Integrating CMSSW in SWAN

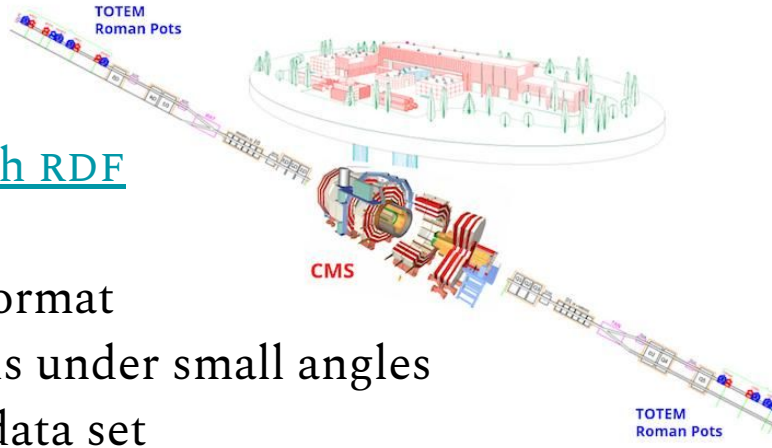
Valentina Avati[1], Karol Remigiusz Bak[1], **Leszek Grzanka**[1], Maciej Malawski[1],
Vincenzo Eduardo Padulano[2], Danilo Piparo[3], Javier Cervantes Villanueva[3]

[1]AGH University of Science and Technology, Krakow, PL
[2]Universita & INFN, Milano-Bicocca, Italy
[3]CERN, Switzerland

Leszek.Grzanka@cern.ch

11.10.2019, 1st SWAN workshop

Project background and goals



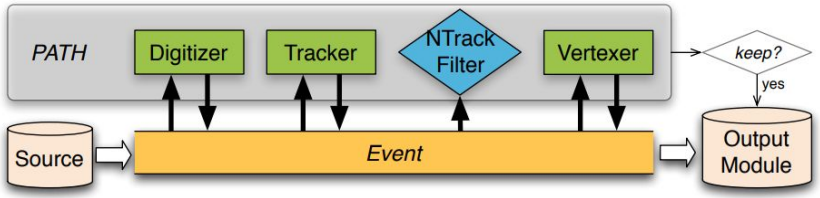
- 2018, summer student project: ↗Analysis with RDF
  - RDataFrame & Apache Spark
  - 4.7 TB preprocessed data in the NTuple format
  - Analysis goal: elastic scattering of protons under small angles
  - promising results, need to test on larger data set

- 2019, extending the scope:
  - RDataFrame & Apache Spark
  - 500 TB in CMSSW RECO format (CTPPSReco, TrackReco, VertexRec)
  - Analysis goal: selection of events, 2 protons with 2 or 4 central tracks

Project goal: investigate feasibility of RDataFrame + Spark in efficient processing of RECO data in the TOTEM experiment.

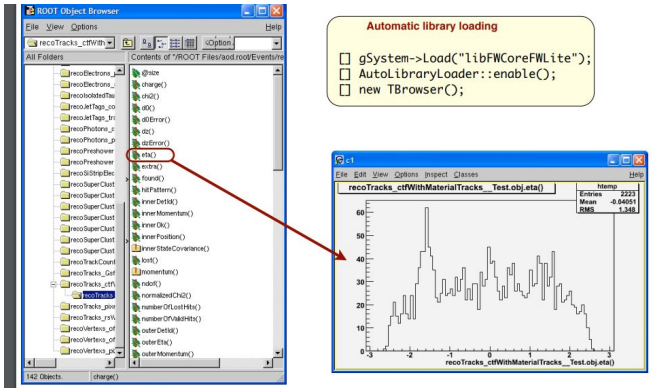
CMSSW and RECO data format

- Analysis use case require detector level data, available in RECO (not in AOD)



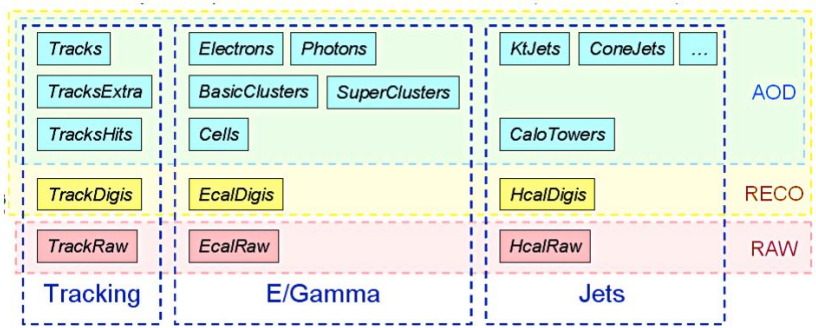
CMSSW framework: event processing using series of modules

- Processing logic: C++ classes + Python config
- cmsRun config.py



CMS Event Data model (EDM):

- Any C++ class serialised, uses ROOT dictionaries
- Data packed in edm:Event container
- Data access via EDM wrapper
- Poor support for reading outside CMSSW



Event data tiers

RDataFrame - introduction

- official ROOT part since v6.14 (June 2018)
- a lot of improvements in ROOT v6.18 (June 2019), development ongoing
- supported in latest CMSSW versions: `11_X` & `10_6_0_pre3`
- scales to many-core architectures
- used by physicists of major LHC experiments
  - ALICE: RDF in the O2 software framework
  - ATLAS: reading xAODs with RDF
  - CMS: real analyses, R&D on reading nanoAODs

↗ RDataFrame in the wild – example real life usages

RDataFrame – general ideas

Read a file, create a custom variable and plot an histogram

Traditional ROOT

```
TFile f(filename);
TTreeReader tree("treename", &f);
TTreeReaderArray<double> px(tree, "px");
TTreeReaderArray<double> py(tree, "py");
TTreeReaderArray<double> E(tree, "E");
TH1F h("pt", "pt", 16, 0, 4);

while (tree.Next())
  for (int i = 0; i < px.GetSize(); i++)
    if (E[i] > 100)
      h.Fill(sqrt(px[i]*px[i] + py[i]*py[i]));

h.Draw();
```

RDataFrame

```
ROOT::EnableImplicitMT();

ROOT::RDataFrame d("treename", filename);

d.Filter("E > 100")
 .Define("good_pt", "sqrt(px*px + py*py)")
 .Histo1D({"pt", "pt", 16, 0, 4}, "good_pt")
 ->Draw();
```

RDataFrame – dealing with CMS data format

Access to RECO data via calls to class methods. Data is not tabular, each event contain variable number of tracks.

So, instead of accessing the parameters directly:

```
ROOT::RDataFrame d(“Events”, filename);
d.Define("pt", "track_pt");
```

We use nested functions (RVec functionality):

```
ROOT::RDataFrame d(“Events”, filename);
d.Define("pt", "return Map(tracks, [](Track tr) { return tr.pt(); });")
```

↗ https://gitlab.cern.ch/kbak/pyrdf-analysis

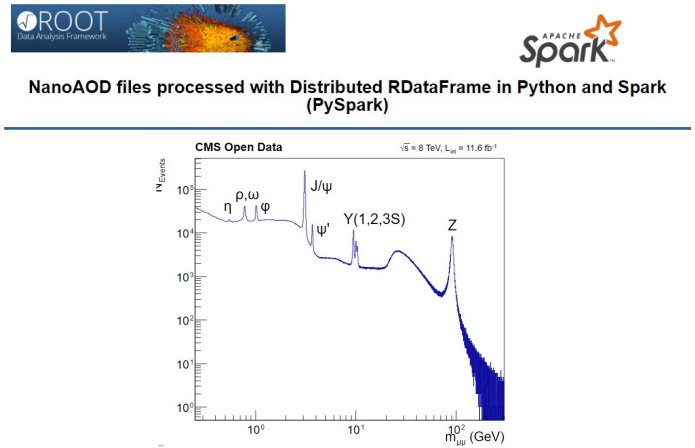
Parallelization options

Multithreading

- Straightforward in CMSSW and in RDataFrame
- Limited by number of available cores

Apache Spark

- Dedicated spark cluster required
- No easy way to parallelize CMSSW analysis on Spark



CERN spark clusters easily available in notebook on SWAN instances at CERN

SWAN offers customization of setup with script

Analysis in CMSSW, using full framework, details

Enable SCRAM build system (`scram` command):

```
bash-4.2$ source /cvmfs/cms.cern.ch/cmsset_default.sh
```

Prepare project area:

```
bash-4.2$ scram project CMSSW CMSSW_11_0_0_pre5 && cd CMSSW_11_0_0_pre5
```

Set libraries (i.e. LD_LIBRARY_PATH), gcc (modifications of PATH), python interpreter + modules (i.e. PYTHONPATH) and ROOT to CMSSW-specific version:

```
bash-4.2$ cmsenv
```

Design and implement C++ analysis code, prepare Python config file and finally run it

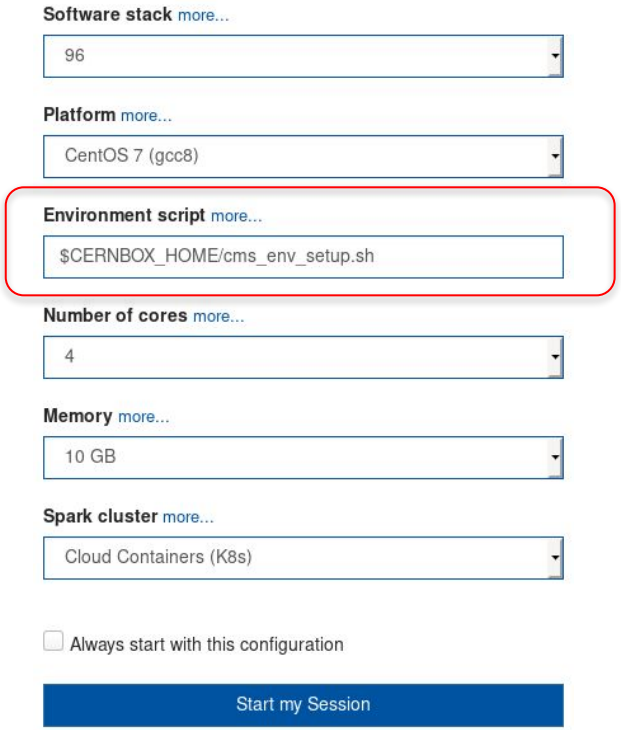
```
bash-4.2$ cmsRun myConfig.py
```

- if we run this in SWAN terminal, it works
- but we need CMSSW libraries inside the notebook
- to be able to run it on Spark cluster (which can be accessed only from SWAN notebook, not terminal)

SWAN - enabling CMSSW environment

Relevant part of what cmsenv command does,
injected as SWAN enviroment script:

```
export LD_LIBRARY_PATH=/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/biglib/slc
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/lib/slc7_amd64_gcc820:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/external/slc7_amd64_gcc820/lib:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/llvm/7.1.0-nmpfii/lib64:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/gcc/8.2.0-pafccj/lib64:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/gcc/8.2.0-pafccj/lib:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/cuda/10.1.105-pafccj2/drivers:\
/usr/lib64/root:$LD_LIBRARY_PATH;
export PATH=/cvmfs/cms.cern.ch/share/overrides/bin:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/bin/slc7_amd64_gcc820:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/external/slc7_amd64_gcc820/bin:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/llvm/7.1.0-nmpfii/bin:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/gcc/8.2.0-pafccj/bin:\
/afs/cern.ch/cms/caf/scripts:\
/cvmfs/cms.cern.ch/common:\
/cvmfs/cms.cern.ch/bin:\
/usr/sue/bin:\
/usr/lib64/qt-3.3/bin:\
/usr/condabin:\
/usr/local/bin:\
/usr/bin:\
/usr/local/sbin:\
/usr/sbin:\
/opt/puppetlabs/bin:\
$PATH;
export PYTHON27PATH=/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/python:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/CMSSW_11_0_0_pre5/lib/slc7_amd64_gcc820:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/coral/CORAL_2_3_21-nmpfii4/slc7_amd64_gcc820/python:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/coral/CORAL_2_3_21-nmpfii4/slc7_amd64_gcc820/lib:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/xrootd/4.9.1-nmpfii/lib/python2.7/site-packages:\
/cvmfs/cms.cern.ch/slc7_amd64_gcc820/external/fastjet/3.3.0-pafccj/lib/python2.7/site-packages:\
```



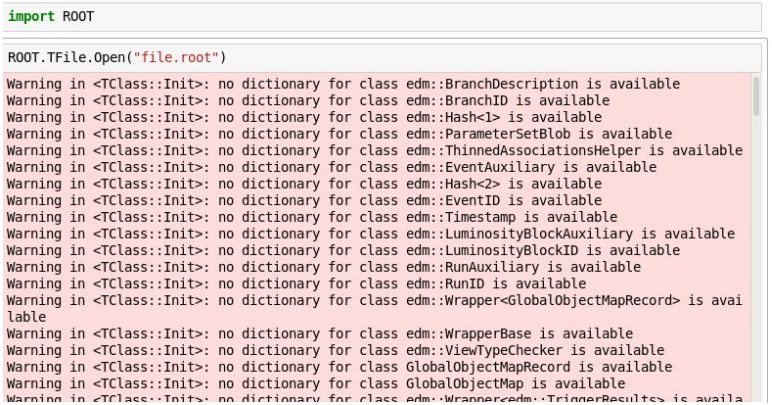
↗gist.github.com/karolBak/a42fa9096efc9fd29f3dff867e479b5f

CMSSW forces a software stack incompatible with SWAN

SWAN with “Software stack: 96”:

```
$ which python
     /cvmfs/sft.cern.ch/lcg/views/LCG_96/
     x86_64-centos7-gcc8-opt/bin/python

$ which root
     /cvmfs/sft.cern.ch/lcg/views/LCG_96/
     x86_64-centos7-gcc8-opt/bin/root
```



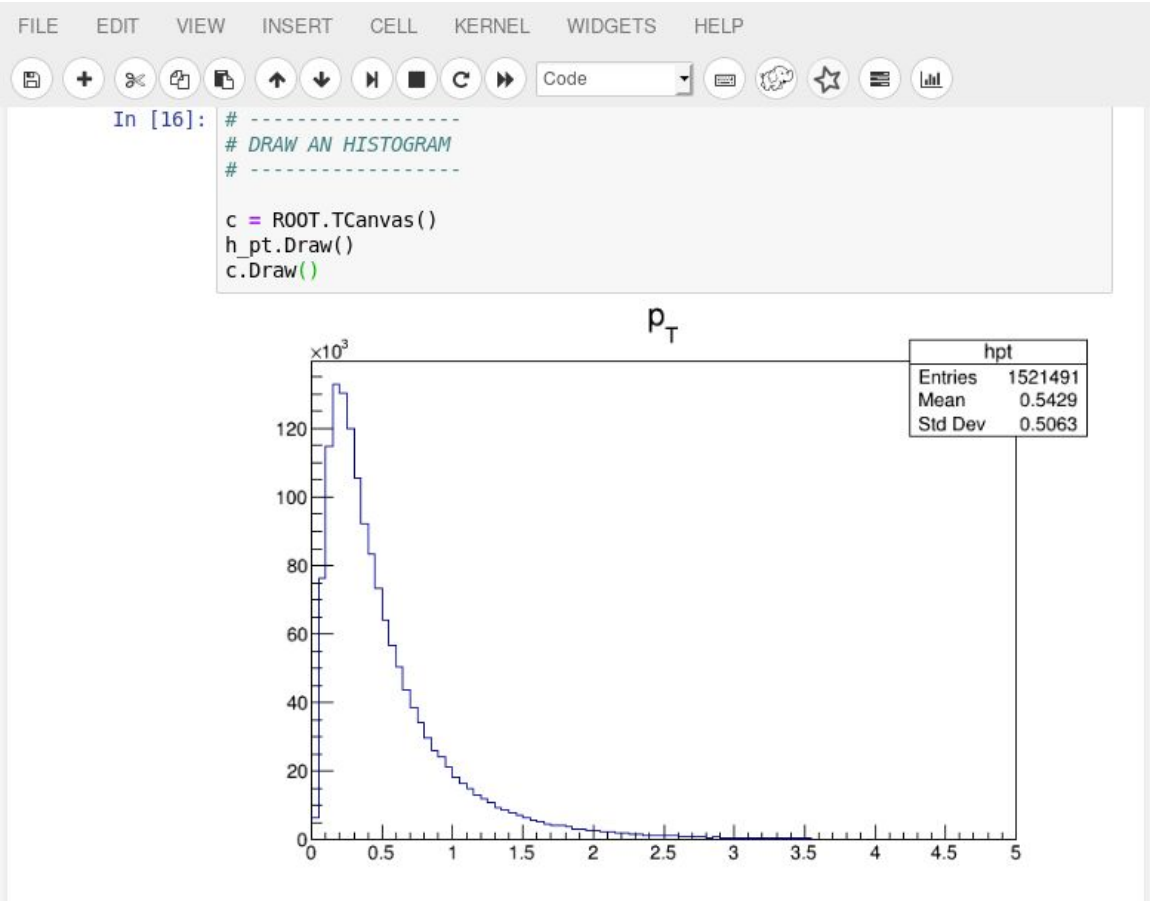
SWAN software stack - missing dictionaries needed to read CMS RECO data

After setting up the paths for CMSSW:

```
$ which python
     /cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/
     CMSSW_11_0_0_pre5/external/slc7_amd64_gcc820/bin/python
$ which root
     /cvmfs/cms.cern.ch/slc7_amd64_gcc820/cms/cmssw/
     CMSSW_11_0_0_pre5/external/slc7_amd64_gcc820/bin/root
```

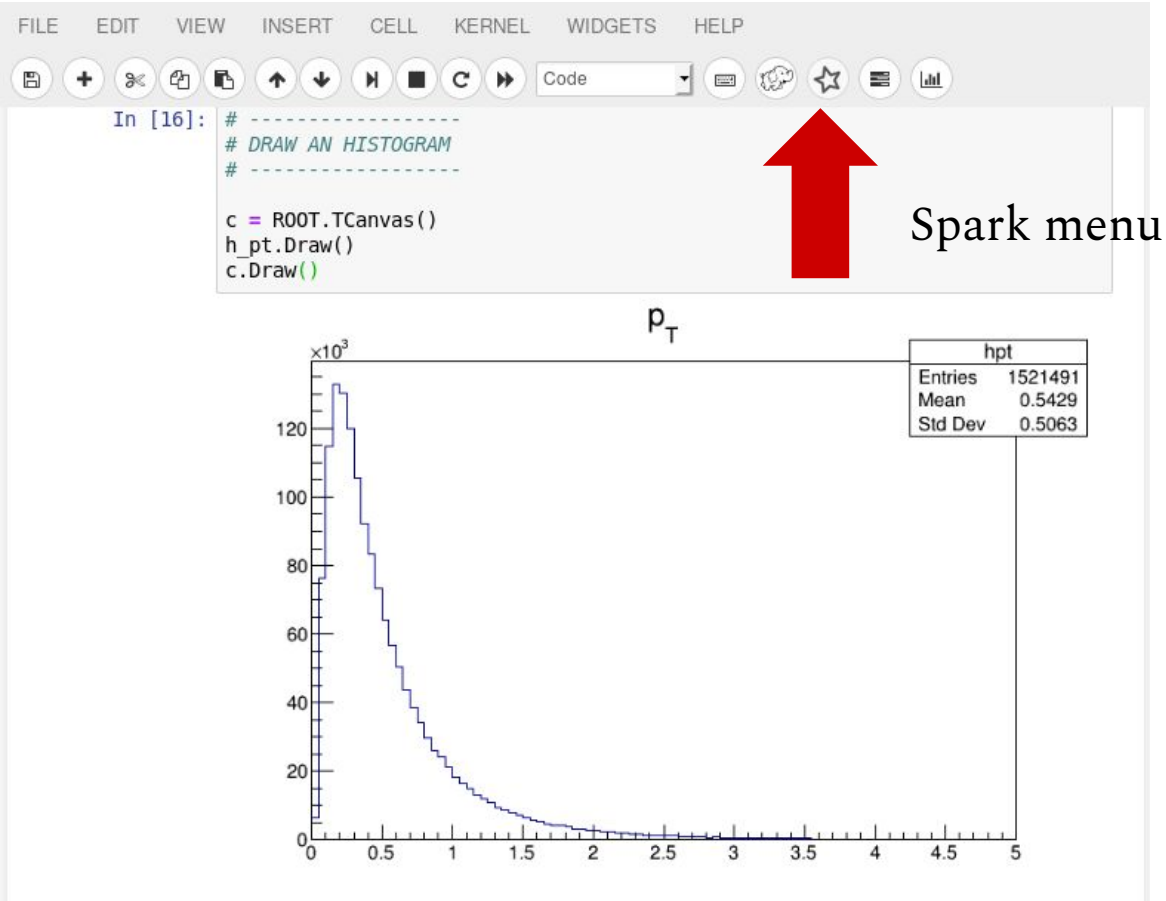
CMSSW software stack - different python and ROOT (deeper: different LLVM, gcc, glibc?), many python packages missing (i.e. spark connectors).

SWAN - multithreaded analysis



Despite incompatibilities, we are able to run the analysis on SWAN instance, not seeing any errors.

SWAN - multithreaded analysis



Spark menu is disabled

Hard to debug anything here, as notebook doesn't throw any errors.

Logs are not available as well.

Conclusions

- integrating CMSSW in SWAN is only possible if considering it as a terminal interface, that is picking only ROOT and the CMS data format libraries via modifying some environment variables
- using the Jupyter notebook and the Spark clusters is incompatible at this moment with the environment needed for the analysis. Such setup would involve picking ROOT from the CMS repository and the Jupyter extensions and Spark framework from the SFT repository

Acknowledgments

This work was partially supported by grant MNiSW DIR/WK/2018/13 by the Polish Ministry of Science and Higher Education