Automatic conversion of data stored in TTree form to RNTuple

Javier Lopez-Gomez
<j.lopez@cern.ch>

Jakob Blomer
<jblomer@cern.ch>

March 9, 2022

1 Introduction

The ROOT Software Framework is the cornerstone of many software stacks used by H igh Energy Physics (HEP) experiments, at CERN and other prestigious laboratories . It provides components which are fundamental for the entire data processing ch ain, from particle collisions to final publications, including final user data a nalysis, including modern machine learning techniques.

Analyses of High-Energy Physics data typically only require a subset of the properties stored for each event. In this context, row-wise event storage (i.e. one record per event) has been proved to be suboptimal as it causes unneeded data to be read from persistent storage. In contrast, a columnar storage organizes the data set in columns and internally groups values that belong to the same data member. TTree is the ROOT's legacy columnar storage that has been used to store more than 1 exabyte of HEP data during the last 25 years. On the other hand, the RNTuple classes provide ROOT's new, experimental I/O subsystem for high-energy physics data. The RNTuple data layout is columnar and supports nested types (e.g., vectors of floats), conceptually similar to Apache Arrow or Apache Parquet. RNTuple has been designed for efficiency and to take advantage of modern storage systems, particularly low-latency high-bandwidth SSDs and object storages, e.g. Intel DAOS.

As said previously, a vast amount of data has been collected by experiments and stored in TTree format. Given that RNTuple is a complete backwards-incompatible redesign, the development of an automatic conversion tool that permits the migration of existing TTree data into RNTuple is a must. In this regard, both the schema (i.e., fields and their types) and the data will have to migrated. Note that replicating the schema is not always possible because RNTuple does not currently support all the column types supported in TTree. If the schema migration succeeds, existing data, i.e. rows, shall also be imported into the RNTuple.

For this first phase of the selection process, you have been assigned two different tasks. The solution to both tasks should be submitted by e-mail to your mentors (CC both) by **20th of March**. As part of the solution, you can send source code and/or a report in PDF format. If you have any further questions regarding this assignment, do not hesitate to contact the mentors.

2 Task 1: Build ROOT

In this first task, you are requested to build ROOT[1] from sources. In this task, you are expected to familiarize with the ROOT source tree and the required dependencies. Also, if you have not used ROOT before, we also encourage you to try some of the features described in [2].

Instructions on how to build ROOT from source can be found in [3]. In order to also build RNTuple, you will have to specify the `-Droot7=ON` option in the CMake command line. In general, you will not require the whole history of the project; therefore, you can save some time if you get the latest revision on the `master` branch, i.e.

```
$ git clone -b master --depth=1 https://github.com/root-project/root

$ # The following option should be specified to build RNTuple
$ cmake -Droot7=ON ...
```

3 Task 2: RNTuple specific task

For this task, you are requested to write a program that takes as input a CSV file and generates a RNTuple ROOT file. The path to both files will be specified on the command line, e.g.

```
$ ./task2 input_file.csv output_file.ntuple
```

This program will make use of the RNTuple interface to create the corresponding fields (one per column) and import data from the input file. Additionally, the first row of the input CSV file will be used as a header, including the name and type for each column. Such entries will be of the form "`name:type`". For the sake of simplicity, we will only have "`int`" and "`float`" columns. In Listing 1, we show an example of input CSV file.

The Doxygen documentation for RNTuple classes can be found in [4]. In [5], you can find some tutorials that illustrate RNTuple's usage. Additionally, Jakob's iotools repository [6] includes some more advanced use cases. In case you need to take a look at RNTuple source code, it is located in the `tree/ntuple/v7/` directory in the ROOT repository. You have no restrictions on how to implement this program, as long as the solution is technically sound and makes correct use of the RNTuple interface.

Listing 1: Example of input CSV file with 6 columns. Note the header line specifying the names and types for each column.

```
An_Integer:int, pX:float, pY:float, pZ:float, Integer_2:int, ProbK:float
0, 2.123, 1.31, 1.0, 0, 9132.1433
1, 2232.3, 3.10, 0.1231, 2, 2.3
2, 322.36, 45.1, 12.1, 85, 124.0
56, 24.1, 2541.2, 15.0, 4, 83.323
...
```

References

[1] ROOT: analyzing petabytes of data, scientifically. URL: https://root.cern/.

[2] ROOT Primer. URL: https://root.cern/primer/.

[3] Installing ROOT. URL: https://root.cern/install/.

[4] ROOT::Experimental::RNTupleWriter Class Reference. URL: https://root.cern.ch/doc/master/classROOT_1_1Experimental_1_1RNTupleWriter.html.

[5] ROOT 7 ntuple tutorials. URL: https://root.cern/doc/master/group__tutorial__ntuple.html.

[6] ROOT RNTuple Virtual Probe Station. URL: https://github.com/jblomer/iotools/.