# Separation of concerns

Refactoring code from spark-root to root4j

- Pratyush Das

## Reading information from ROOT files

For more than a decade, the main way to read ROOT files directly and display the contents has been by using the C++ ROOT data analysis framework.

But it is incredibly hard to run C++ ROOT in Java.

## Solution

A set of libraries were written in java in a project called root4j and a project called spark-root was coded in scala to use the large scala data analysis framework apache spark and display the information from the ROOT file.

This is meant to be a drop-replacement for C++ ROOT reading that works in a Java environment.

But what exactly is spark-root and root4j and what do they do?

## root4j

- It is a library which can be used to read ROOT files.
- It has no connections to the original implementation of ROOT in C++.
- It is written completely in Java.
- It is a fork of the original freeHEP ROOTIO.

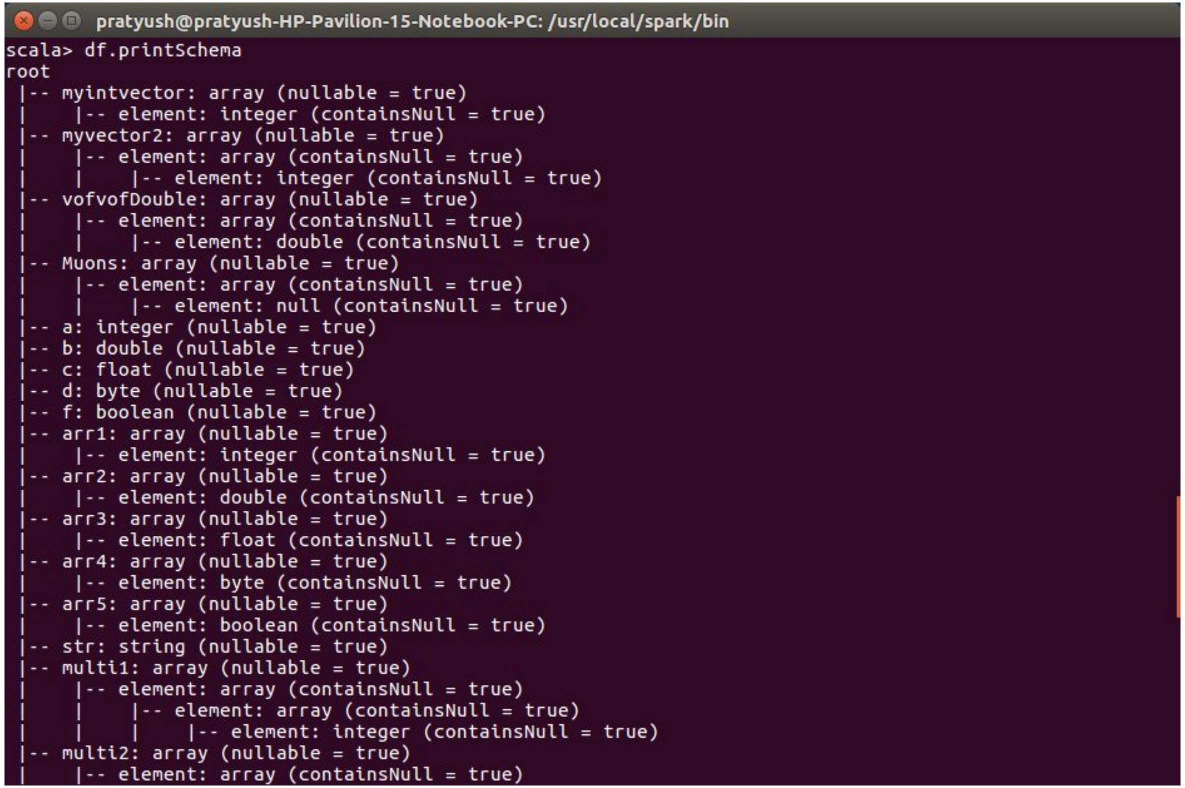
## Spark - root

- Spark-root directly displays the contents of root files as Spark DataFrames by importing root4j libraries as a dependency.
- It is written in scala.
- Physicists around the world can use spark-root and projects built on spark-root to read ROOT files without having to reformat petabytes of data to something else first.

# Example of spark-root reading

Schema of sample file -

test_root4j.root



pratyush@pratyush-HP-Pavilion-15-Notebook-PC: /usr/local/spark/bin

```
scala> df.printSchema
root
 |-- myintvector: array (nullable = true)
 |    |-- element: integer (containsNull = true)
 |-- myvector2: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: integer (containsNull = true)
 |-- vofvofDouble: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: double (containsNull = true)
 |-- Muons: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: null (containsNull = true)
 |-- a: integer (nullable = true)
 |-- b: double (nullable = true)
 |-- c: float (nullable = true)
 |-- d: byte (nullable = true)
 |-- f: boolean (nullable = true)
 |-- arr1: array (nullable = true)
 |    |-- element: integer (containsNull = true)
 |-- arr2: array (nullable = true)
 |    |-- element: double (containsNull = true)
 |-- arr3: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- arr4: array (nullable = true)
 |    |-- element: byte (containsNull = true)
 |-- arr5: array (nullable = true)
 |    |-- element: boolean (containsNull = true)
 |-- str: string (nullable = true)
 |-- multi1: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: array (containsNull = true)
 |    |    |    |-- element: integer (containsNull = true)
 |-- multi2: array (nullable = true)
 |    |-- element: array (containsNull = true)
```

## Previous functionality

- Perfect separation between the functionality of spark-root and root4j was not maintained.
- Root4j could neither take ROOT files independently as input from the user nor display the information stored in it.
- A lot of the code to iterate over the TTrees was written in spark-root as well as the code to generate the output from the ROOT files.

## Project goal

- Root4j should be a standalone application which would be able to read and display the contents of all ROOT files without having to depend on external libraries to do so.
- Root4j should just be a java based ROOT reader.
- Spark-root should just contain the spark interface.
- Root4j works entirely on the JVM(Java Virtual Machine) so it should be accessible as a dependency for all java products to be used as a medium to read ROOT files.

## Work done for implementation

- Moved the TTree iterating code from spark-root to root4j. (From scala to java)
- Wrote a class to display the contents of the ROOT file.
- When the hdfs implementation is complete and the file is read by root4j, it should be able to directly iterate over all the branches of the TTree and print out the results .
- Root4j cannot as of now directly take files as input from the user and I am currently working on getting it done via the HDFS (Hadoop Distributed File System).

## Utility of reading ROOT files via the JVM

Since the root reader is in java, we can link it with other big data analysis softwares based on the JVM.

Like -

1. Hadoop
2. Apache HBase
3. Apache Storm

## Final functionality to be present in spark-root

- Read the abstractly typed tree generated in the newer version of root4j.
- Convert the data types into a spark readable format.
- Creating a schema to convert the data into rows and columns which can be displayed via Spark.

## Test case - GROOT

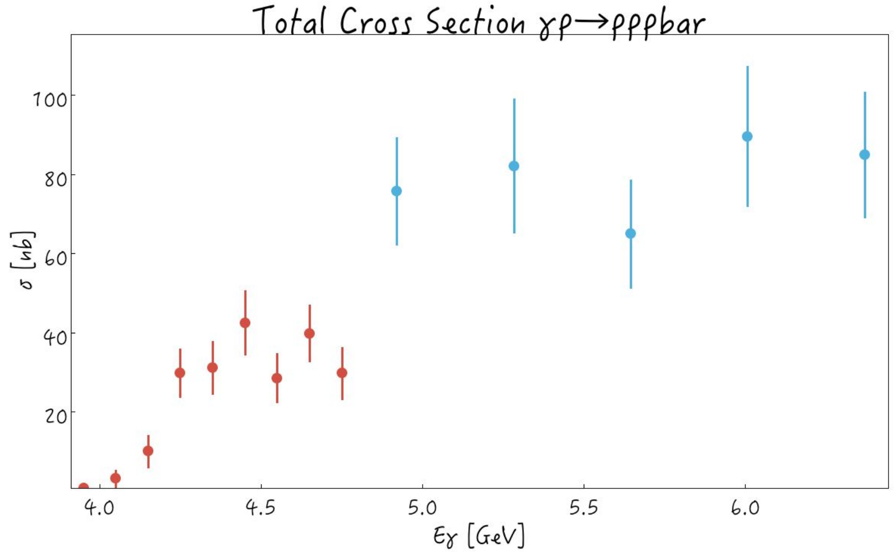
A nuclear physics group at Jefferson Lab has developed ROOT like visualizations in Java by developing GROOT.

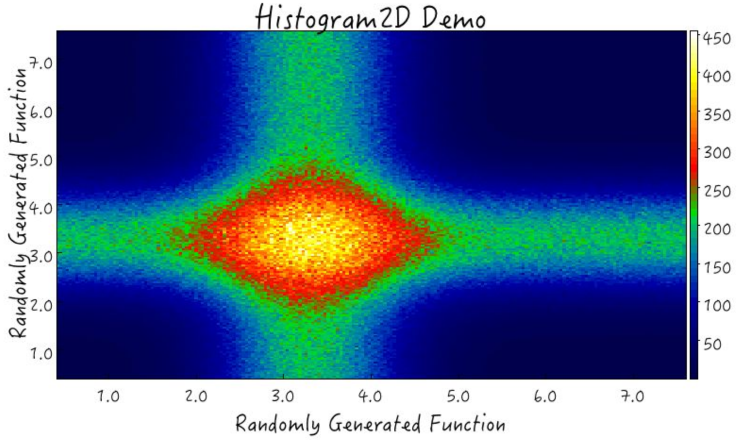
GROOT is a java data visualization project designed to make histogramming, plotting and fitting accessible to all java users.

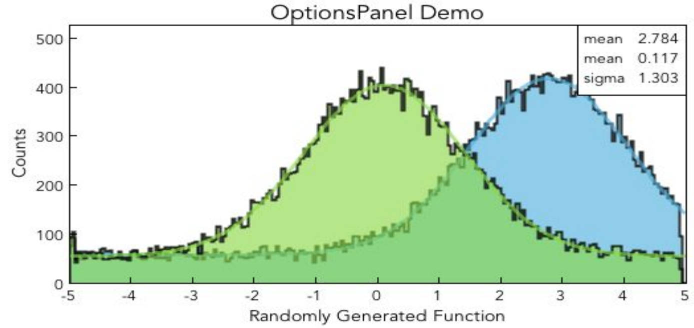
We can test if all the functionality has been ported by importing root4j as a dependency to GROOT and seeing if spark-root and GROOT can read the same files.

Currently we cannot test this since the hdfs implementation to read files directly into root4j isn’t complete.

# GROOT data visualization - Example







## Work to be done

1. Testing out the functionality of the code by loading files to root4j using HDFS (Hadoop Distributed File System).
2. Making sure root4j is able to read the file from the local filesystem instead of having to upload the file first to the HDFS and then calling it from that.
3. Removing the translated code from spark-root and replacing it with function calls to the now refactored over code in root4j to omit any code duplicacy.

# Thank you