would be to find a configuration of rectangles such that their aspect ratio is as close to 1 as possible while remaining proportionally consistent. However, finding an optimal solution to this problem is NP-hard. Fortunately, the squarification algorithm computes a good-enough solution in efficient time.

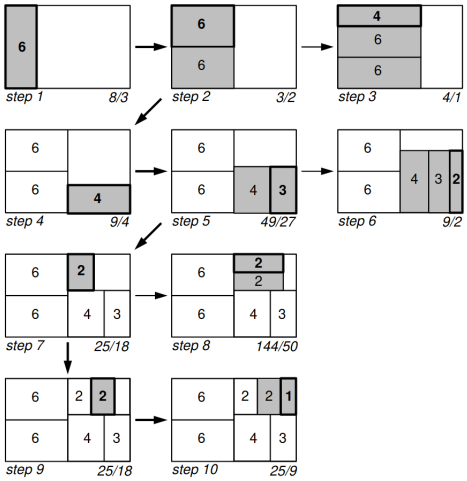


Figure 3: Squarified tiling with the previous example [1]

1.3 Motivation

As it is planned that RNTuple will be used as the main storage format for high-luminosity LHC data in run 4, it is highly desirable to make interfacing with RNTuple more user-friendly. Adding visualizations revolving around RNTuples can be a way to do so.

2 Visualizing the RNTuple model

RNTupleInspector is a utility class providing functions to inspect several aspects of an RNTuple. As part of this work, we added a new functionality that produces the tree of fields that represent an RNTuple, using the method `RNTupleInspector::PrintFieldTreeAsDot()`. It outputs the tree in .dot format, that can then be visualized with a library like graphviz [2].

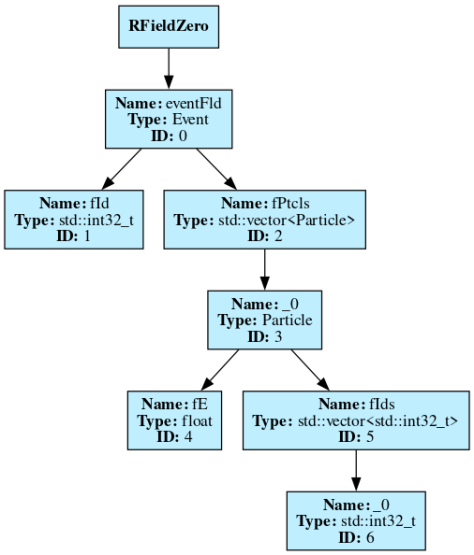


Figure 4: Field tree of an RNTuple

3 RTreeMap

As part of this work, we developed a new component called RTreeMap that lets users visualize RNTuple's fields and columns as a squarified treemap, while remaining sufficiently generic to allow for any tree-structured data to be used. The rectangles that represent fields are gray-colored and have the field name as their label.

On the other hand, the colored rectangles represent columns to which the parent field is mapped to. The legend displayed on the right side of the visualization gives the column type for the corresponding color, as well as additional information like their space usage.

One can visualize an arbitrary RNTuple with just three lines of C++ code:

```
auto treemap = ROOT::Experimental::CreateTreeMapFromRNTuple("file.root", "ntuple_name");
auto c = new TCanvas("c_tm","TreeMap");
c->Add(treemap.release());
```

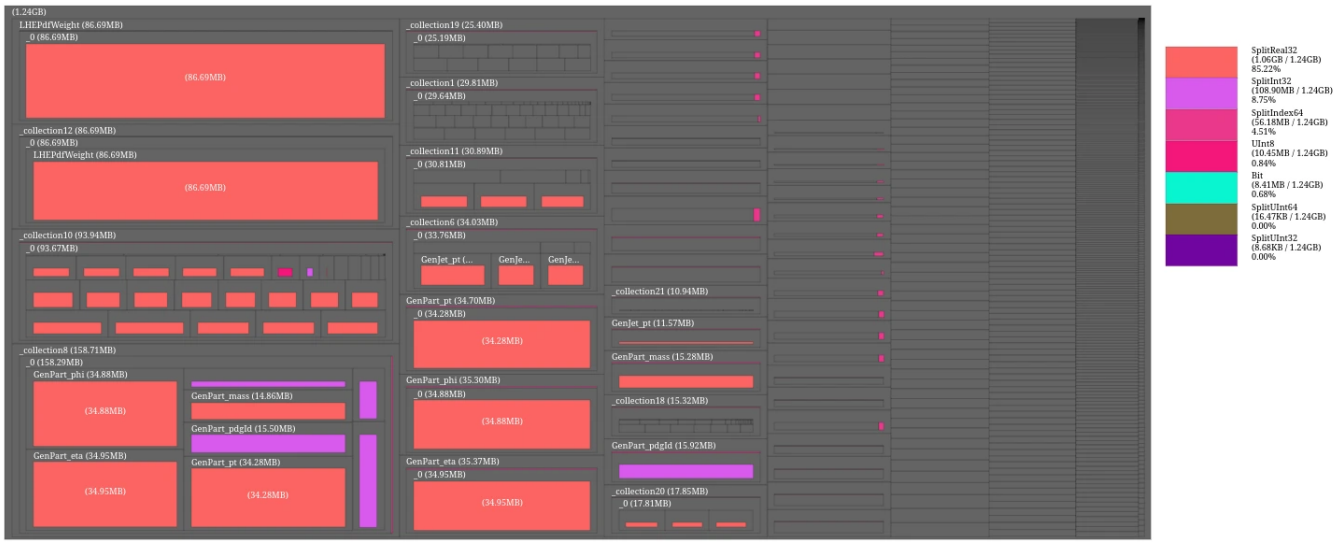


Figure 5: RTreeMap of a large (1GB) RNTuple

4 ROOT Browser Integration

The component is integrated to both to the classic and the web-based ROOT browser. The JSROOT implementation of RTreeMap provides extra interactive features:

- Hovering on a rectangle displays additional information
- Hovering on a legend entry highlights all rectangles with the hovered color
- Clicking on a sub-rectangle "zooms" into it by one level, producing sub-treemap with the clicked sub-rectangle as its new bounding rectangle
- Clicking on the bounding rectangle inside a sub-treemap zooms the treemap out by one level
- Text above the treemap that displays current directory that it is zoomed into

5 Future work

Possible future directions on RNTuple visualization include:

- Adding an attributes dictionary to `RTreeMapBase::Node` which can be displayed when hovering on the rectangles
- Extending the field tree visualization by giving an option to also visualize the columns as leaves in the tree
- Improve the squarification of rectangles by adding a heuristic to the tiling algorithm [3]
- Adding an in-depth tutorial on how to make custom RTreeMaps to the documentation

Acknowledgements

I would like to sincerely thank Giacomo, my supervisor, for his continuous support and invaluable expertise throughout this stimulating project.

I am also deeply grateful to Florine and the ROOT team for their feedback and for giving me the opportunity to contribute to the ROOT project.

Bibliography

[1] Bruls, M., Huizing, K., & Wijk, J. J. V. (2000). Squarified Treemaps. https://doi.org/10.2312/VISSYM/VISSYM00/033-042

[2] Graphviz. (n.d.). Graphviz. https://graphviz.org/

[3] Cesarano, A., Ferrucci, F., & Torre, M. (2016). A heuristic extending the Squarified treemapping algorithm. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1609.00754