

Basic Physics Analysis Implemented using Apache Spark

Luca Canali

CERN IT Spark and Analytics Service and ATLAS DBA team

PyHEP, September2022



Motivations and Scope

- Context: Spark service at CERN
- Recent work on implementing example analyses using PySpark on Jupyter notebooks
 - Blog: Can High Energy Physics Analysis Profit from Apache Spark APIs?
<https://db-blog.web.cern.ch/node/186>
- I will mix an intro to Spark with notebooks examples
- Final thoughts on what I believe works OK with this approach and what needs improvements
- Not a goal: compete with state-of-the art software for analysis

Apache Spark Adoption

- Who is using Spark, how, and why?

- **Databricks**



They sell a cloud-based analytics platform, centered around Spark

- Development in the Spark ecosystem (Data Lakehouse, MLFlow)
- They also have custom Spark improvements
- Top contributors and drivers of the open-source development
- **Cloud vendors**
 - All offer user-facing “Big Data” platforms, typically including Spark
- Many **tech** giants use it internally
 - Facebook, Apple, MS, Baidu, Netflix, etc
 - Some **contribute** back to Spark development (Apache Spark PMCs)

Spark @ CERN

- Key component of the Hadoop platforms.
 - IT monitoring, IT security
 - Experiments computing data (e.g. reports for the Rucio project)
 - Physics: RDataFrame, CMS Spark, CMS Muon POG
- We also provide Spark on Kubernetes clusters
- User-access
 - **Notebooks**, often via SWAN. Some use of Spark on R
 - Batch jobs: Python, Scala, Java
- NXCals platform
 - Critical and large (~**PBs**) logging system for the accelerator complex
 - Platform based on Hadoop, API based on Spark

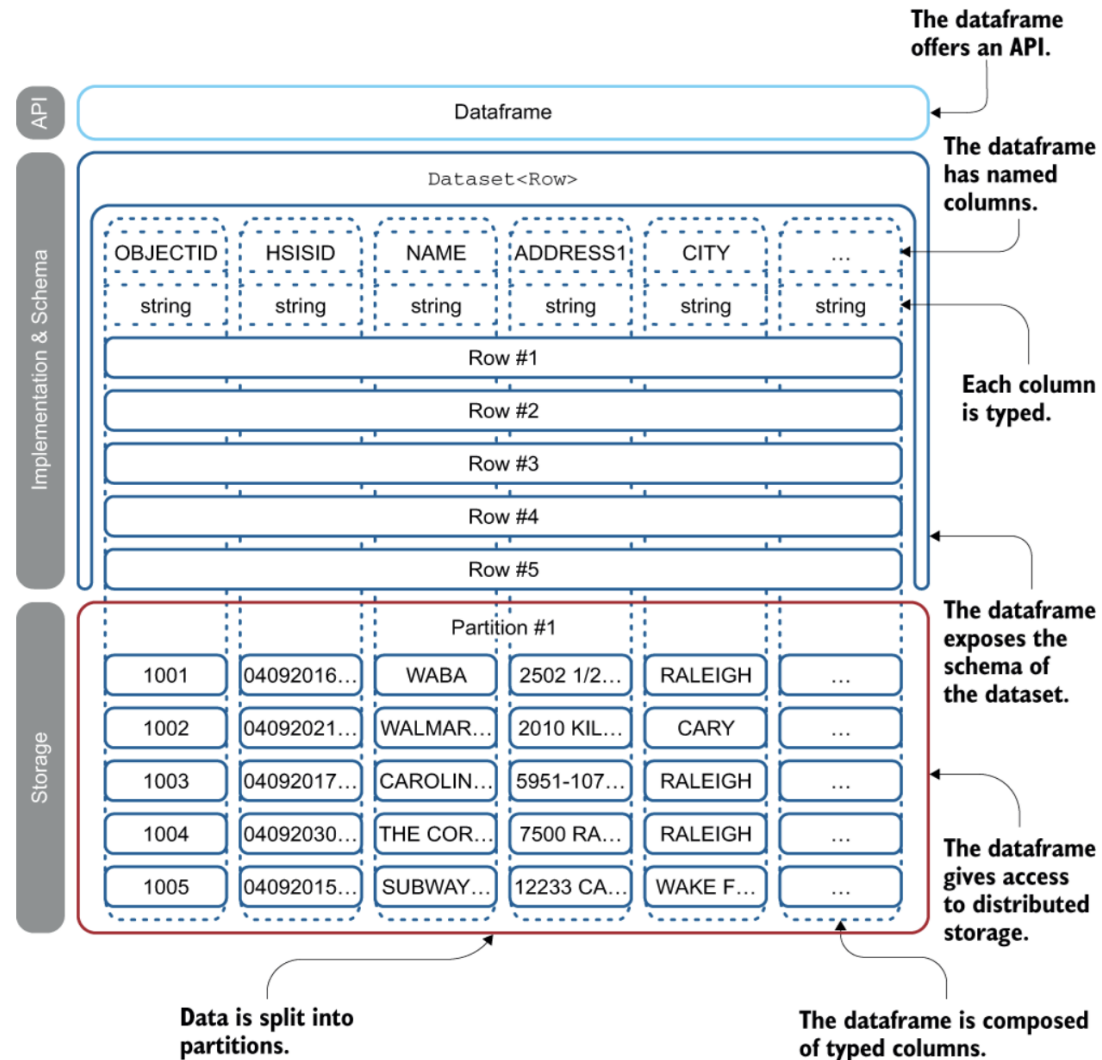
Demo 1

- Can Apache Spark run basic physics analysis?
- Let's start with a "Hello World!" example

- Dimuon mass spectrum analysis at

https://github.com/LucaCanali/Miscellaneous/tree/master/Spark_Physics

Main Data Abstraction: Spark DataFrames



- DataFrame is a table-like abstraction
 - similar to Pandas DF
- Handles data with a schema
- DFs are partitioned and immutable
 - enables parallel execution
 - and fault tolerance at scale

DataFrame API Basics

- Selections and projection are easily scalable
- Filter operations naturally fit with the DataFrame API

```
# Apply filters to the input data  
# - select only events with 2 muons  
# - select only events where the 2 muons have opposite charge  
  
df_muons = df_muons.filter("nMuon == 2").filter("Muon_charge[0] != Muon_charge[1]")
```

- Expressions and formulas

```
df_with_dimuonmass = df_muons.selectExpr("""  
    sqrt(2 * Muon_pt[0] * Muon_pt[1] *  
        ( cosh(Muon_eta[0] - Muon_eta[1]) - cos(Muon_phi[0] - Muon_phi[1]) )  
    ) as Dimuon_mass""")
```

Histograms with the DataFrame API

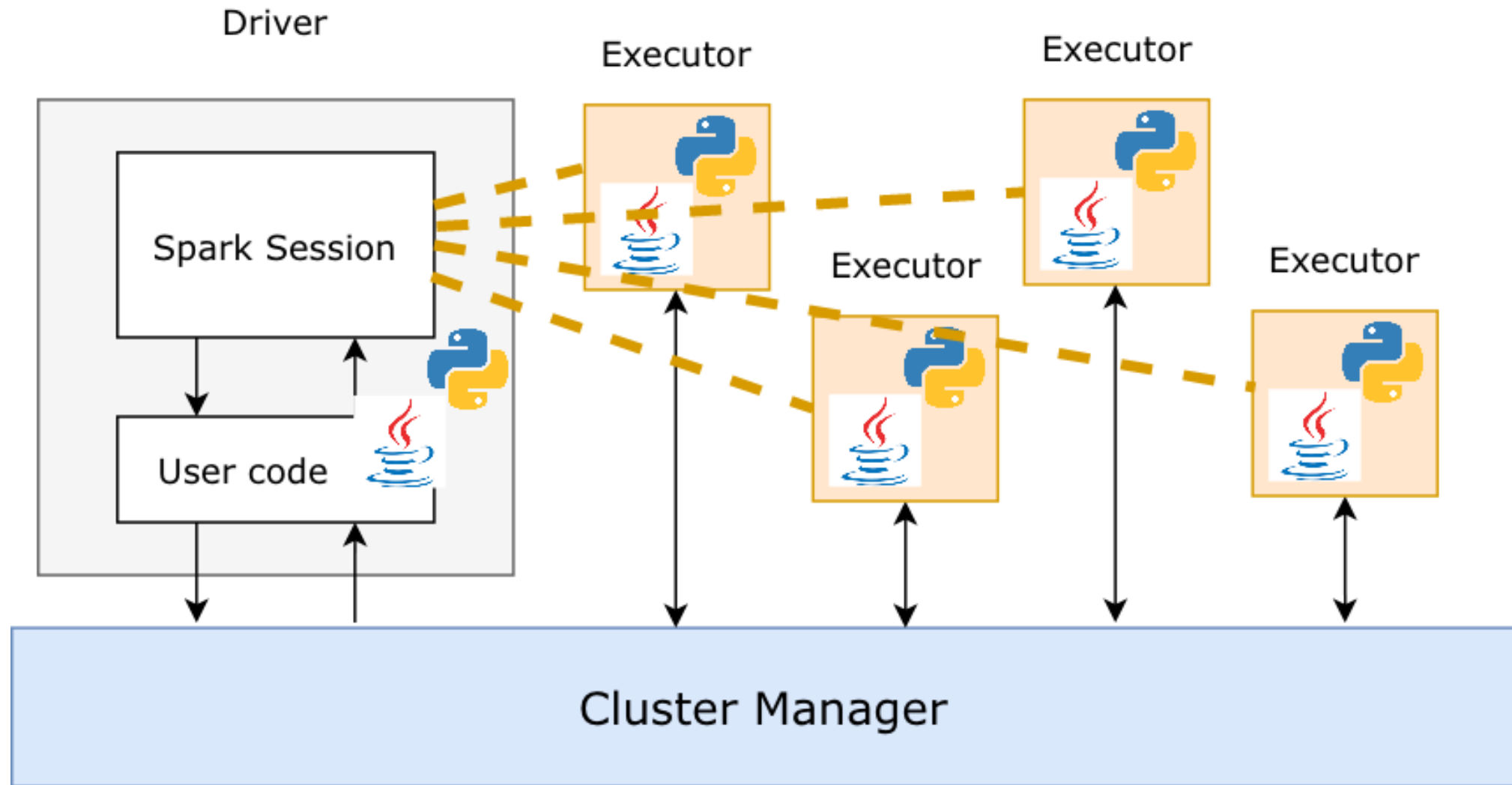
- Histograms are generated using Spark DataFrame API
 - Data aggregation/shuffle with Spark, can run at scale
 - Details hidden in a small custom package: [sparkhistogram](#)

```
from sparkhistogram import computeHistogram

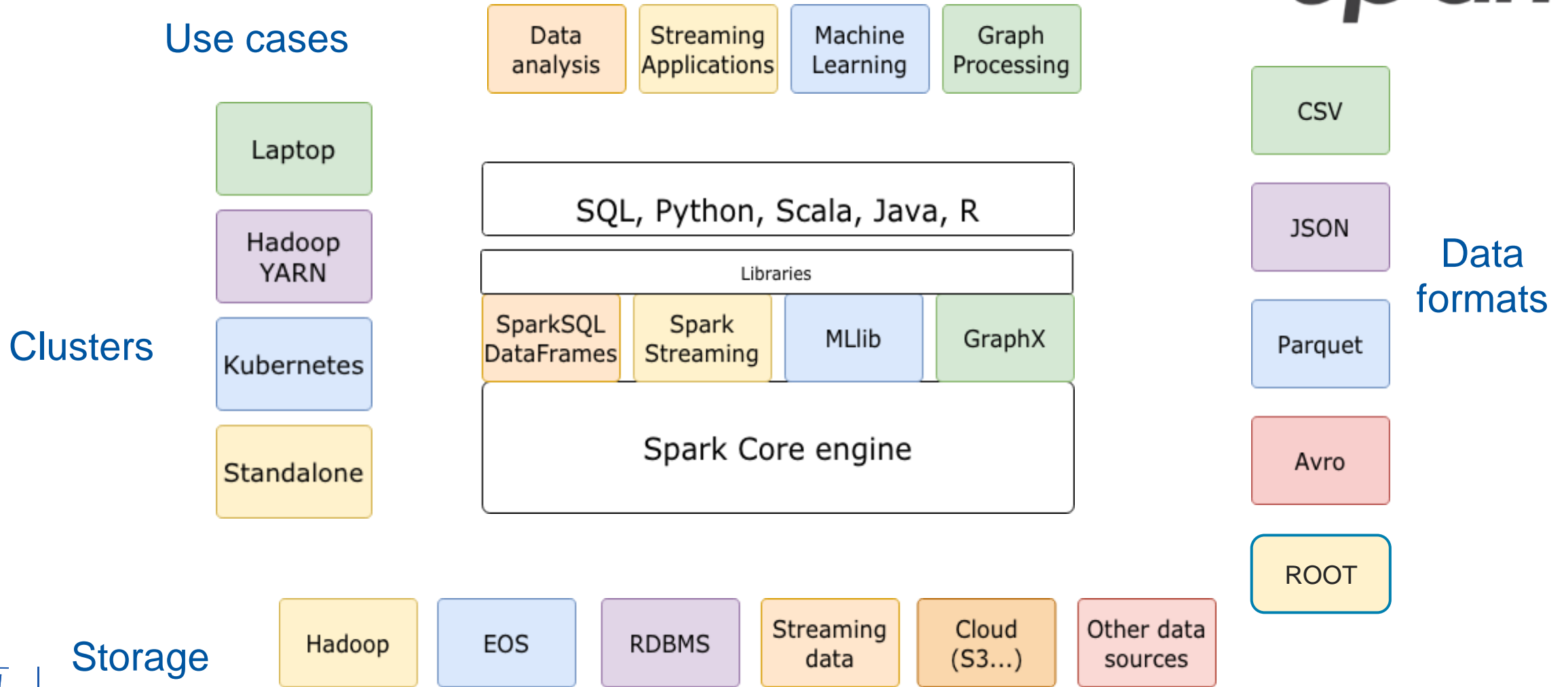
# histogram parameters
min_val = 0.25
max_val = 300
num_bins = 30000

# use the helper function computeHistogram in the package sparkhistogram
histogram_data = computeHistogram(df_with_dimuonmass, "Dimuon_mass", min_val, max_val, num_bins)
```


How Spark Runs Jobs at Scale

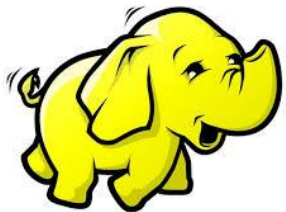


Apache Spark Ecosystem



Apache Spark Clusters at CERN

- Spark running on clusters:
 - **YARN**/Hadoop -> established, use with HDFS storage
 - Spark on **Kubernetes** -> recommended when using EOS or S3 (cloud model)

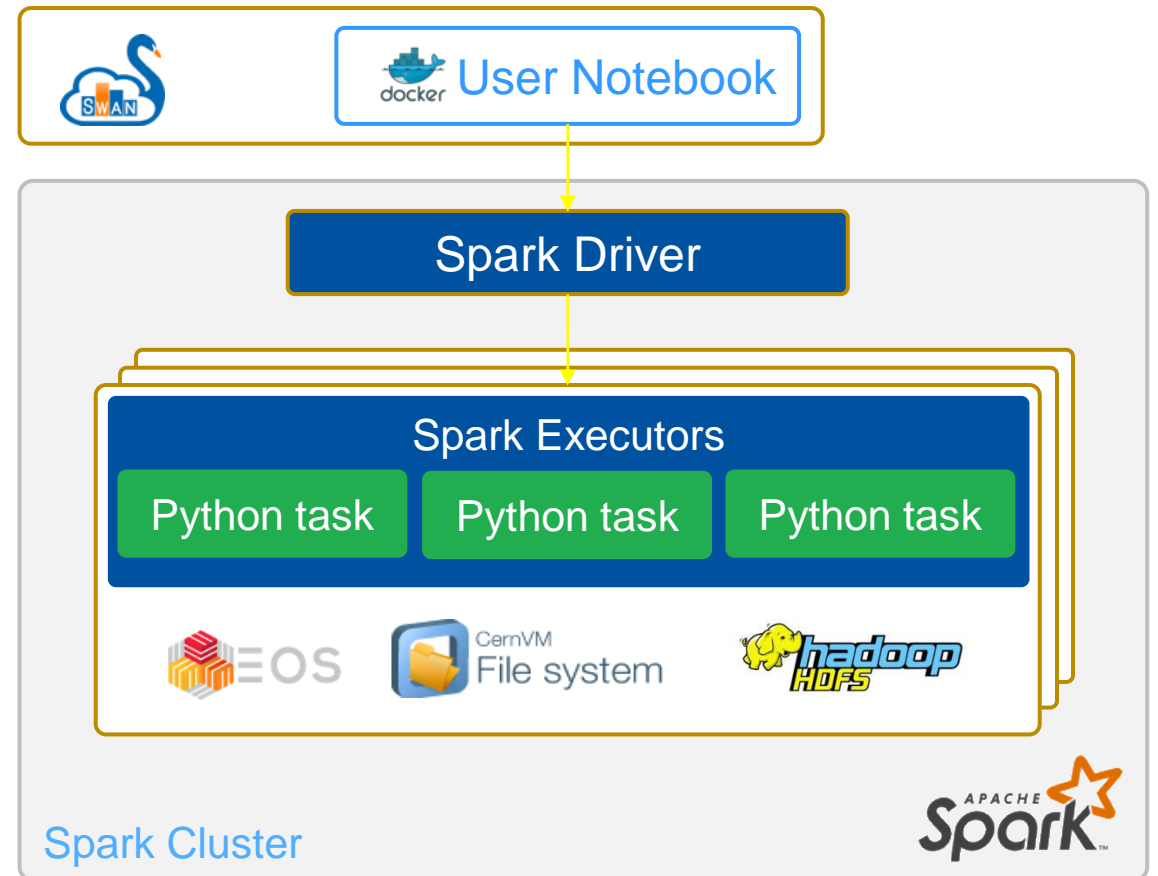


Accelerator logging (part of LHC infrastructure)	Hadoop - YARN - 41 nodes (Cores - 1800, Mem - 18 TB, Storage – 13 PB)
General Purpose	Hadoop - YARN, 58 nodes (Cores – 2.6k, Mem – 30 TB, Storage – 21 PB)
Cloud containers	Kubernetes on Openstack VMs, Cores - 270, Mem – 2 TB Storage: remote HDFS or custom storage (CERN EOS, for physics data, S3 on Ceph also available). Note: GPU resources available.



SWAN Integration with Apache Spark

- SWAN service: <https://swan.web.cern.ch/swan/>
- Notebooks for web-based analysis
- Integrated with CERN Spark Clusters
- Reduces configuration complexity for users
- CERN software environment
- Software from CVMFS
- Graphical Jupyter extensions developed
- Spark Connector
- Spark Monitor
- Access to Spark Clusters
- NXCal: – Dedicated cluster for accelerator logging
- **Analytix**: – General purpose YARN cluster
- **Cloud Containers**: – General purpose Kubernetes cluster
- Storage access: HDFS, **EOS**, S3



Example 2

- Dimuon mass spectrum analysis on a cluster:

[https://github.com/LucaCanali/Miscellaneous/tree/master/Spark Physics](https://github.com/LucaCanali/Miscellaneous/tree/master/Spark_Physics)

Spark can Handle Complex Schemas

- Example of complex schema for HEP

```
schema = "event LONG, HLT struct<flag1:boolean, flag2:boolean>, muons  
ARRAY<STRUCT<pt:FLOAT, eta:FLOAT, phi:FLOAT, mass:FLOAT>>"
```

```
df.printSchema()
```

```
|-- event: long (nullable = true)  
|-- HLT: struct (nullable = true)  
|   |-- flag1: boolean (nullable = true)  
|   |-- flag2: boolean (nullable = true)  
|-- muons: array (nullable = true)  
|   |-- element: struct (containsNull = true)  
|   |   |-- pt: float (nullable = true)  
|   |   |-- eta: float (nullable = true)  
|   |   |-- phi: float (nullable = true)  
|   |   |-- mass: float (nullable = true)
```

Handling Arrays

- Nested data is hard to handle
 - Does not fit naturally to DataFrame and SQL operations
- Available solutions in Spark
 - **Array functions**
 - Several available, example: array_min, array_sort, array_zip, ...
 - “Explode” function
 - Transforms array values into DataFrame rows
 - **Higher order functions**
 - Process map/filter/aggregate on arrays elements
 - **UDF: User-defined functions**
 - General solution.
 - Spark UDF can be in Python or Scala

Demo 3

- HEP analysis benchmark notebooks

[https://github.com/LucaCanali/Miscellaneous/tree/master/Spark Physics](https://github.com/LucaCanali/Miscellaneous/tree/master/Spark%20Physics)

- Thanks to: <https://iris-hep.org/projects/adl-benchmarks-index.html>

Demo 4

- Outreach-style analysis
- Using Spark and Parquet: more familiar tools to data scientists outside HEP
- See example Higgs boson analysis at:
[https://github.com/LucaCanali/Miscellaneous/tree/master/Spark Physics](https://github.com/LucaCanali/Miscellaneous/tree/master/Spark_Physics)



Lessons Learned and Wrap-up

What Spark Can Offer To HEP

- Spark DataFrame API:
 - Provide powerful abstractions and rich language(s)
 - Both for data **preparation** and **analysis**
 - Mature and an **industry** reference
 - Can handle complex schemas
- Run DataFrame locally and at **scale** using distributed computing
 - Runs on clusters and **cloud** (YARN/Hadoop, Kubernetes)
 - HPC and batch: can run Spark stand-alone (requires extra integration work)
- Integration with a large ecosystem
 - **File formats**: Parquet, csv, **ROOT**, ...
 - Storage systems: HDFS, S3, **EOS**, ...
 - External systems: databases, elastic search, streaming, etc

Performance Gap

- Apache Spark **Performance**
 - Identified several areas of improvement for Apache Spark (3.3)
- Python **UDF** performance
 - Sending data to Python workers has been improved but still slow
- Spark functions
 - Higher order functions performance for array processing need improvements
 - More array functions and functions for Lorentz vector processing?
- Spark engine
 - Apache Spark does not (yet) have **vectorized** execution
 - State-of-the-art HEP tools have large parts running native code vs. Spark is currently mostly Scala and Java code running on the **JVM** (JIT compiled)

Open Questions: Data Formats

- **ROOT** data format ingestion is not optimized for Spark
 - It's a hard job with limited resources
 - Kudos to Andrew for Laurelin library + uproot team
- Apache **Parquet** and ORC
 - Optimized reader for Spark, supported by a large community
- The flatter the data the better
 - Data in nanoAOD format already much easier to process with Spark than deeply nested AOD

Plans and Future Work

- Gather feedback
- Develop further examples and benchmarks
 - Trailing on work done with Coffea/Awkward array and with ROOT/RDataFrame
- Piggyback on Apache Spark improvements
 - Spark is still improving quite fast
- Work with **community** (HEP and Spark)
 - We noticed some interest by Apache Spark committers on understanding HEP use cases
 - Occasionally work together, as in the case of arrow UDF, also currently open [SPARK-34265](#) and [SPARK-38098](#)
 - Some **physicists also interested in trying out Spark ?**

References + Acknowledgments

- Machine Learning Pipelines with Modern Big Data Tools for High Energy Physics, Matteo Migliorini, Riccardo Castellotti, Luca Canali, Marco Zanetti, *Comput Softw Big Sci* **4**, 8 (2020).
- Using Big Data Technologies for HEP Analysis, M. Cremonesi *et al.*, EPJ Web of Conferences 214, 06030 (2019)
- Evolution of the Hadoop Platform and Ecosystem for High Energy Physics, Z. Baranowski *et al.*, EPJ Web of Conferences 214, 04058 (2019)
- Big Data Tools and Cloud Services for High Energy Physics Analysis in TOTEM Experiment, V. Avati *et al.*, 2018, Proceeding of: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)
- CMS Analysis and Data Reduction with Apache Spark, O. Gutsche *et al.* 2018 J. Phys.: Conf. Ser.1085 042030
- Evaluating Query Languages and Systems for High-Energy Physics Data, Dan Graur, Ingo Müller, Mason Proffitt, Ghislain Fourny, Gordon T. Watts, Gustavo Alonso, Proc. VLDB Endow., Vol 15, Issue 2 (2021).

- Get started with Spark at CERN: <https://hadoop-user-guide.web.cern.ch/>

- Thanks:
 - to the teams in the IT Hadoop, Spark and streaming service, and in the SWAN service, in particular to Riccardo Castellotti.
 - to the members and contributors to the (now finished) openlab project on data analysis with CMS
 - to Jim Pivarski, Andrew Melo, Lindsey Grey for discussion and their work on Spark arrow integration improvements and Spark-ROOT format integration
 - Lukas Heinrich, Gordon Watts, Ghislain Fourny, Ingo Müller, for discussions.
 - Ruslan Dautkhanov and Hyukjin Kwon from Databricks for their work and support on adding mapInArrow
 - Matteo Migliorini and Marco Zanetti for the work on ML pipelines.