ROOT4J / SPARK-ROOT: ROOT I/O for JVM and Applications for Apache Spark

V. Khristenko[1] J. Pivarski [2]

[1] Department of Physics
The University of Iowa

[2] Princeton University - DIANA

ROOT I/O Workshop, 2017

Outline

Motivation

- Enable access to Physics Data from SPARK.
- ROOT Data Format is, almost, self-descriptive -> JVM-based I/O is therefore a realistic goal!
- Open up ROOT for the use with Big Data Platforms (Spark is just a single example)

What SPARK-ROOT is

Only I/O

The primary objective of this work is to provide a JVM-based access to ROOT's binary format

- SPARK-ROOT is a ROOT's I/O Library for JVM.
- SPARK-ROOT is purely Java/Scala based.
- SPARK-ROOT implements a new Spark Data Source, similar to Parquet, Avro.

TTree as Spark Dataframe

SPARK-ROOT allows to access binary ROOT format within JVM directly and represent ROOT TTree as Spark's Dataset/Dataframe/RDD.

Supported Datatypes

- Basic Types: Integer, Boolean, Float, Double, Long, Char, Char*
- Fixed-size Arrays and variable sized arrays
- Multidimensional Arrays
- Pointers to basic Types - a la dynamic arrays
- Structs (in multi-leaf style)
- STL Collections (for now, map/vector) of basic types
- Nested STL Collections of basic types
- STL String
- Composite Classes of Basic Types and of Composite Classes
- STL Collections of Composite Classes
- STL Collections of Composite with STL Collections of Composite as class members - multi-level hierarchy
- TClonesArray, when member class is available before Read-Time!

Supported Functionality

- JIT compilation using TStreamerInfo to get to TTree
- Automatic Spark Schema Inferral for supported types in the TTree.
- Proper Branch Flattening
- Hadoop DFS Support
- Early Stage Filtering

Limitations

Run/Read-Time Limitations of Spark

Spark builds a schema before the actual reading is done. It imposes constraints that all the data types must be known a priori to reading! Not the case for ROOT!

```
class Base {...};
class Derived1 : public Base {...};
class Derived2 : public Base {...};

std::vector<Base*> - at read/run-time can be ...
1) std::vector<Derived1>
2) std::vector<Derived2>
3) std::vector<Base>

Same idea applies to TClonesArray.
```

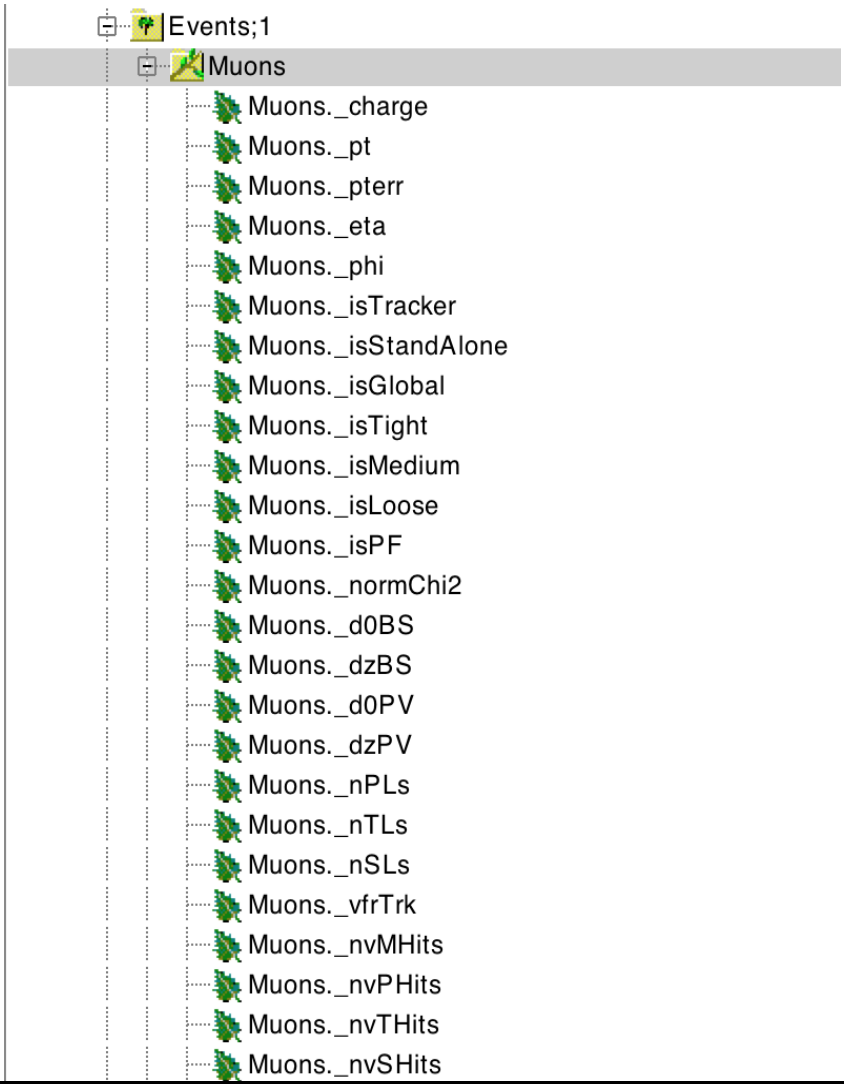
CMS Higgs Analysis

```
./spark-shell --packages
    org.diana-hep:spark-root_2.11:0.1.7

import org.dianahep.sparkroot._

scala> val df = spark.sqlContext.read.root(
    "file:/Users/vk/software/Analysis/files/test/
    ntuple_drellyan_test.root")
```

CMS Higgs Analysis

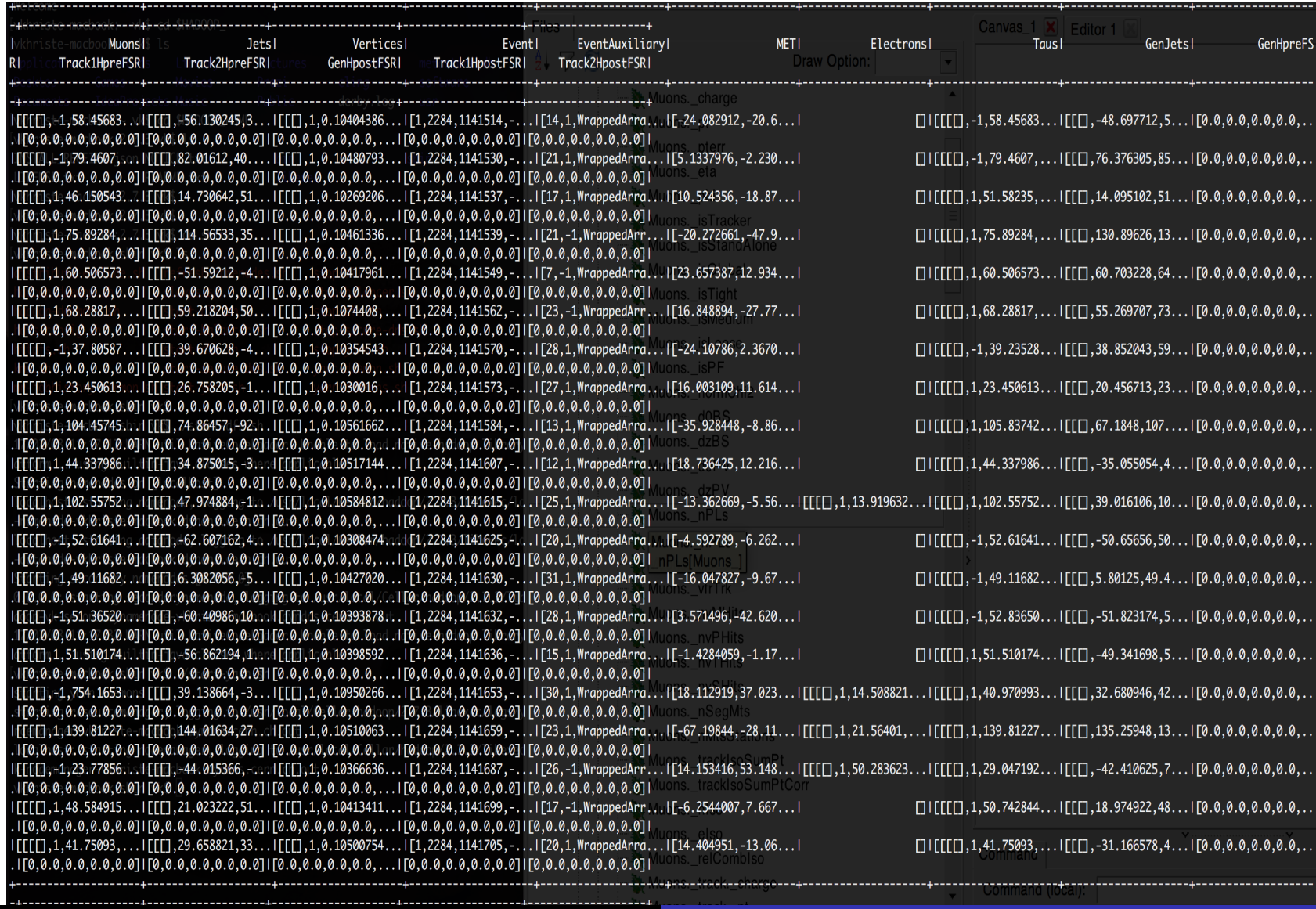
```
scala> df.printSchema
 |-- Muons: array (nullable = true)
 |     |-- element: struct (containsNull = true)
 |     |     |-- analysis::core::Track: struct (nullable = true)
 |     |     |     |-- analysis::core::Object: struct (nullable =
 |     |     |     |-- _charge: integer (nullable = true)
 |     |     |     |-- _pt: float (nullable = true)
 |     |     |     |-- _pterr: float (nullable = true)
 |     |     |     |-- _eta: float (nullable = true)
 |     |     |     |-- _phi: float (nullable = true)
 |     |     |-- _isTracker: boolean (nullable = true)
 |     |     |-- _isStandAlone: boolean (nullable = true)
....
 |     |     |-- _track: struct (nullable = true)
 |     |     |     |-- analysis::core::Object: struct (nullable =
 |     |     |     |-- _charge: integer (nullable = true)
 |     |     |     |-- _pt: float (nullable = true)
 |     |     |     |-- _pterr: float (nullable = true)
```

CMS Higgs Analysis




```
|-- Muons: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- analysis::core::Track: struct (nullable = true)
|    |    |    |-- analysis::core::Object: struct (nullable = true)
|    |    |    |    |-- _charge: integer (nullable = true)
|    |    |    |    |-- _pt: float (nullable = true)
|    |    |    |    |-- _pterr: float (nullable = true)
|    |    |    |    |-- _eta: float (nullable = true)
|    |    |    |    |-- _phi: float (nullable = true)
|    |    |    |-- _isTracker: boolean (nullable = true)
|    |    |    |-- _isStandAlone: boolean (nullable = true)
|    |    |    |-- _isGlobal: boolean (nullable = true)
|    |    |    |-- _isTight: boolean (nullable = true)
|    |    |    |-- _isMedium: boolean (nullable = true)
|    |    |    |-- _isLoose: boolean (nullable = true)
|    |    |    |-- _isPF: boolean (nullable = true)
|    |    |    |-- _normChi2: float (nullable = true)
|    |    |    |-- _d0BS: float (nullable = true)
|    |    |    |-- _dzBS: float (nullable = true)
|    |    |    |-- _d0PV: float (nullable = true)
|    |    |    |-- _dzPV: float (nullable = true)
|    |    |    |-- _nPLs: integer (nullable = true)
|    |    |    |-- _nTLs: integer (nullable = true)
|    |    |    |-- _nSLs: integer (nullable = true)
|    |    |    |-- _vfrTrk: float (nullable = true)
|    |    |    |-- _nvMHits: integer (nullable = true)
|    |    |    |-- _nvPHits: integer (nullable = true)
|    |    |    |-- _nvTHits: integer (nullable = true)
|    |    |    |-- _nvSHits: integer (nullable = true)
|    |    |    |-- _nSegMts: integer (nullable = true)
|    |    |    |-- _nMtsStations: integer (nullable = true)
|    |    |    |-- _trackIsoSumPt: float (nullable = true)
|    |    |    |-- _trackIsoSumPtCorr: float (nullable = true)
|    |    |    |-- _hIso: float (nullable = true)
|    |    |    |-- _eIso: float (nullable = true)
|    |    |    |-- _relCombIso: float (nullable = true)
|    |    |-- _track: struct (nullable = true)
|    |    |    |    |-- analysis::core::Object: struct (nullable = true)
|    |    |    |    |-- _charge: integer (nullable = true)
|    |    |    |    |-- _pt: float (nullable = true)
|    |    |    |    |-- _pterr: float (nullable = true)
|    |    |    |    |-- _eta: float (nullable = true)
|    |    |    |    |-- _phi: float (nullable = true)
|    |    |    |-- _segmentCompatibility: float (nullable = true)
|    |    |    |-- _combinedQChi2LocalPosition: float (nullable = true)
```

CMS Higgs Analysis



CMS Higgs Analysis

Scaling up

Very easy to scale up to the whole dataset - 400GB of Run 2 data.

```
./spark-shell --packages
    org.diana-hep:spark-root_2.11:0.1.7

import org.dianahep.sparkroot._

scala> val df = spark.sqlContext.read.root(
    "hdfs:/cms/bigdatasci/vkhriste/
    data/higgs/data/SingleMuon")
```

CMS AOD public Muonia Dataset

Public 2010 data

1.2TB of public Muonia dataset on CERN's hdfs.

```
./spark-shell --packages
    org.diana-hep:spark-root_2.11:0.1.7

import org.dianahep.sparkroot._

scala> val df = spark.sqlContext.read
    .option("tree", "Events")
    .root("hdfs:/cms/bigdatasci/vkhriste/
        data/publiccms_muionia_aod")
```

CMS AOD public Muonia Dataset

```
scala> df.printSchema
root
 |-- EventAuxiliary: struct (nullable = true)
 |    |-- processHistoryID_: struct (nullable = true)
 |    |    |-- hash_: string (nullable = true)
 |    |-- id_: struct (nullable = true)
 |    |    |-- run_: integer (nullable = true)
 |    |    |-- luminosityBlock_: integer (nullable = true)
 |    |    |-- event_: integer (nullable = true)
 |    |-- processGUID_: string (nullable = true)
 |    |-- time_: struct (nullable = true)
 ...
 |-- recoMuons_muons__RECO_: struct (nullable = true)
 |    |-- edm::EDProduct: struct (nullable = true)
 |    |-- present: boolean (nullable = true)
 |    |-- recoMuons_muons__RECO_obj: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- reco::RecoCandidate: struct (nullable = true)
 |    |    |    |    |-- reco::LeafCandidate: struct (nullable = true)
 |    |    |    |    |    |-- reco::Candidate: struct (nullable = true)
 |    |    |    |    |    |-- qx3_: integer (nullable = true)
 |    |    |    |    |    |-- pt_: float (nullable = true)
 |    |    |    |    |    |-- eta_: float (nullable = true)
 |    |    |    |    |    |-- phi_: float (nullable = true)
 |    |    |    |    |    |-- mass_: float (nullable = true)
 |    |    |    |    |    |-- vertex_: struct (nullable = true)
 |    |    |    |    |    |    |-- fCoordinates: struct (nullable = true)
 |    |    |    |    |    |    |    |-- fX: float (nullable = true)
 |    |    |    |    |    |    |    |-- fY: float (nullable = true)
 |    |    |    |    |    |    |    |-- fZ: float (nullable = true)
 ...
```

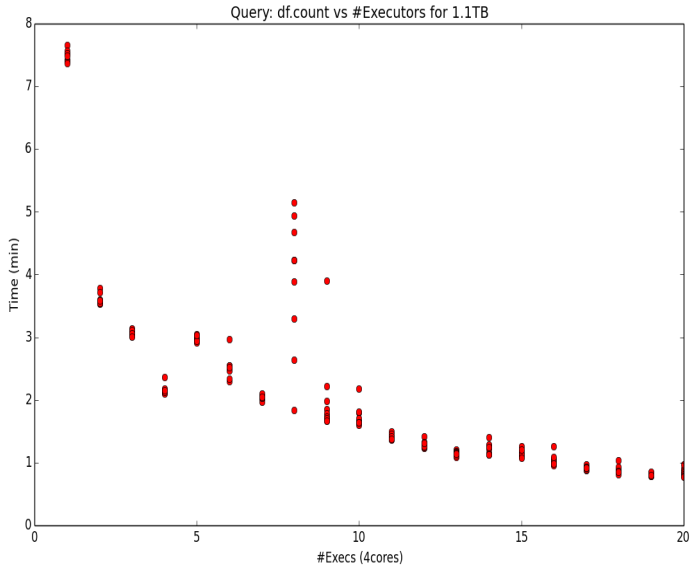
CMS AOD public Muonia Dataset

```
scala> df.select(s"recoMuons_muons__RECO_.recoMuons_muons__RECO_obj.
    reco::RecoCandidate.reco::LeafCandidate.pt_").show
recoMuons_muons__RECO_
[Lorg.apache.spark.sql.sources.Filter;@5133bd56
+--------------------+
|                 pt_|
+--------------------+
|[3.085807, 1.2784...|
|[4.1558356, 1.025...|
|[3.8067229, 2.142...|
|[2.4893947, 1.337...|
|          [4.5430374]|
|[3.1356623, 1.431...|
|[2.1504705, 2.129...|
|          [3.2125602]|
|          [4.3416142]|
|[2.1879413, 0.956...|
|           [5.258412]|
|           [5.627528]|
|[3.8034406, 6.120...|
|          [2.0771139]|
|           [3.891133]|
|           [5.891902]|
|[2.226252, 3.6012...|
|          [6.2603984]|
|          [1.8396659]|
|[1.7337813, 1.278...|
+--------------------+
only showing top 20 rows
```

Selecting Muon's pt and dumping first 20 entries

Basic Performance

- CMS Public Dataset for benchmarks
- Spark's Listeners to collect performance information.
- **Preliminary Results for 1.2TB (>1K files) for df.count**



Summary

- Huge Huge Thanks to Philippe, Danilo, Axel, Sergey Linev for replying to my questions!
- root4j/spark-root - JVM-based ROOT I/O library. It Works!
- spark-root allows one to view TTree as Spark Dataframe
- spark-root 0.1.7 is available on Maven Central for use
- Limitations do exist, but resolveable!

What's next?!

- There is no I/O Optimization implemented yet
- HDFS Locality - right now only HDFS access is done.
- Tuning Partitioning/Splitting - currently it's file-based
- Name Aliasing - useful for physicists
- Cross-references, a la TRef???
- Overcome the limitations
- In principle, root4j should be rewritten from scratch
- Prepare a decent TestBed - given Scala has a superb support for that!

Github and Useful Links

- spark-root
- spark-root Scala User Guide
- root4j