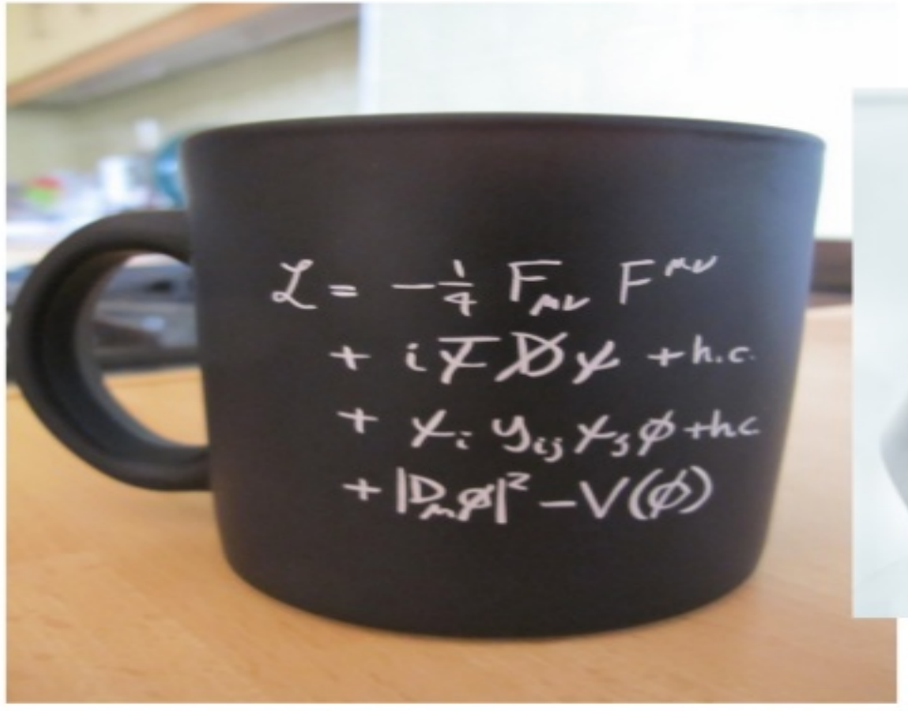
HEP Data Processing with Apache Spark

Viktor Khristenko (CERN)

Agenda

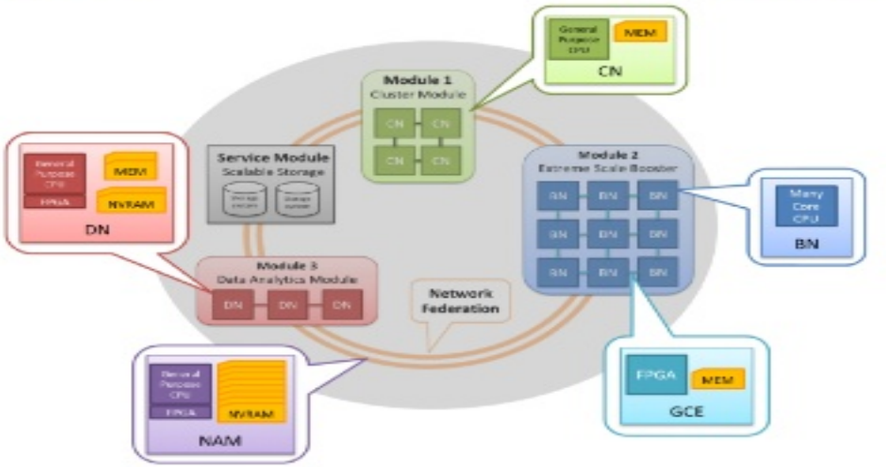
- HEP, CERN, LHC
- The DEEP-EST Project
- Motivation
- Current HEP
- A new Data Source
- Examples, examples, examples…

Experimental High Energy Physics



Employing HPC: The DEEP-EST Project

- DEEP– Extreme Scale Technologies
- R&D for Exascale HPC
- CERN is a collaborating partner
- European Project aiming to build Modular Supercomputing Architecture. Located at Juelich Supercomputing Center (JSC)

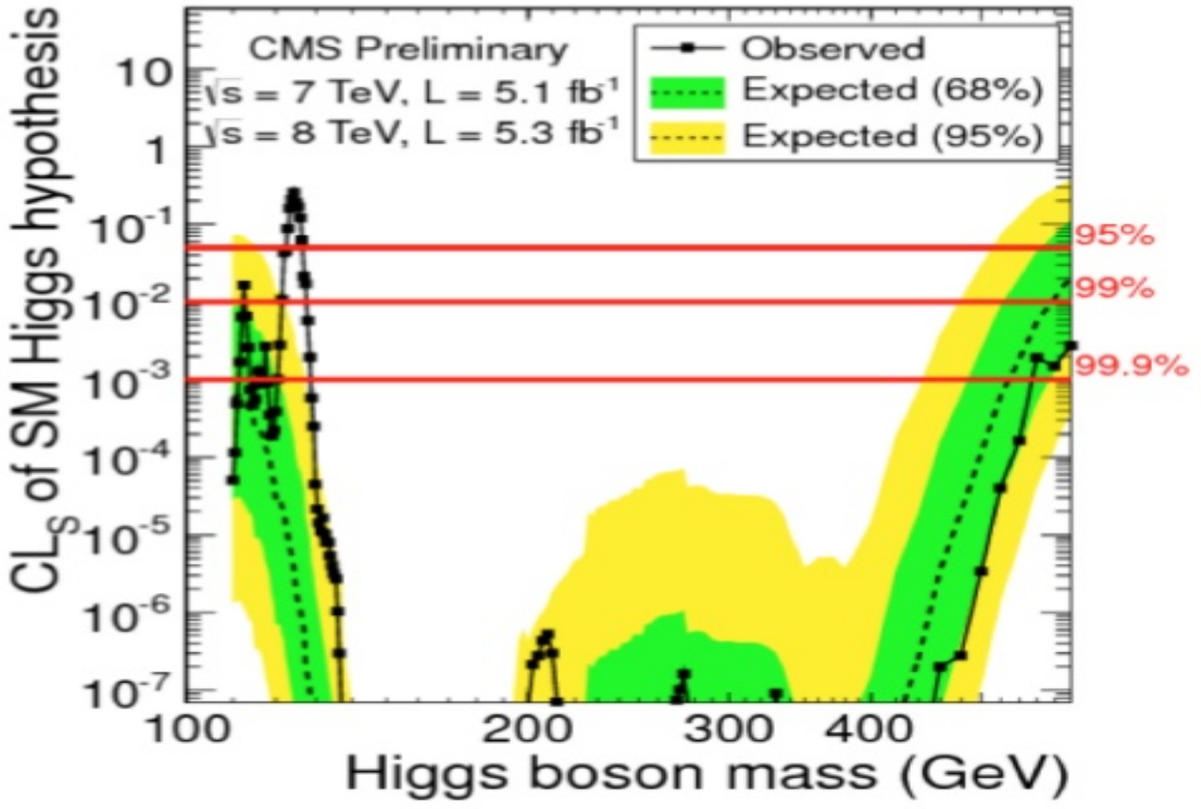


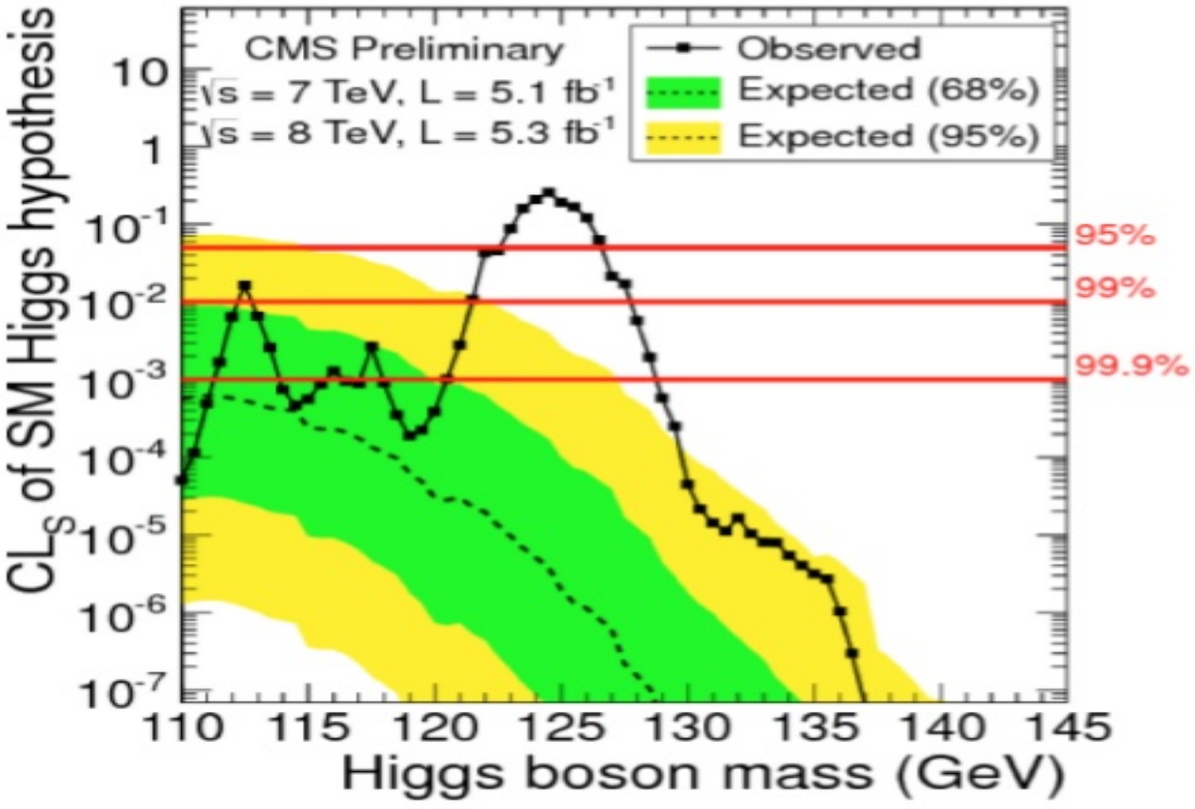
Motivation

- Explore novel approaches to perform HEP data analysis

- Explore Deep Learning HEP use-cases
  - Unified API

- Explore High Performance Computing resources for HEP Data Analysis


CMS Preliminary
√s = 7 TeV, L = 5.1 fb-1
√s = 8 TeV, L = 5.3 fb-1
Observed
Expected (68%)
Expected (95%)
95%
99%
99.9%
CLs of SM Higgs hypothesis
Higgs boson mass (GeV)


CMS Preliminary
√s = 7 TeV, L = 5.1 fb-1
√s = 8 TeV, L = 5.3 fb-1
Observed
Expected (68%)
Expected (95%)
95%
99%
99.9%
CLs of SM Higgs hypothesis
Higgs boson mass (GeV)

HEP Data Analysis

- c++ / python based
- PBs of HEP data stored in ROOT File Format
- Most of the HEP specific functionality comes from ROOT Data Analysis Framework
  - Histogramming
  - Fitting, Regression, ML
  - Graphics
  - Optimized Math
  - And much more
- Batch Processing, typically custom workload distribution.

Example

```
// open up a new file and retrieve a tree structure
TFile *f = new TFile(fileName.c_str());
TTree *t = (TTree*)f->Get("TestIO");

// select a column to use
double darr[NUM][NUM];
t->SetBranchAddress("darr", &darr);

// loop over all rows and compute a reduction
double totalSum = 0;
for (auto i=0; i<NUM_EVENTS; i++) {
    t->GetEntry(i);
    for (auto ii=0; ii<NUM; ii++)
        for (auto jj=0; jj<NUM; jj++)
            totalSum += darr[ii][jj];
}
```

No Joins

ROOT File Format and I/O

- Binary format
- Persistence (Serialization + I/O) of c++ objects
  - More general than just storing a collection of rows
- ROOT's Datasets allow
  - Columnar persistent storage
  - Splitting of nested columns
- C++ integration
  - c++ type system used directly – no intermediate representation
  - Schema (all the serialized c++ types) is stored within a ROOT file

Alright, we have

- ROOT files
- Apache Spark
- Conversion to parquet is not feasible => conversion will not scale
- We need to create a new Data Source to utilize the Dataset API

https://github.com/diana-hep/spark-root

Let's jump right in => spark-root

```
// data can be found from CERN OpenData portal:
// http://opendata.atlas.cern/extendedanalysis/datasets.php
import org.dianahep.sparkroot.experimental._
val df = spark.read.root("DataMuons.root")

// schemas are typically at least several pages long
df.printSchema

df.select("lep_E").show
+--------------------+
|               lep_E|
+--------------------+
|           [42227.465]|
|            [74975.45]|
|            [95780.08]|
|           [105389.39]|
|[44500.715, 43901...|
```

```
root
 |-- runNumber: integer (nullable = true)
 |-- eventNumber: integer (nullable = true)
 |-- channelNumber: integer (nullable = true)
 |-- mcWeight: float (nullable = true)
 |-- pvxp_n: integer (nullable = true)
 |-- vxp_z: float (nullable = true)
 |-- scaleFactor_PILEUP: float (nullable = true)
 |-- scaleFactor_ELE: float (nullable = true)
 |-- scaleFactor_MUON: float (nullable = true)
 |-- scaleFactor_BTAG: float (nullable = true)
 |-- scaleFactor_TRIGGER: float (nullable = true)
 |-- scaleFactor_JVFSF: float (nullable = true)
 |-- scaleFactor_ZVERTEX: float (nullable = true)
 |-- trigE: boolean (nullable = true)
 |-- trigM: boolean (nullable = true)
 |-- passGRL: boolean (nullable = true)
 |-- hasGoodVertex: boolean (nullable = true)
 |-- lep_n: integer (nullable = true)
 |-- lep_eta: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- lep_phi: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- lep_E: array (nullable = true)
 |    |-- element: float (containsNull = true)
```

spark-root internals 1

- Functionality to read files in ROOT File Format
- Extends Spark's Data Source v1 API
- Represents a single ROOT TTree as Dataset[Row]
  - TTree is ROOT's notion of a Dataset
- File based partitioning

spark-root limitations

- Pointers: Anything that requires Run Time Type Information

- Example
  - class Base { /* class body */ };
  - class Derived : public Base { /* class body */ }
  - std::vector<Base*>
  - Type of the element of the vector is not known until you actually start deserialization of the element

spark-root internals 2

- Transforms TTree => IR Schema => Spark Schema
  - IR schema removes anything that is c++ specific

- Optimizations on IR
  - Column Pruning (nested as well on top of https://issues.apache.org/jira/browse/SPARK-4502)

  - Schema clean up
    - *Empty Rows removal*
    - *Flatten out base classes*
    - *Attempt to remove run time types (nested within a column)*

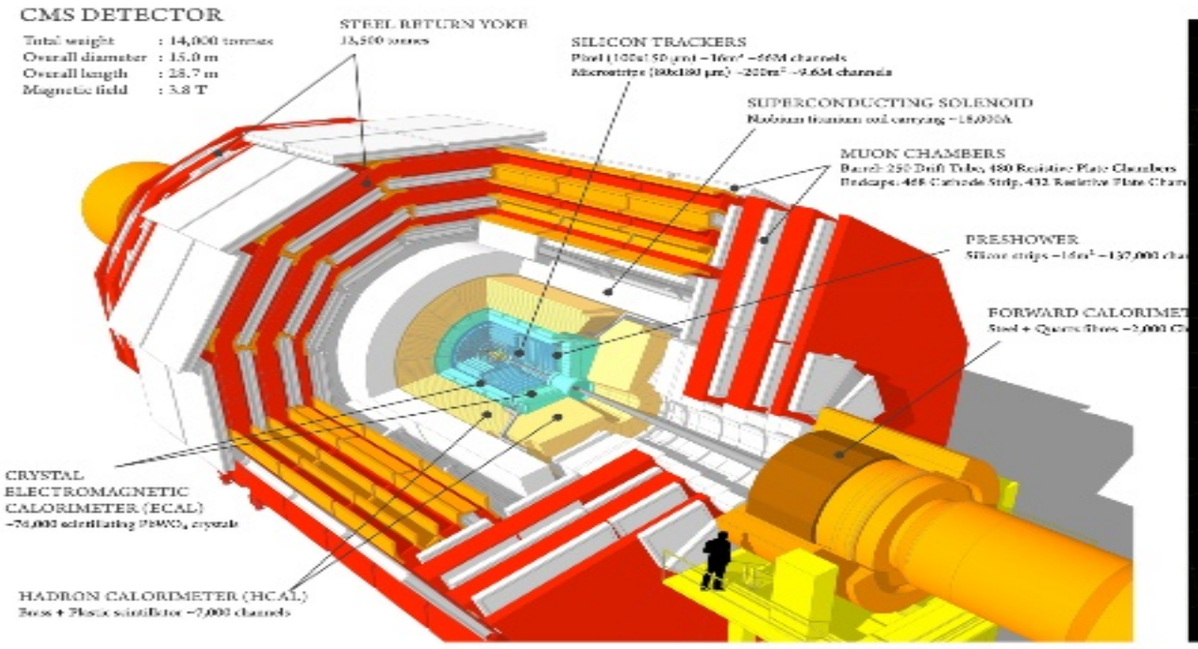
Hello World example

```
// 50K rows of 10K of 8B (double fp)
TFile *f = new TFile(fileName.c_str());
TTree *t = (TTree*)f->Get("TestIO");

// select a column to use
double darr[NUM][NUM];
t->SetBranchAddress("darr", &darr);

// perform a reduction
double totalSum = 0;
for (auto i=0; i<NUM_EVENTS; i++) {
    t->GetEntry(i);
    for (auto ii=0; ii<NUM; ii++)
        for (auto jj=0; jj<NUM; jj++)
            totalSum += darr[ii][jj];
}
```

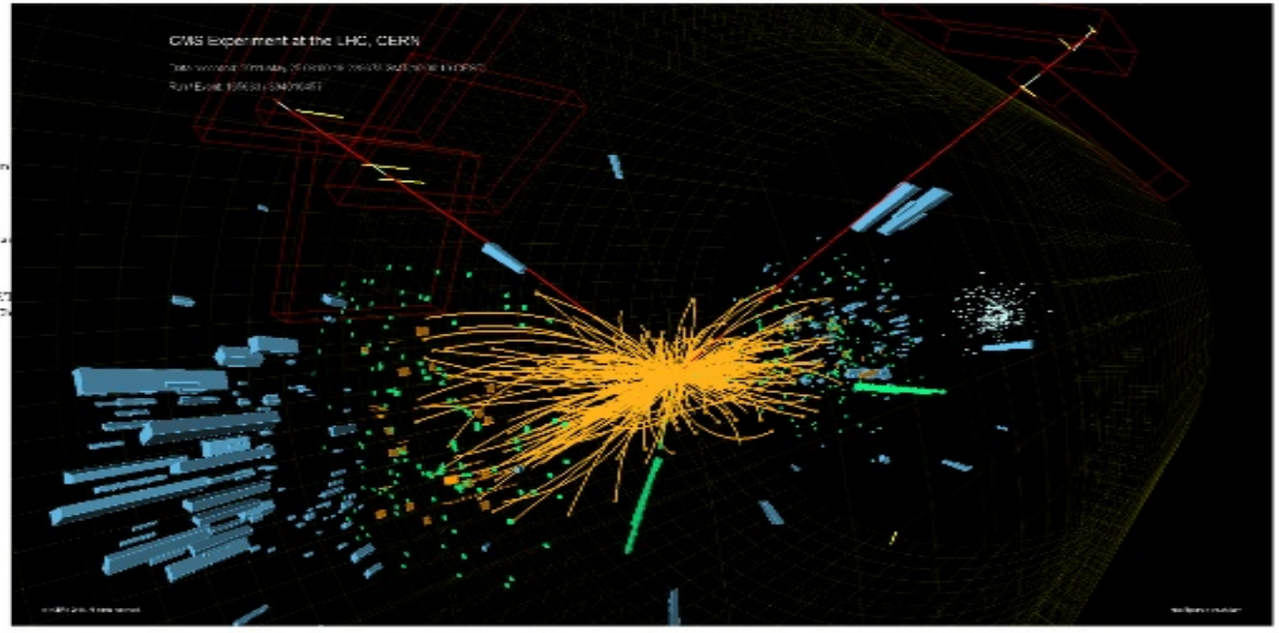
```
// "tree" option specifies the Dataset
val df = spark.read
    .option("tree", "TestIO")
    .root(fileName)

// cast the Dataset[Row] to ...
val ds = df.as[Seq[Seq[Double]]]
ds.flatMap({
    case l => l.flatMap({case v => v})
    }).reduce(_ + _)
```

https://github.com/vkhristenko/test-spark-io

Let's look at some real world example: CMS Experiment



A single LHC event = 1 row to process



CMS Data Analysis

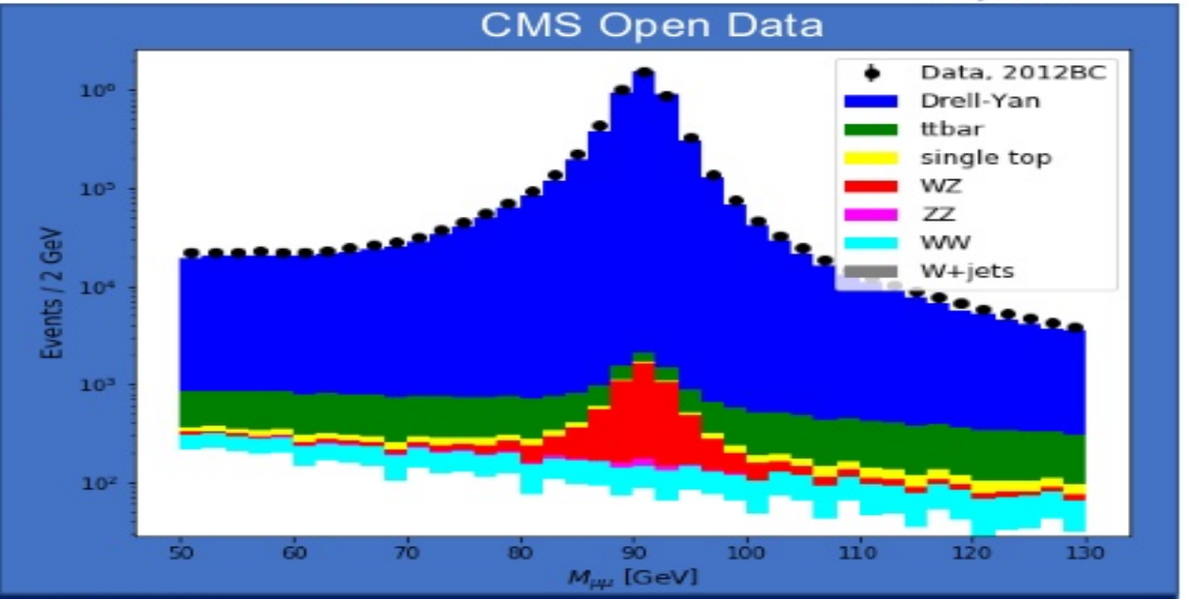
- Compact Muon Solenoid (CMS) Experiment Open Data
- http://opendata.cern.ch/record/32
- 400 top level nested columns
- Deeply nested Data Structures

A glimpse of the schema -1 top level column (shortened)

```
|-- recoMuons_muons__RECO_: struct (nullable = true)
|    |-- present: boolean (nullable = true)
|    |-- recoMuons_muons__RECO_obj: array (nullable = true)
|    |    |-- element: struct (containsNull = 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)
|    |    |    |-- pdgId_: integer (nullable = true)
|    |    |    |-- status_: integer (nullable = true)
|    |    |    |-- innerTrack_: struct (nullable = true)
```

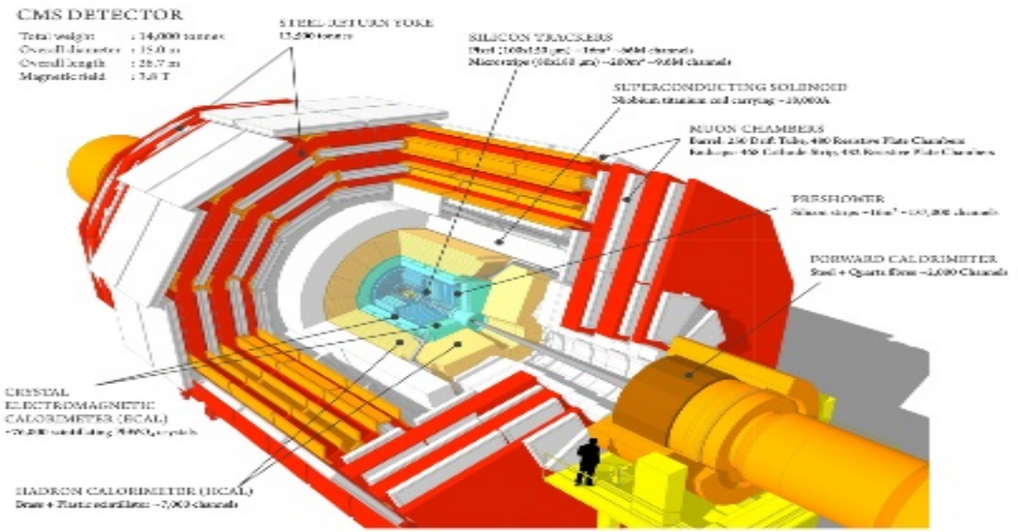
Data Analysis: Following LHC discoveries

```
# read in the data
df = spark.read\
    .format("org.dianahep.sparkroot.experimental")\
    .load("<open data files>")

# select only muons
muonColumn = "<muons>"
muons = df.select(muonColumn)\
    .toDF("muons")

# map each event to an invariant mass
inv_masses = muons.rdd\
    .filter(lambda row: row.muons.size==2)\
    .map(toInvMass)

# Use histogrammar to perform aggregations
empty = histogrammar.Bin(200, 0, 200, lambda row: row.mass_)
h_inv_masses = inv_masses\
    .aggregate(empty,
               histogrammar.increment,
               histogrammar.combine)
```

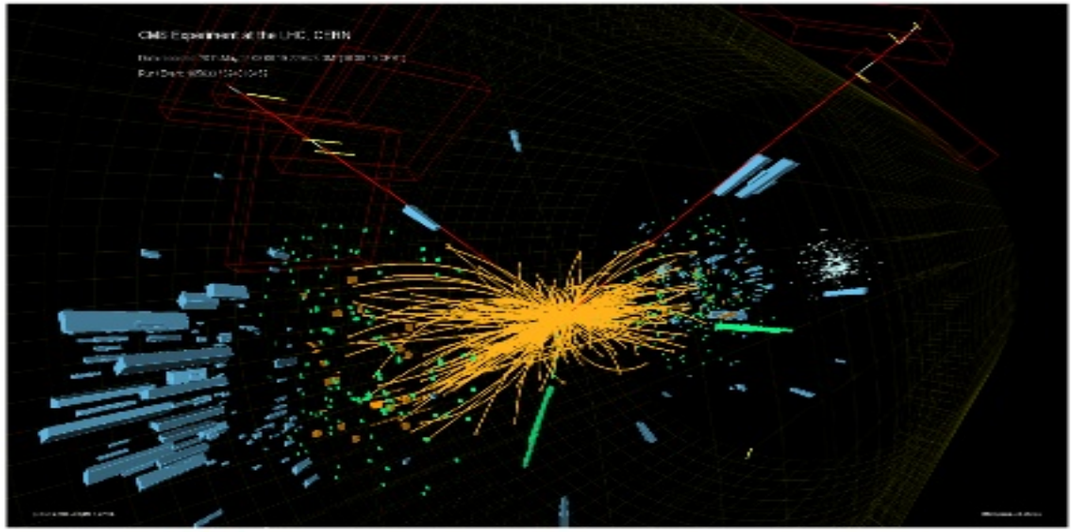


Deep Learning for HEP with Apache Spark

HEP Detector



Detector Readout



Trigger and Identify different High Energy Particle Collisions based on the content of the event (Row).

Deep Learning for HEP with Apache Spark

- ROOT files: several TBs of total input size
- Perform feature engineering
  - Build High level features: a collection of physics quantities per row
  - Build Low level features: 2d matrix of particle properties
  - Build image: convert low level features into 3d matrix (500, 314, 3)
- Perform Training / Inference
- Credits for pipeline also go to @Mmiglio

DL for HEP: Input Schema

A bunch of top level columns representing collections of different particles

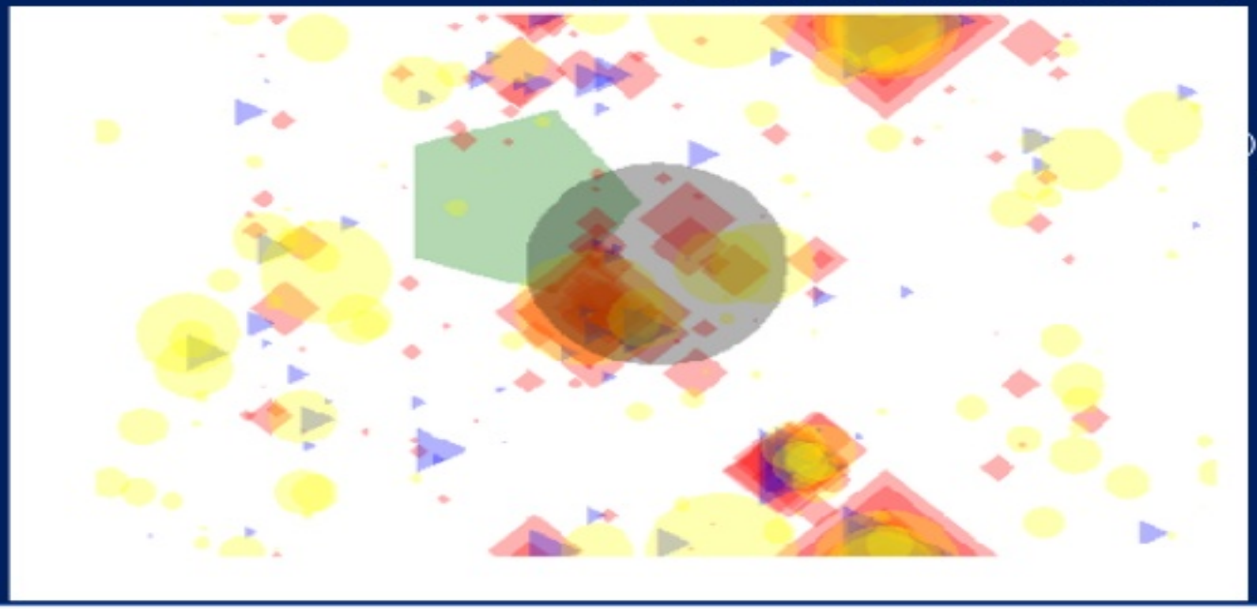
```
|-- EFlowPhoton: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- fUniqueID: integer (nullable = true)
|    |    |-- fBits: integer (nullable = true)
|    |    |-- ET: float (nullable = true)
|    |    |-- Eta: float (nullable = true)
|    |    |-- Phi: float (nullable = true)
|    |    |-- E: float (nullable = true)
|    |    |-- T: float (nullable = true)
|    |    |-- NTimeHits: integer (nullable = true)
|    |    |-- Eem: float (nullable = true)
|    |    |-- Ehad: float (nullable = true)
|    |    |-- Edges: array (nullable = true)
|    |    |    |-- element: float (containsNull =
true)
```

```
|-- EFlowTrack: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- fUniqueID: integer (nullable = true)
|    |    |-- fBits: integer (nullable = true)
|    |    |-- PID: integer (nullable = true)
|    |    |-- Charge: integer (nullable = true)
|    |    |-- PT: float (nullable = true)
|    |    |-- Eta: float (nullable = true)
|    |    |-- Phi: float (nullable = true)
|    |    |-- EtaOuter: float (nullable = true)
|    |    |-- PhiOuter: float (nullable = true)
|    |    |-- X: float (nullable = true)
|    |    |-- Y: float (nullable = true)
|    |    |-- Z: float (nullable = true)
|    |    |-- T: float (nullable = true)
```

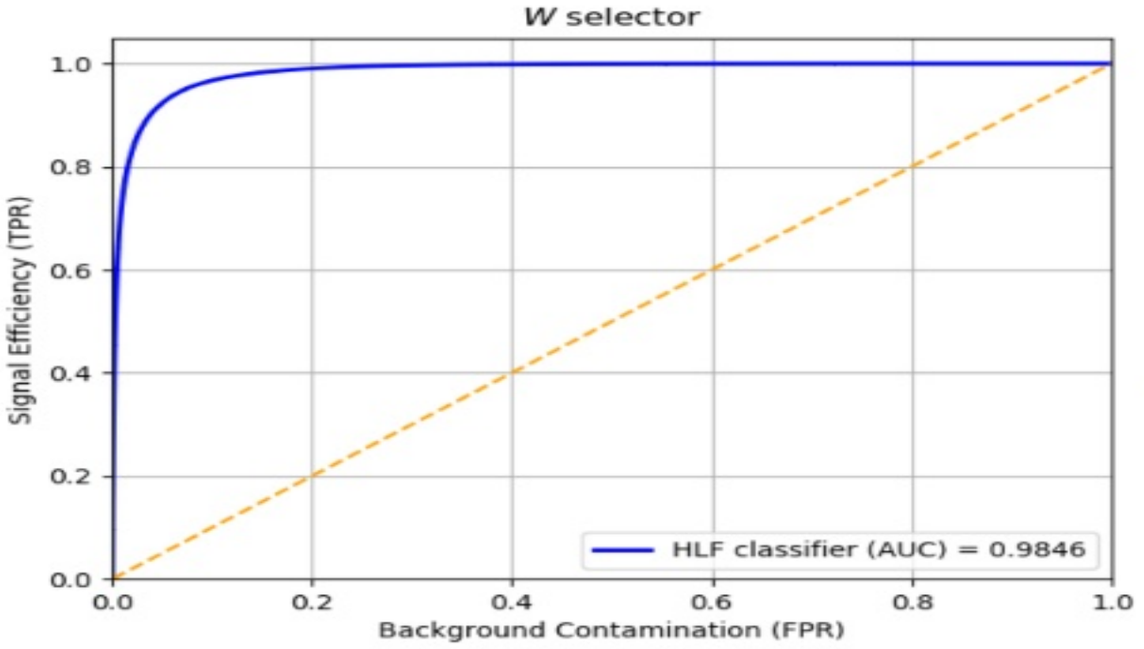
DL for HEP: Data Preparation

```
# read in the data
df = spark.read\
    .format("org.dianahep.sparkroot.experimental")\
    .load("<open data files>")

# select needed columns
requiredColumns = […]
data = df.select(requiredColumns)\
    .toDF(*requiredColumns)

# build high level and low level features
features = data.rdd\
    .map(build_features)

# build an image from low level features
images = features.rdd\
    .map(build_image)

show_image(images.take(1)[0].image)
```

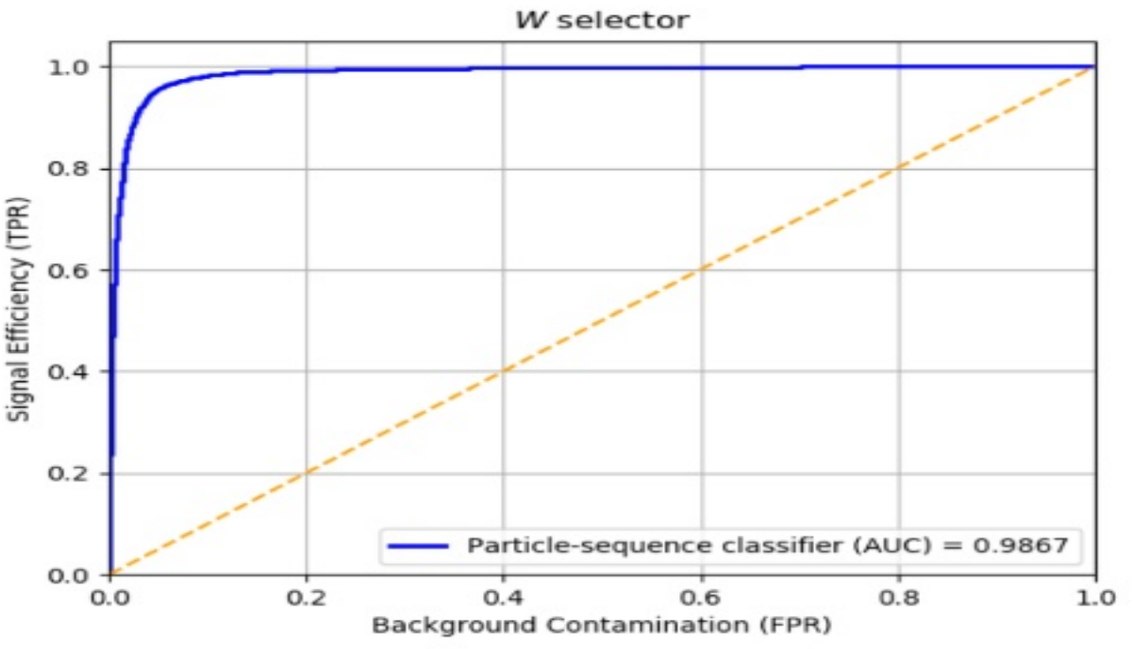
```
root
 |-- hfeatures: array (nullable = true)
 |    |-- element: double (containsNull = true)
 |-- lfeatures: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: double (containsNull = true)
```



DL for HEP: Train/Test Models



Model trained on Particle Properties (High Level Features)



Model trained on images

Summary

- spark-root => a new Data Source for Apache Spark tailored towards High Energy Physics

- Apache Spark works for HEP “pretty much out of the box”

Thoughts / Outlook / Ideas

- Scientific Computing field has a huge software stack
  - Not directly accessible/usable from JVM
  - Difficult to integrate, JNI….

DEEP Projects

The DEEP projects have received funding from the European Union's Seventh Framework Programme (FP7) for research, technological development and demonstration and the Horion2020 (H2020) funding framework under grant agreement no. FP7-ICT-287530 (DEEP), FP7-ICT-610476 (DEEP-ER) and H2020-FETHPC-754304 (DEEP-EST).

Contacts

- viktor.khristenko@cern.ch
- Github: @vkhristenko