Web-based interactive data analysis for HEP with Spark and ROOT DataFrame

V. E. Padulano, E. Tejedor, J. Cervantes

ROOT
Data Analysis Framework
https://root.cern

Outline

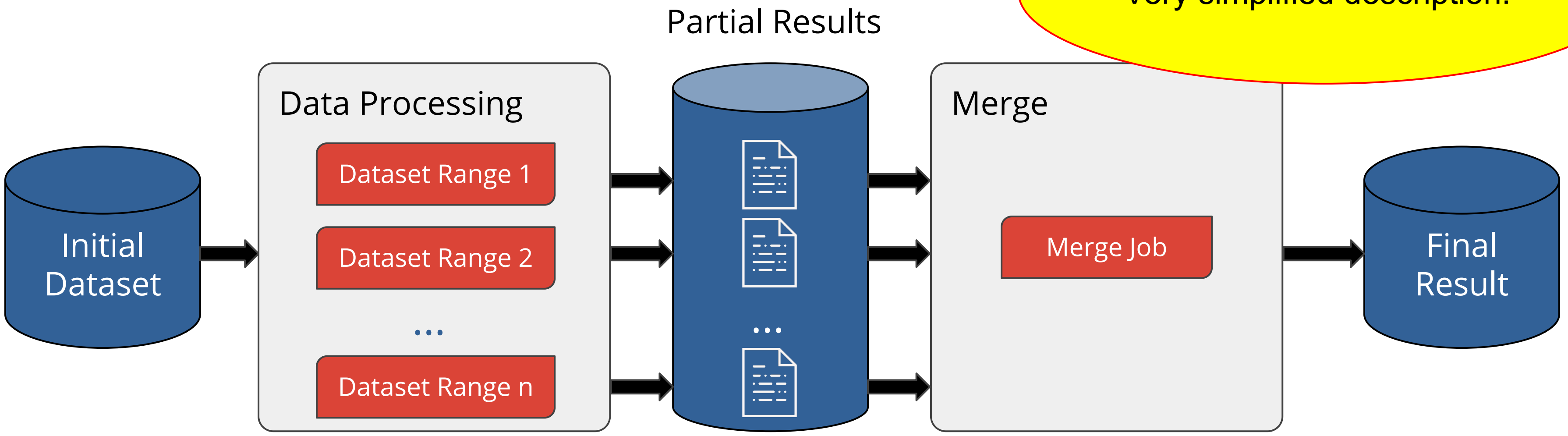
- Motivation
- Spark distributed infrastructure at CERN
- Web-based interactive data analysis
    - SWAN, storage, software, Spark
    - Distributing ROOT analysis
- Use cases
    - Infrastructure data
    - Physics data

Motivation

In a Nutshell: HEP Distributed Computing

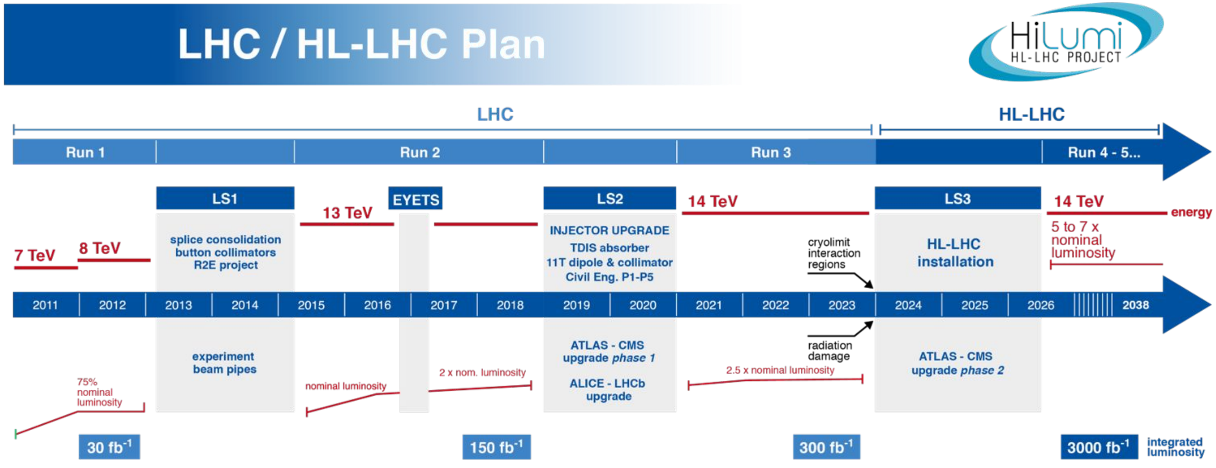
- Parallel processing through batch (local or grid) jobs on statistically independent events
- Merging of partial results happens in a separate stage (more space needed for data and sequential operations)
- Working, but we always strive for improvement

Very simplified description!

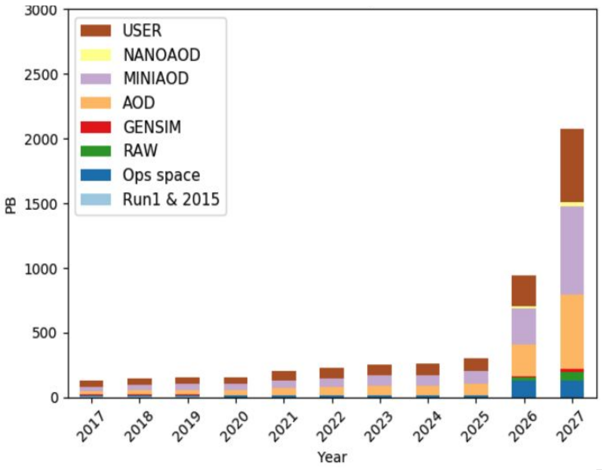


More Data Incoming!

- HL-LHC will bring ~30x more data w.r.t. Run 2
- Automation of processing will be key
- Both hardware and software challenge
  - Currently CMS expects to need significantly more Tape, Disk and CPU by 2027



CMS estimated disk space required by tier



Source : HSF Community WhitePaper and CMS for HOW 2019

Complementing the existing approaches

New Tools

To complement existing approaches, we can make use of new tools, not specific to HEP and backed by large communities that have already proved their potential.

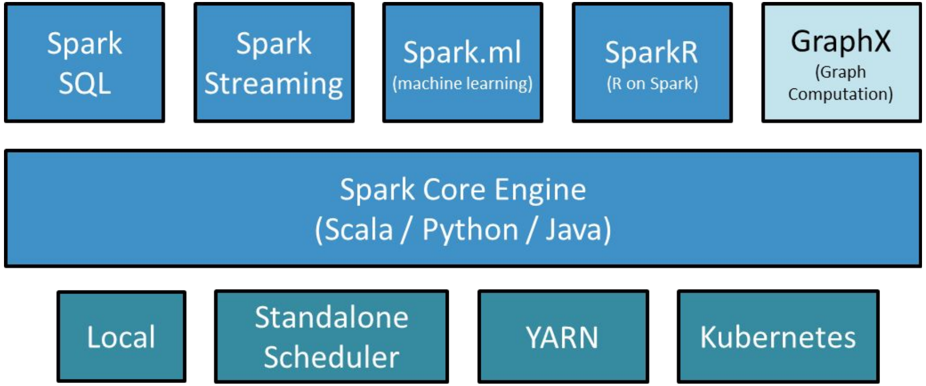
Orchestrated Parallel Computing

Interactive Data Analysis

Spark Distributed Infrastructure at CERN

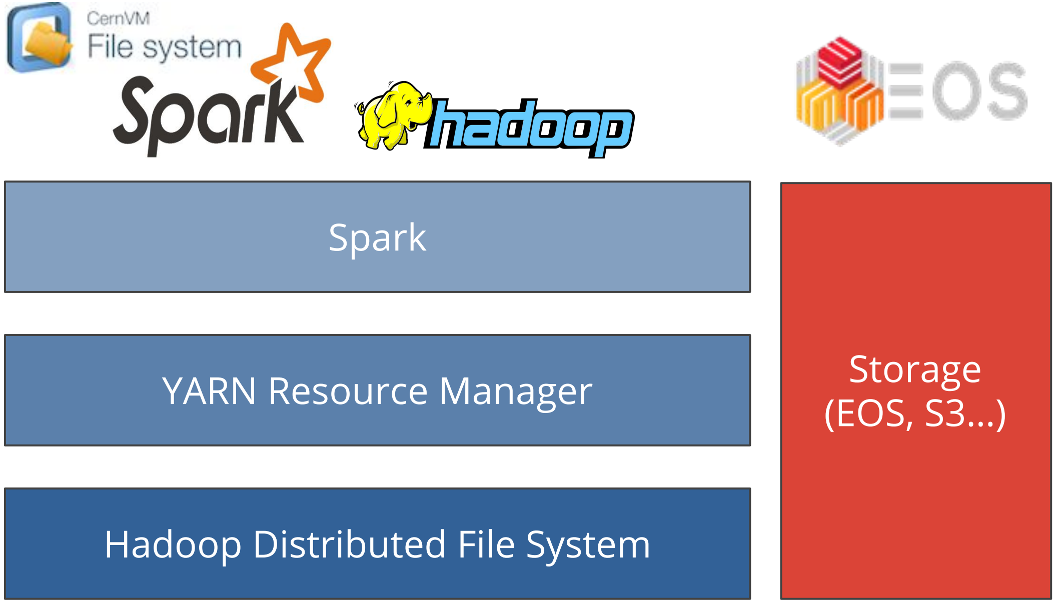
What is Spark?

- Open-source, general-purpose cluster computing system
- High level APIs and interactive execution in Scala, Java, Python
- Offers data management, machine learning and query capabilities
- Runs on multiple cluster frameworks, such as Hadoop, Kubernetes and more



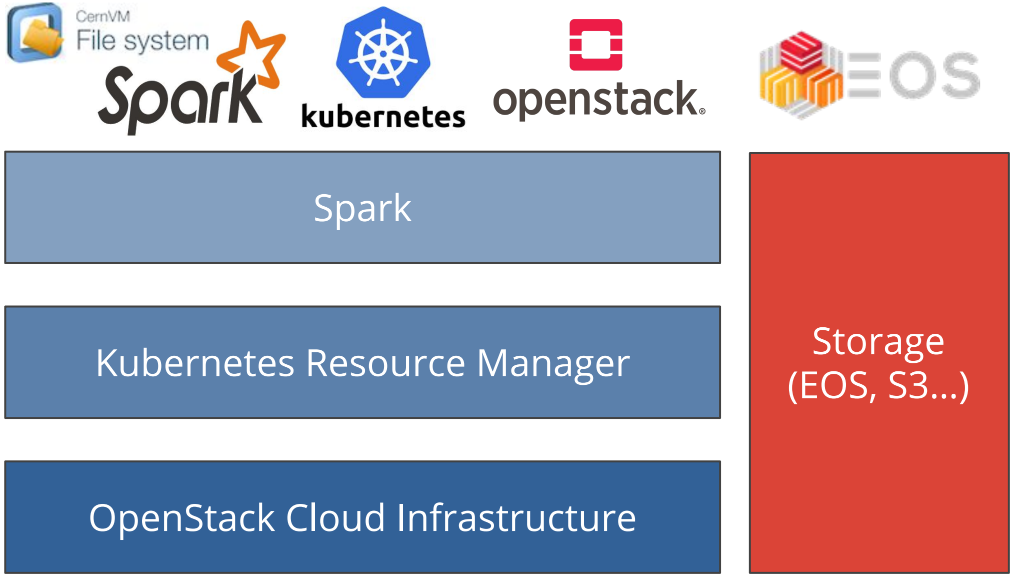
On-Premise Clusters

- CERN clusters managed by IT department
- Spark runs on top of YARN/HDFS
- Data Locality: storage and computation on the same machines
- 4 clusters ~1850 physical cores and 15 PB capacity
  - different configurations based on users' needs

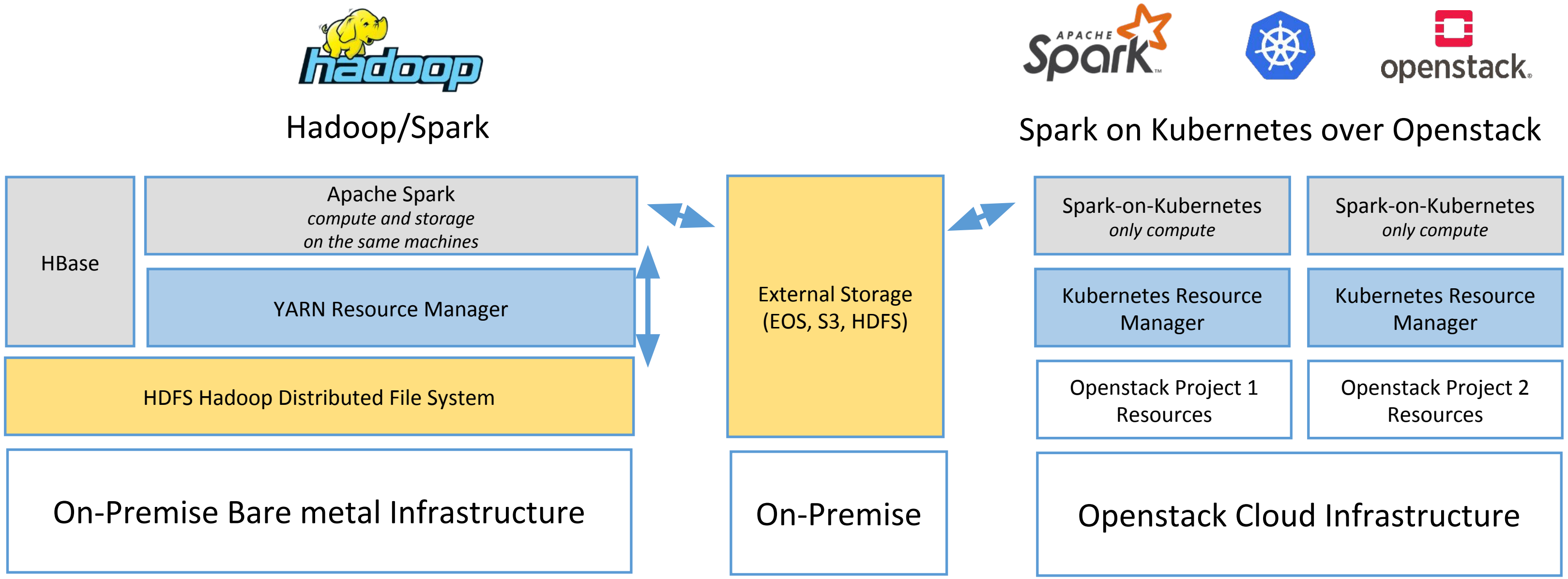


Cloud-Managed Kubernetes Clusters

- Hosted on CERN OpenStack
- Spark runs on cloud VMs
- No persistent storage, data resides in external storage clusters
- Capacities available in production today:
  - 60 VMs
  - 260 Cores
  - 480 Gb Memory
  - + VM local storage



On-Premise vs Cloud-managed



- Stable production workloads
- Data Locality
- No on-demand resource elasticity
- Used if the data resides on HDFS

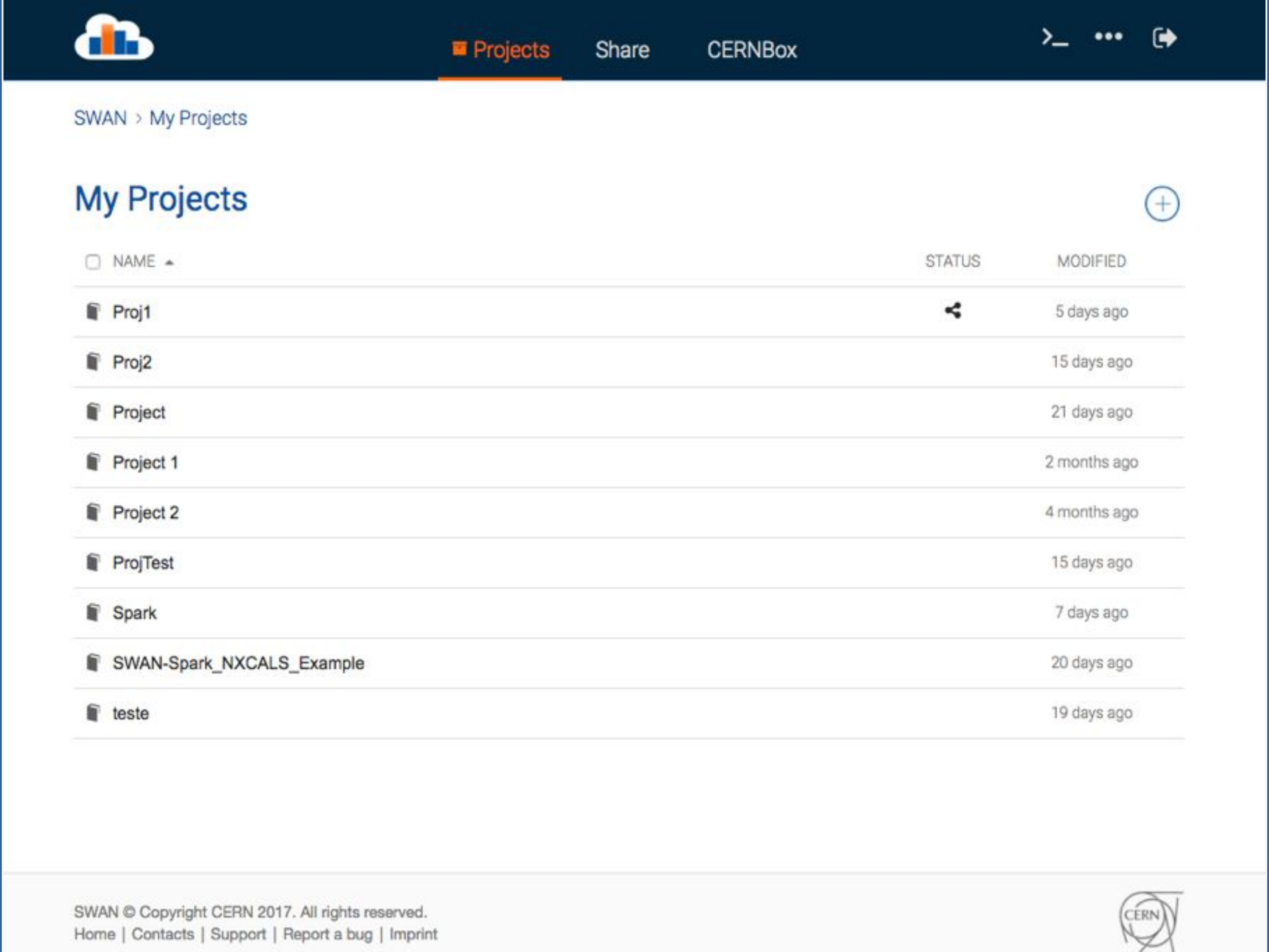
- Cloud-native (rapid resource provisioning)
- Elasticity (Scale out cluster resources)
- Separation of storage and compute
- Recommended for physics analysis, since experiments store data on EOS

Web-based Interactive Data Analysis

SWAN

- SWAN: Service for Web-based Analysis
- Interactive computing platform for scientists
  - Based on Jupyter technology
- Analysis with only a web browser
- Easy sharing of results
- Integrated with CERN resources
  - Storage, software and computing

https://swan.web.cern.ch


Projects
Share
CERNBox
SWAN › My Projects
My Projects
NAME
STATUS
MODIFIED
Proj1
5 days ago
Proj2
15 days ago
Project
21 days ago
Project 1
2 months ago
Project 2
4 months ago
ProjTest
15 days ago
Spark
7 days ago
SWAN-Spark_NXCALS_Example
20 days ago
teste
19 days ago
SWAN © Copyright CERN 2017. All rights reserved.
Home | Contacts | Support | Report a bug | Imprint
CERN

Simple_ROOTbook_cpp.ipynb
(view-only)

Simple ROOTbook (C++)

This simple ROOTbook shows how to create a histogram, fill it and draw it. The language chosen is C++.

In order to activate the interactive visualisation we can use the JSROOT magic:

In [1]:
```
%jsroot on
```

Now we will create a histogram specifying its title and axes titles:

In [2]:
```
TH1F h("myHisto","My Histo;X axis;Y axis",64, -4, 4)
```

```
(TH1F &) Name: myHisto Title: My Histo NbinsX: 64
```

If you are wondering what this output represents, it is what we call a "printed value". The ROOT interpreter can indeed be instructed to "print" according to certain rules instances of a particular class.

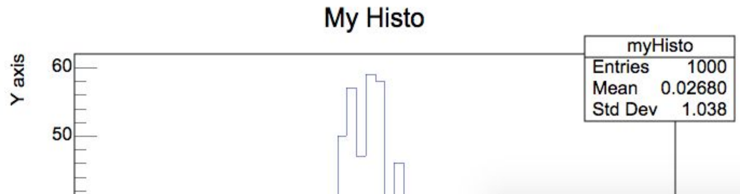
Time to create a random generator and fill our histogram:

In [3]:
```
TRandom3 rndmGenerator;
for (auto i : ROOT::TSeqI(1000)){
    auto rndm = rndmGenerator.Gaus();
    h.Fill(rndm);
}
```

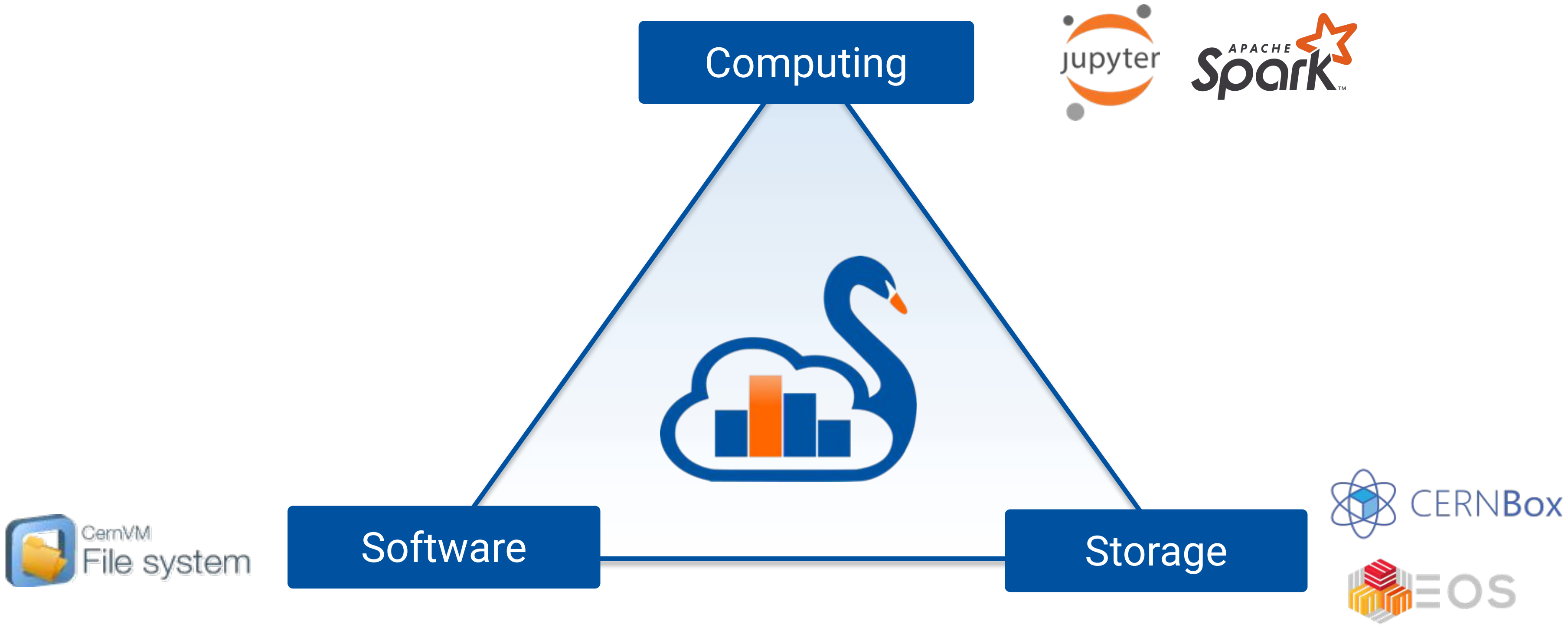
We can now draw the histogram. We will at first create a canvas, the entity which in ROOT holds graphics primitives.

In [4]:
```
TCanvas c;
```

In [5]:
```
h.Draw();
c.Draw();
```



SWAN Pillars



Software Releases: CVMFS

CernVM File system

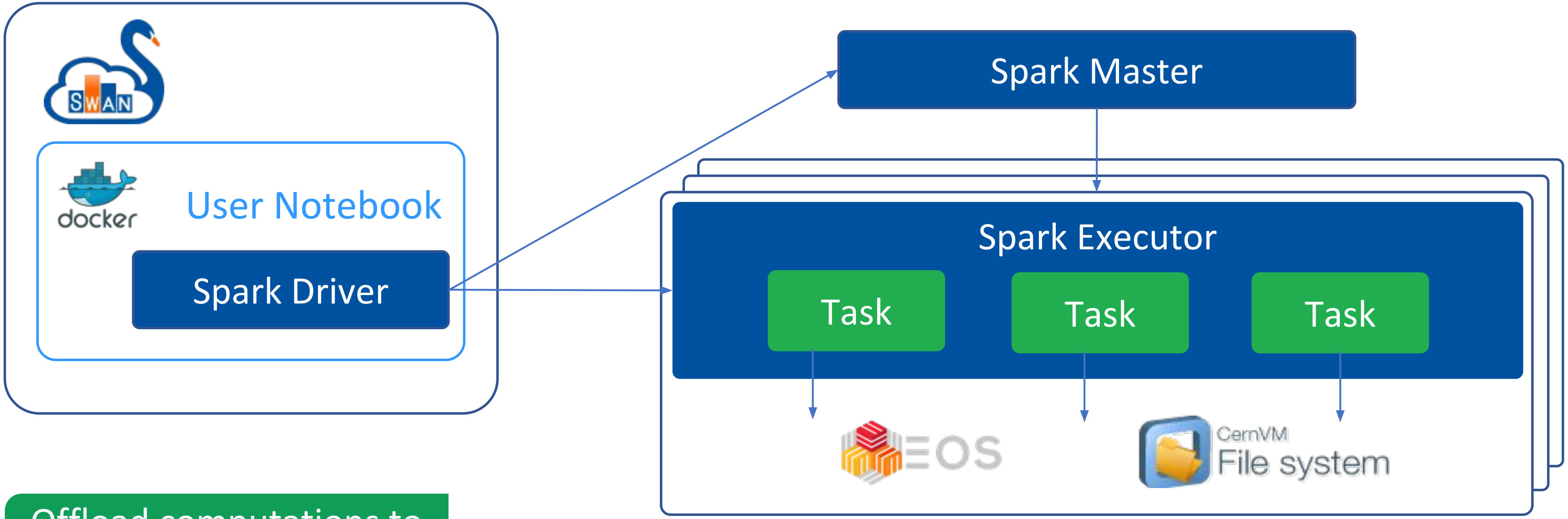
- Software releases for all CERN users
- Designed for distributing small files, fits code needs
- Read-only
- Implements versioning through hashed folders + sqlite meta-data catalogues
- Lazy evaluation: first list files, then download them on-demand
- Aggressively cached at all-levels
- Publisher-subscribers paradigm

User Storage: CERNBox

CERNBox EOS

- Provides cloud data storage to all CERN users
- Based on EOS: the disk-based, low-latency storage service at CERN
- Share data with other users
- Synchronize data across devices
- Up to 1TB personal quota

Integration with Spark

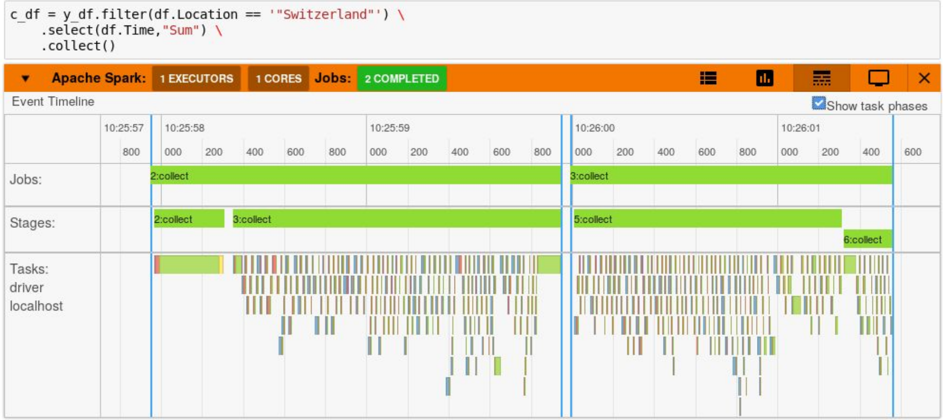


Offload computations to pluggable resources

Spark Cluster

Spark Monitor

- Bridge the gap between interactive computing and distributed data processing
- Automatically appears when a Spark job is submitted from a cell
- Progress bars, task timeline, resource utilisation

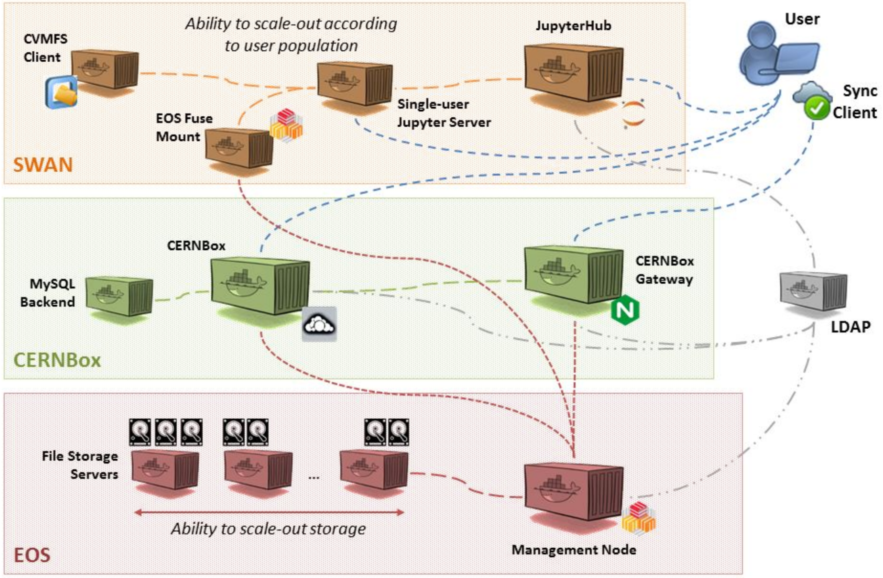


[Code here!]

ScienceBox: Everything in a Box!

- EOS, CERNBox, CVMFS and SWAN together in one place: Science Box.
- Container-based packaging of all these services
- Single-machine demo and scalable deployment with Kubernetes
- Deployable on-premises: have a look here!

ScienceBox distributed infrastructure configuration



ScienceBox: Everything in a Box!

- Self-contained, Docker-based package with:

EOS + CERNBox + SWAN + CernVM File system

docker

docker-compose

kubernetes

One-Click Demo Deployment

- Single-box installation via **docker-compose**
- No configuration required
- Download and run services in 15 minutes

https://github.com/cernbox/uboxed

Production-oriented Deployment

- Container orchestration with **Kubernetes**
- Scale-out storage and computing
- Tolerant to node failure for high-availability

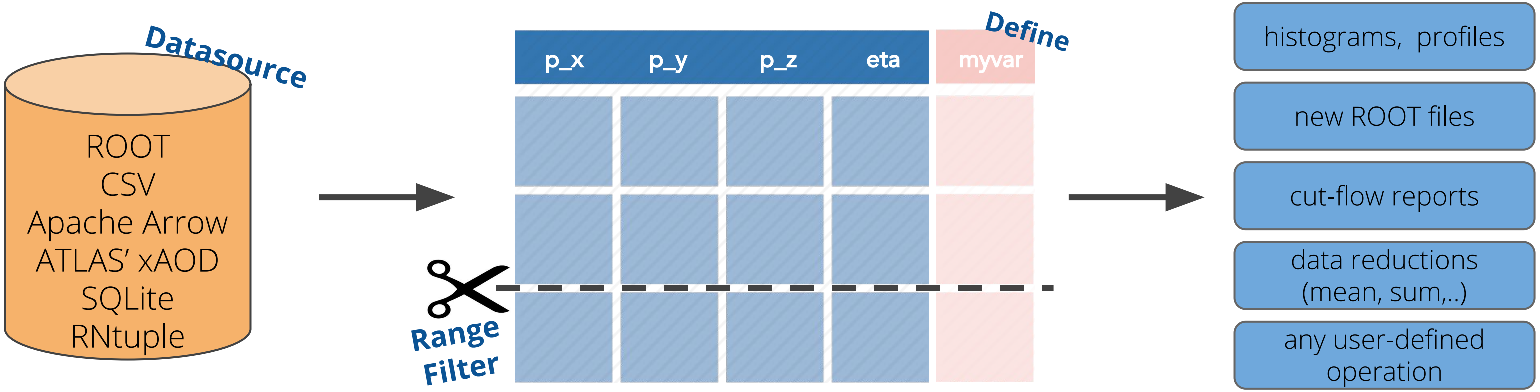
https://github.com/cernbox/kuboxed

E. Bocchi at HEPiX 2019

Distributing ROOT Analysis

ROOT RDataFrame

- Offers high-level declarative API to perform analyses on data
- Multiple data sources
- Columnar data structure
- Consistently supports C++ and Python interfaces
- Implicit optimizations for the chain of transformations and actions performed on data

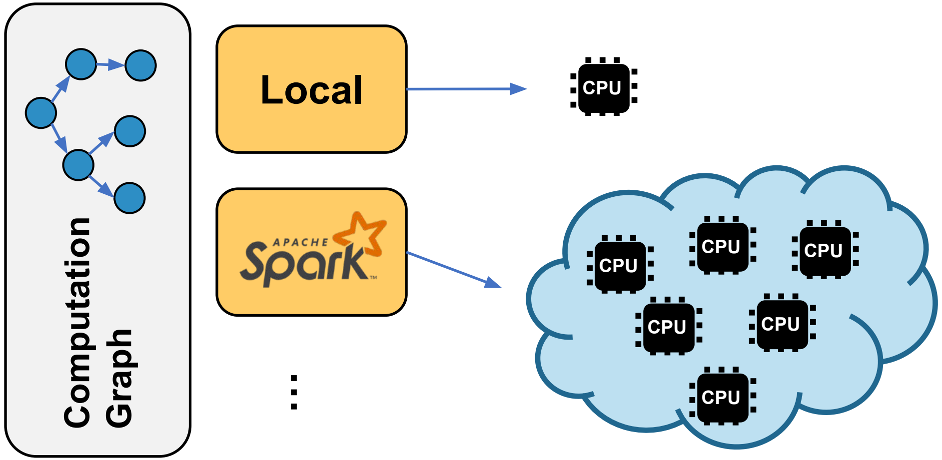


Distributed ROOT RDataFrame

- Creates a DAG from the chain of operations
- Can be distributed to Spark clusters via a map-reduce workflow
- Run analysis in C++ with Spark thanks to the C++ interpreter provided by ROOT

```
d = RDataFrame(dataset)
f = d.Define(...)
     .Define(...)
     .Filter(...)

h1 = f.Histo1D(...)
h2 = f.Histo2D(...)
g   = f.Graph(...)
```



PyRDF

- Python package in development by the ROOT team
- Exploits PyROOT bindings and ROOT RDataFrame DAG
- Exposes a declarative API to users, mirroring the existing ROOT API and adding other features
- Allows local execution (native in RDataFrame) and offload of heavy computation to distributed resources
- Integrated in SWAN (recently added to software releases common to all experiments)

PyROOT at ACAT 2019

CERN HSF Google Summer of Code

PyRDF : The Python ROOT DataFrame Library

build passing

A pythonic wrapper around ROOT's RDataFrame with support for distributed execution.

Sample usage

```
import PyRDF, ROOT
PyRDF.use('spark', {'npartitions':4})

df = PyRDF.RDataFrame("data", ['https://root.cern/files/teaching/CMS_Open_Dataset.root',])

etaCutStr = "fabs(eta1) < 2.3"
df_f = df.Filter(etaCutStr)

df_histogram = df_f.Histo1D("eta1")

canvas = ROOT.TCanvas()
df_histogram.Draw()
canvas.Draw()
```

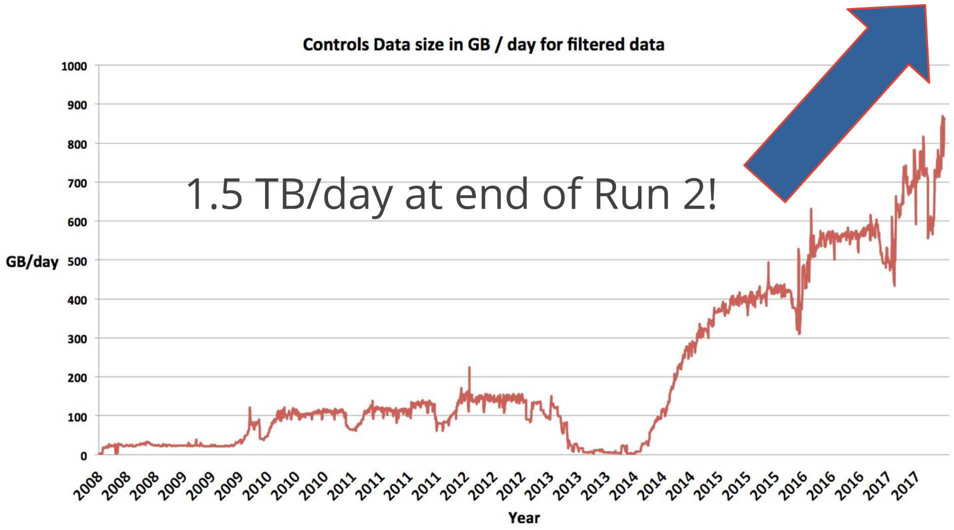
Code, Report

Google Summer of Code

Use Cases

CERN Accelerator Logging Service

- Centralized database queried by control room applications and users
- Built for 1 TB / year throughput
- Exposes a Java API (and a Python wrapper to it)
- Based on SQL DBMS:
  - hard to scale horizontally
  - slow ETL operations
- GUI application called Timber



NXCALS (Next CALS)

- Relies on SWAN as their data analysis platform
- Exposes Java, Python, Scala APIs through Spark
- Connection to Spark clusters
- Better API integration with outside community (Python)
- Stores data in Parquet data format

Inspect data

In [2]:
```
df1.select('acqStamp','voltage_18V','current_18V','device','pt100Value').toPandas()[:5]
```

Out[2]:

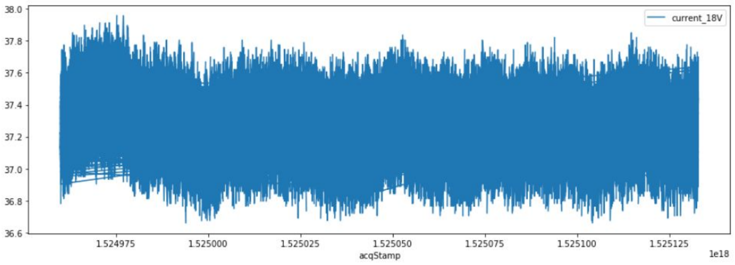
	acqStamp	voltage_18V	current_18V	device	pt100Value
0	1524960103132865000	NaN	37.301794	RADMON.PS-10	106.578911
1	1524960284134584000	NaN	NaN	RADMON.PS-10	107.246742
2	1524960322134942000	NaN	37.560940	RADMON.PS-10	106.504707
3	1524960353135244000	20.099066	NaN	RADMON.PS-10	107.068654
4	1524960911140548000	20.111261	37.698135	RADMON.PS-10	106.578911

Draw a plot with matplotlib

In [3]:
```
import matplotlib
import pandas as pd
%matplotlib inline
```

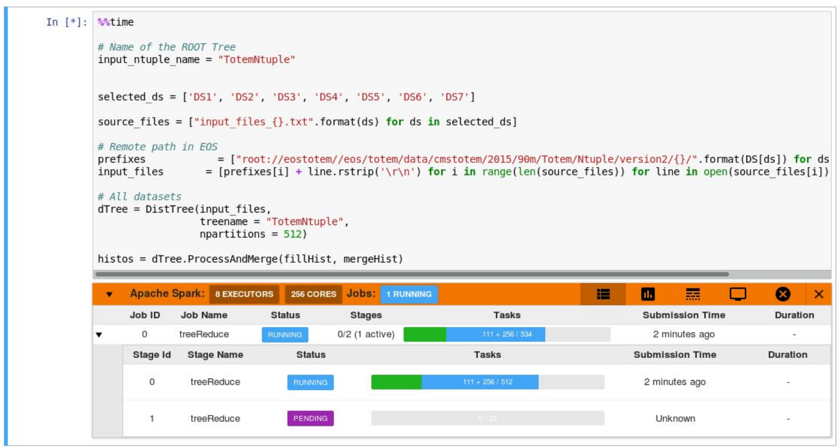
In [4]:
```
p_df = df1.select('acqStamp','current_18V').toPandas()
p_df.plot('acqStamp','current_18V',figsize=(15,5))
# p_df.sort_values(by='acqStamp').plot(pd.to_datetime(p_df['acqStamp'],unit='ns'),'current_18V',figsize=(15,5))
```

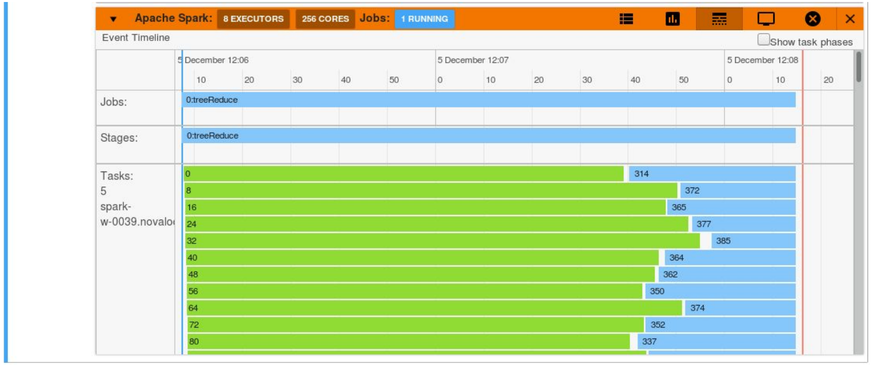
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd8fa2bcc50>



Example real workload: TOTEM

- TOTEM experiment analysis converted to a declarative approach using ROOT RDataFrame
  - Real physics analysis that led to a thesis at CERN (ref.)
- Distributed to Spark clusters with SWAN
- Map-reduce jobs monitored in real-time on the jupyter notebook
  - Spark monitor helped to find performance issues and optimize the workload

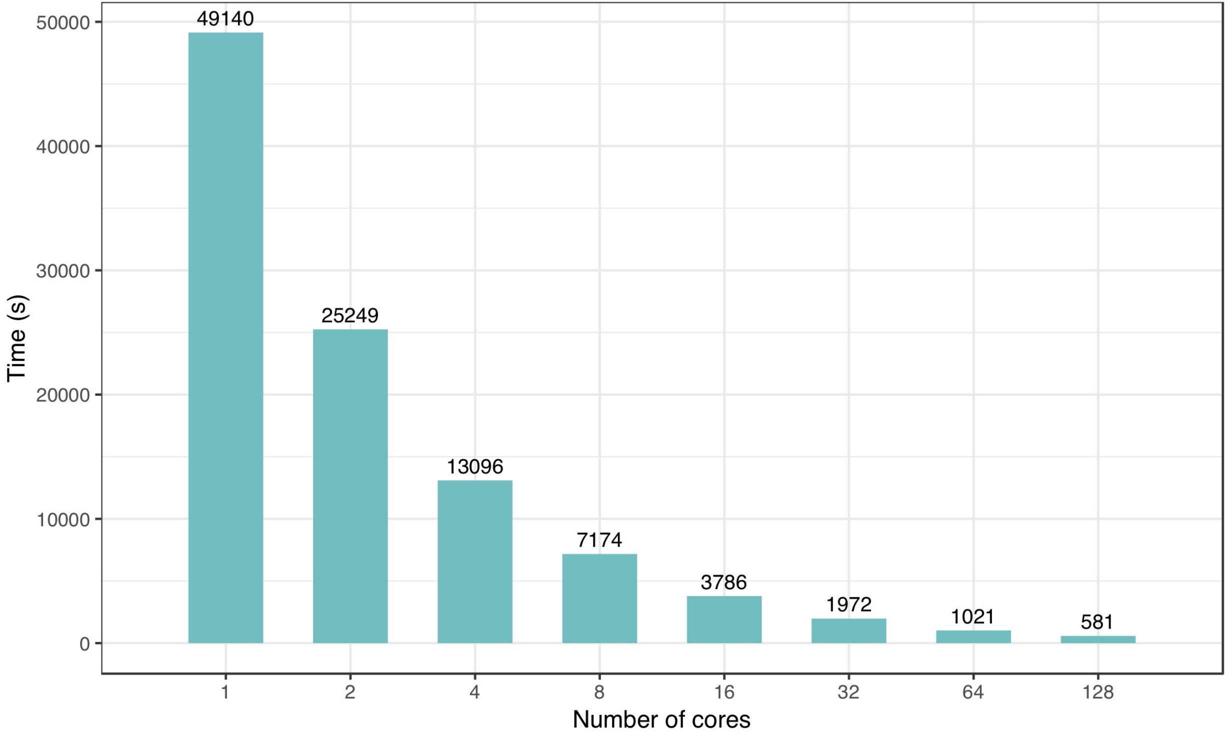




Distributed Execution

Test setup

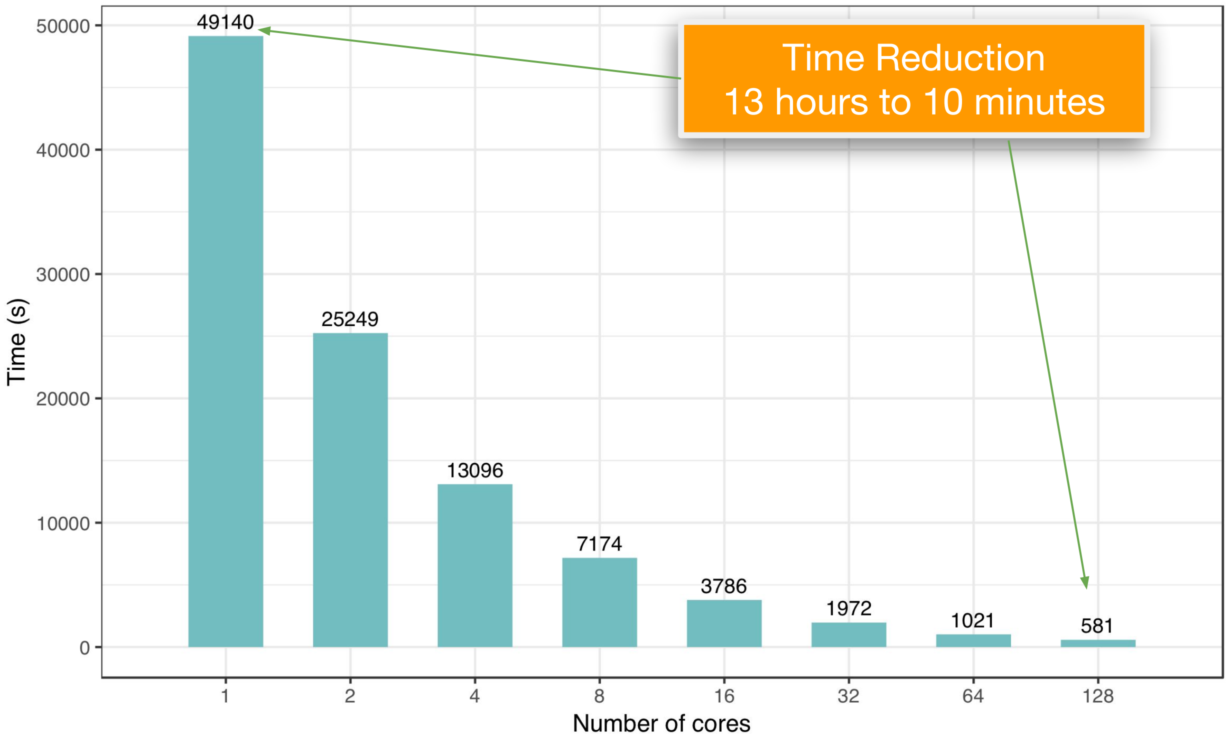
- Data
  - 4.7 TB TOTEM Dataset
- Cluster
  - 15 workers
  - 16 cores/worker
- Requirements
  - Data access (EOS)
  - Software (CVMFS)



Distributed Execution

Test setup

- Data
  - 4.7 TB TOTEM Dataset
- Cluster
  - 15 workers
  - 16 cores/worker
- Requirements
  - Data access (EOS)
  - Software (CVMFS)



Conclusions

Accomplishments

- **Deployment** of a Spark infrastructure, using both on-premise and cloud-managed clusters
- Integration with **SWAN**, a web-based interactive analysis tool and “service federator”
  - Modern and ergonomic interface
  - Easy to access, use and share notebooks
  - Real-time monitoring of resources
- **Simplifying** the interface to physics analysis:
  - ROOT RDataFrame allows for declarative analysis, thus enabling optimisations behind the scenes
  - PyRDF wraps RDataFrame and enables distributed computation via Spark in a seamless way for the scientists
  - SWAN provides an interface for such an interactive and distributed approach

Challenges

- The increase in physics and controls data volumes and complexity is pushing software at CERN
  - Adoption of Spark and other big data technologies still in its early stages
- Large codebase developed over decades
  - Cannot change overnight
- Spread new paradigms to users
  - Declarative, interactive, web-based analysis vs local and compiled
  - Map-reduce dealing with columnar data
  - On-demand computing resources
- Prepare for HL-LHC data workflows
  - Test new technologies further with more data

Thank you!