

CERN's Platform for Data Analysis with Spark

Enric Tejedor, Prasanth Kothuri, CERN

##SAISExp1



Outline

1. What we do at CERN
2. Interactive data analysis with Spark
3. Use cases
 1. Physics data
 2. Controls data

Founded in 1954

Mission: fundamental research in Physics



Accelerating Science

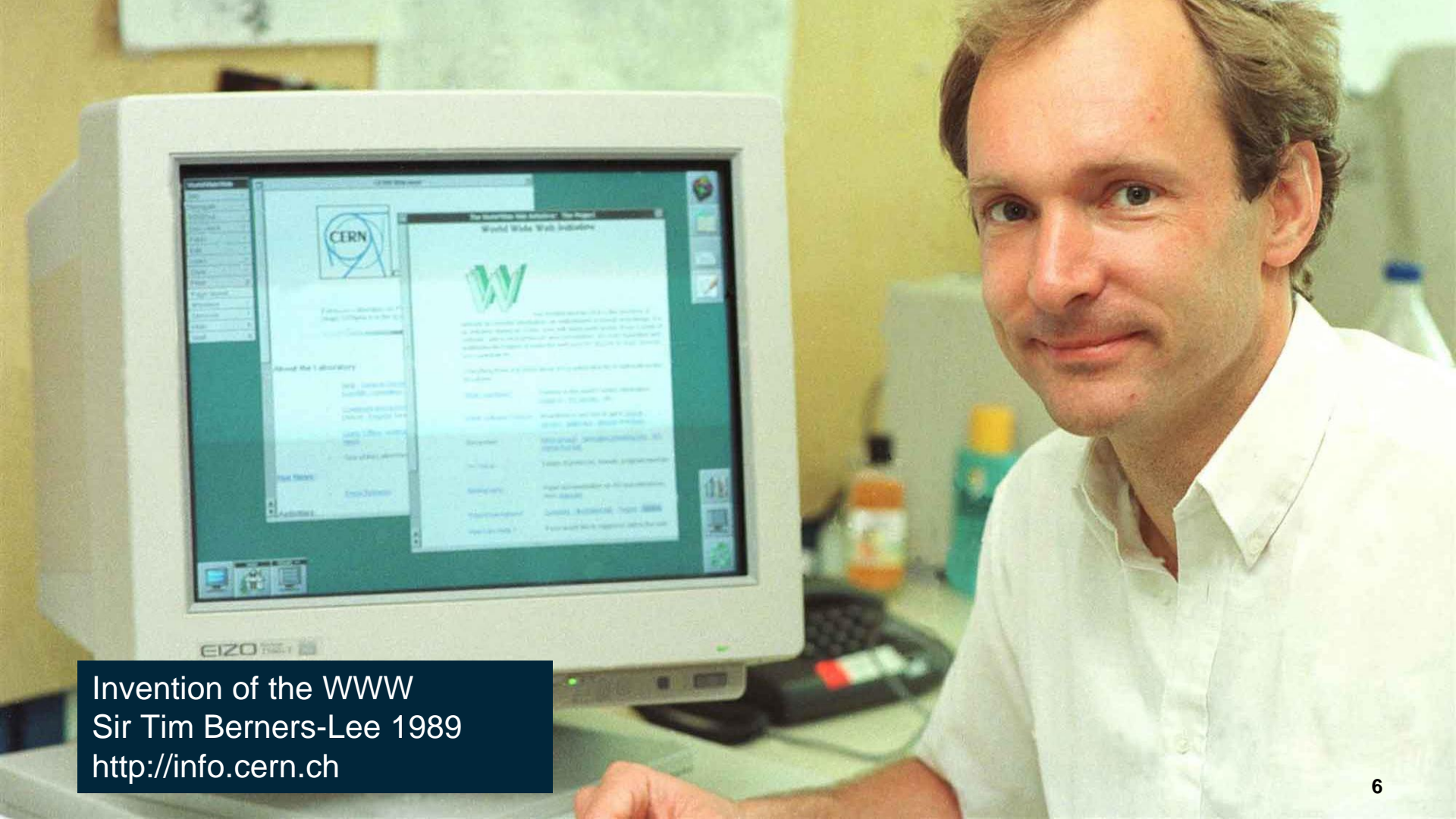


The Large Hadron Collider (LHC)
The largest scientific experiment to date
1 PB / second of raw physics data

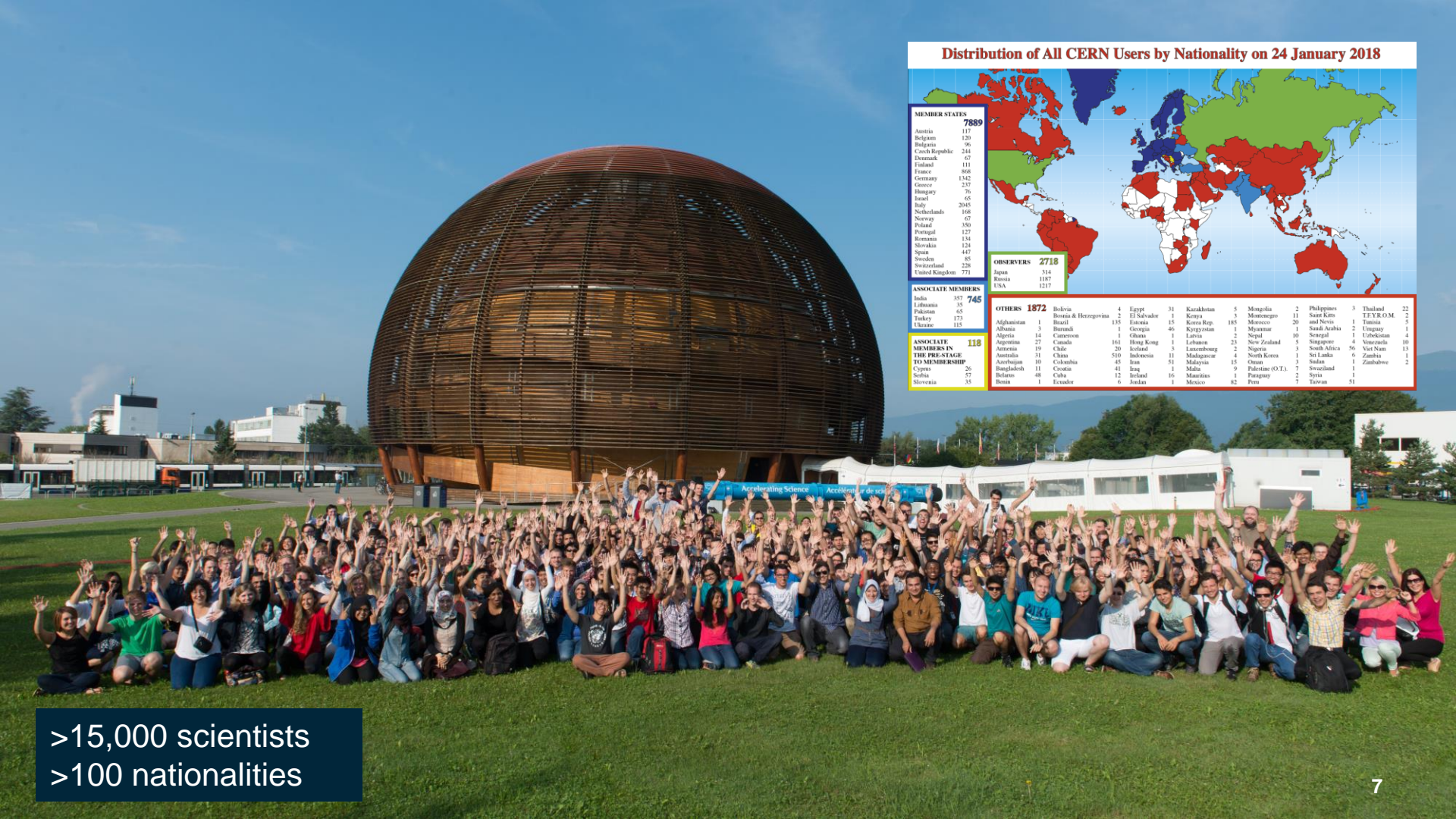




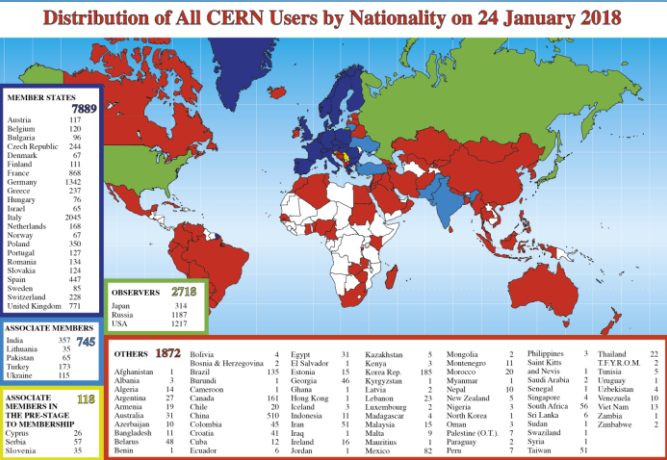
Discovery of the Higgs boson
ATLAS & CMS 2012



Invention of the WWW
Sir Tim Berners-Lee 1989
<http://info.cern.ch>



>15,000 scientists
>100 nationalities



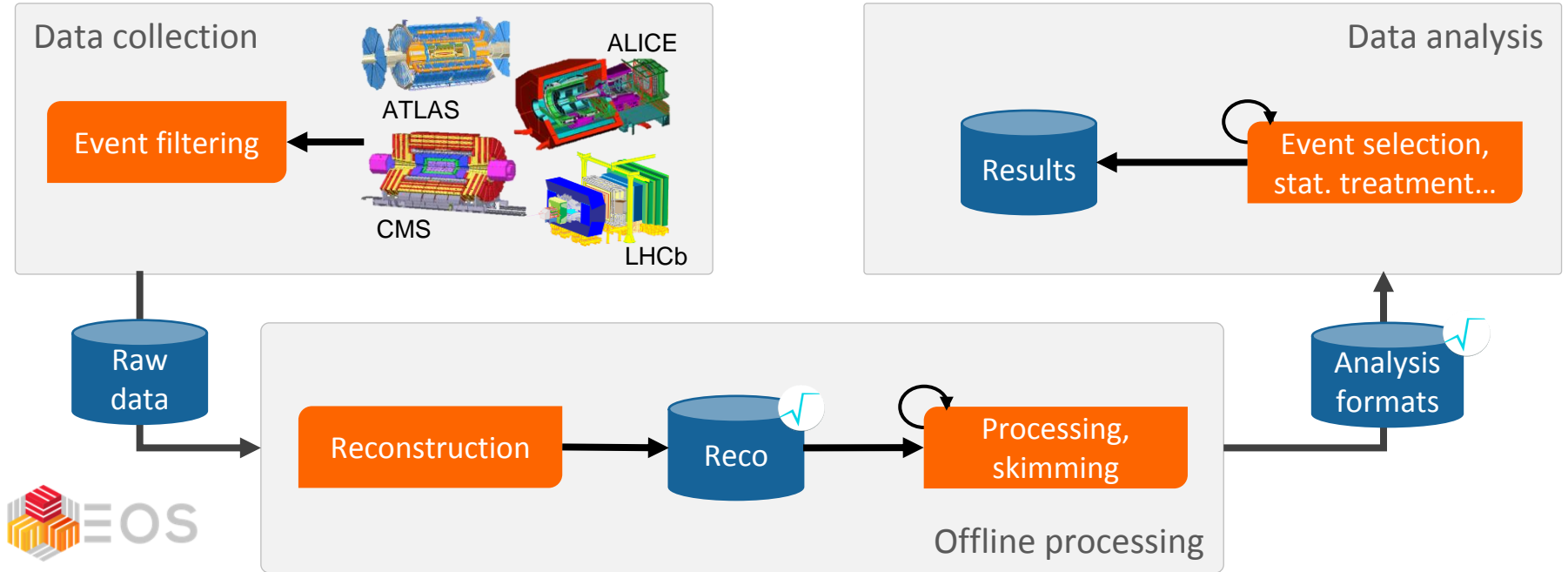


SPARK+AI
SUMMIT EUROPE

Physics Data Processing and Analysis at CERN

##SAISExp1

LHC Data Pipeline at CERN

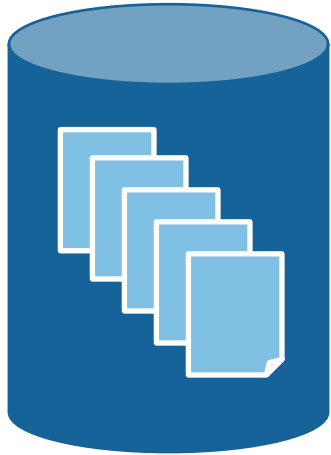


ROOT

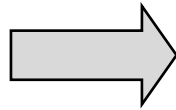
- **The** data analysis framework for high-energy physics (HEP)
- Data processing, statistical analysis, visualisation and storage
- **~1 EB** stored in ROOT format
 - Binary, compressed
 - Columnar



ROOT Dataset



ROOT dataset stored in
one or multiple files



px	py	pz	eta

Column:
physics entity



Row:
collision event

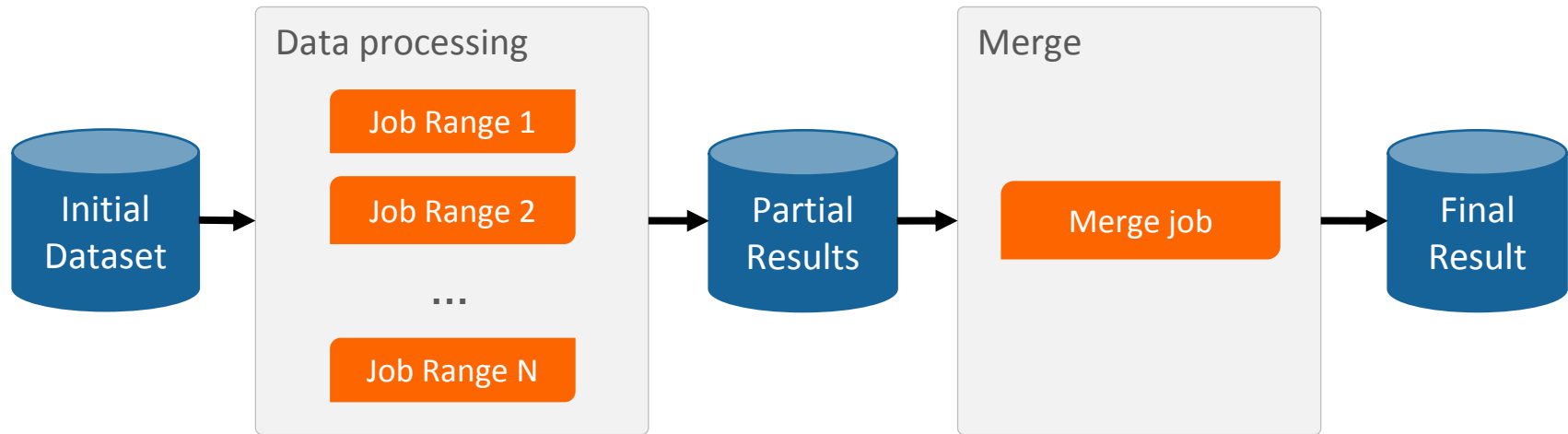
The LHC Computing Grid

Started in 2002
Provide processing power and data
access to physicists
~170 centres in 42 countries



Distributed Computing in HEP

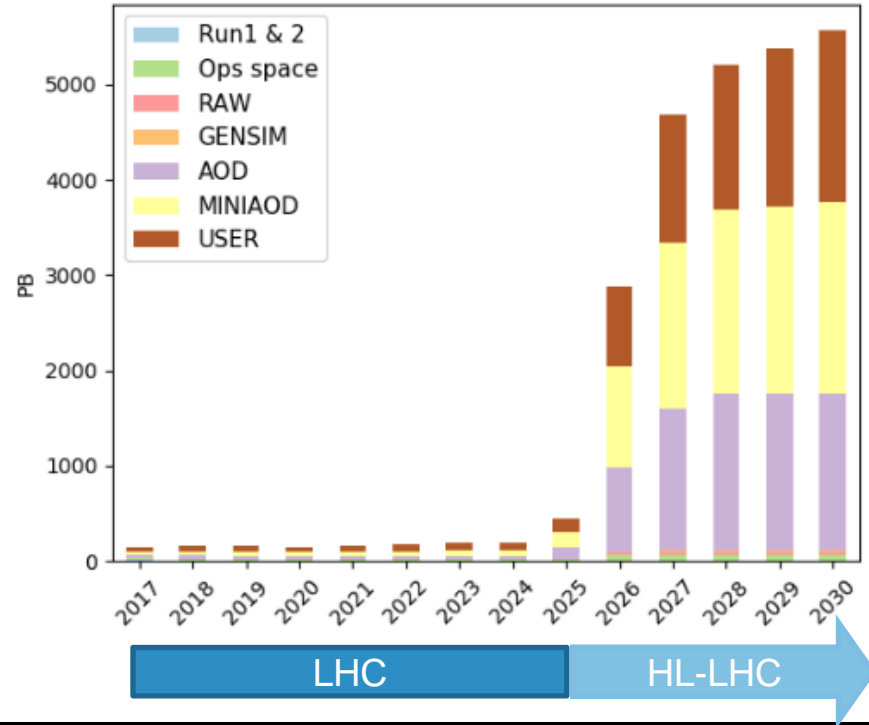
- Physicists use Grid and batch resources to process LHC data in parallel



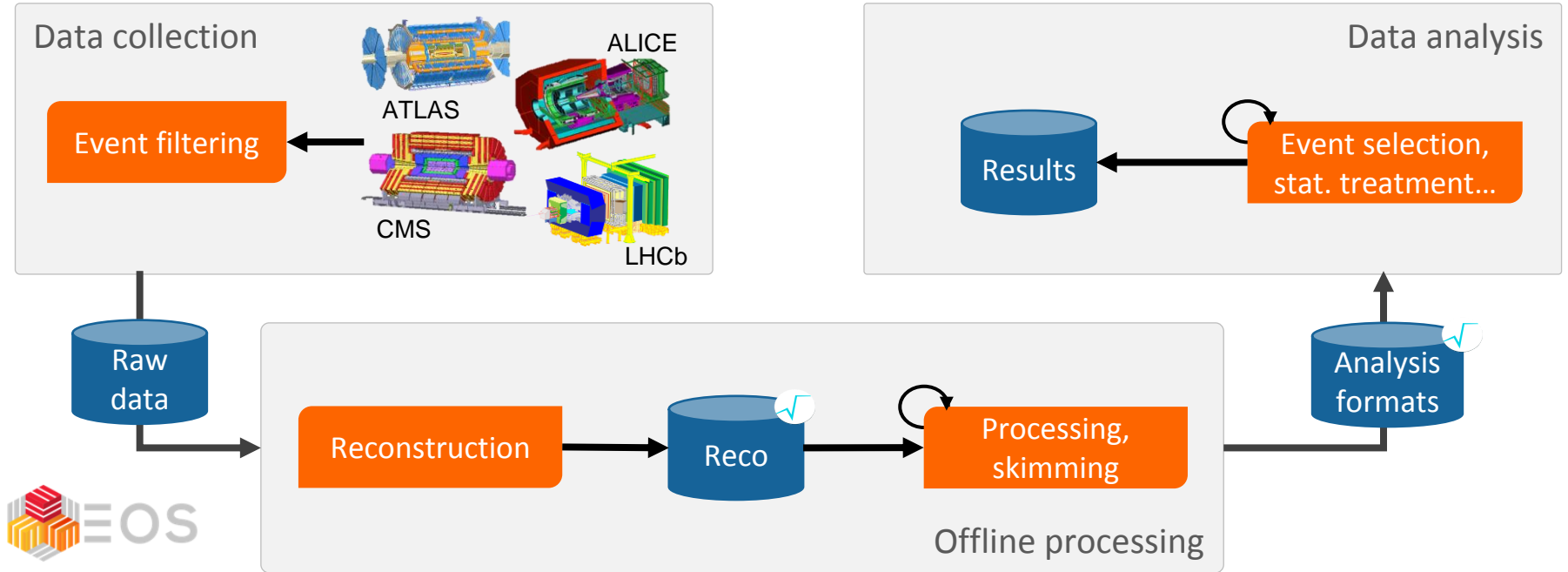
HL-LHC: Even More Data!

- Coming upgrade: High-Luminosity LHC
- **30x more data** collected
- Big challenge for software and computing

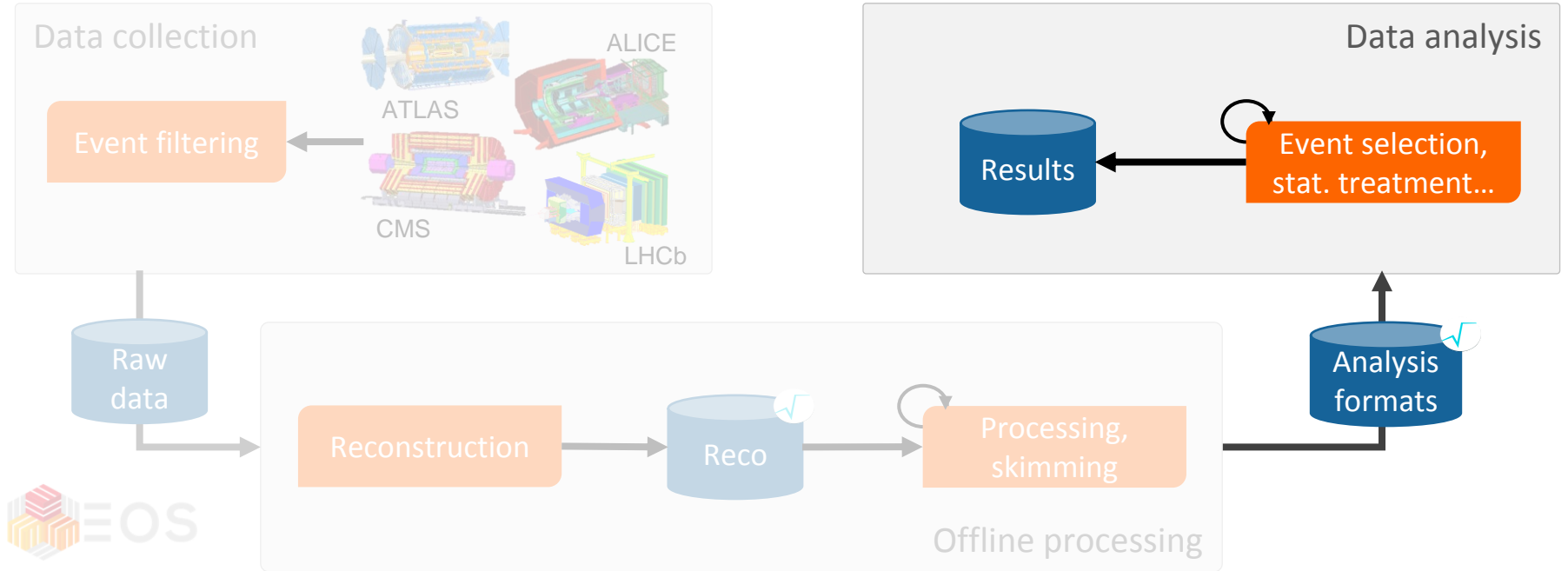
CMS estimated disk space required by tier



LHC Data Pipeline at CERN



Final Steps of Data Analysis





SPARK+AI
SUMMIT EUROPE

Interactive Data Analysis

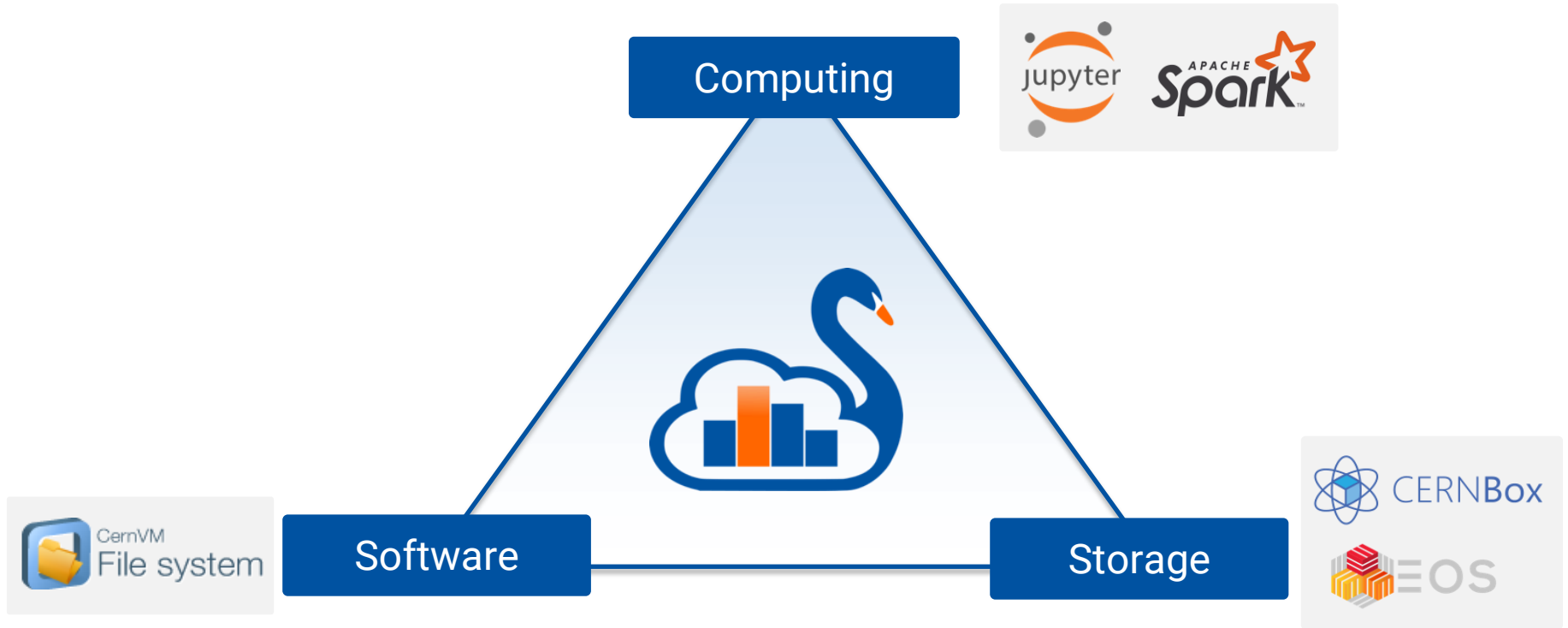
##SAISExp1

The SWAN Service

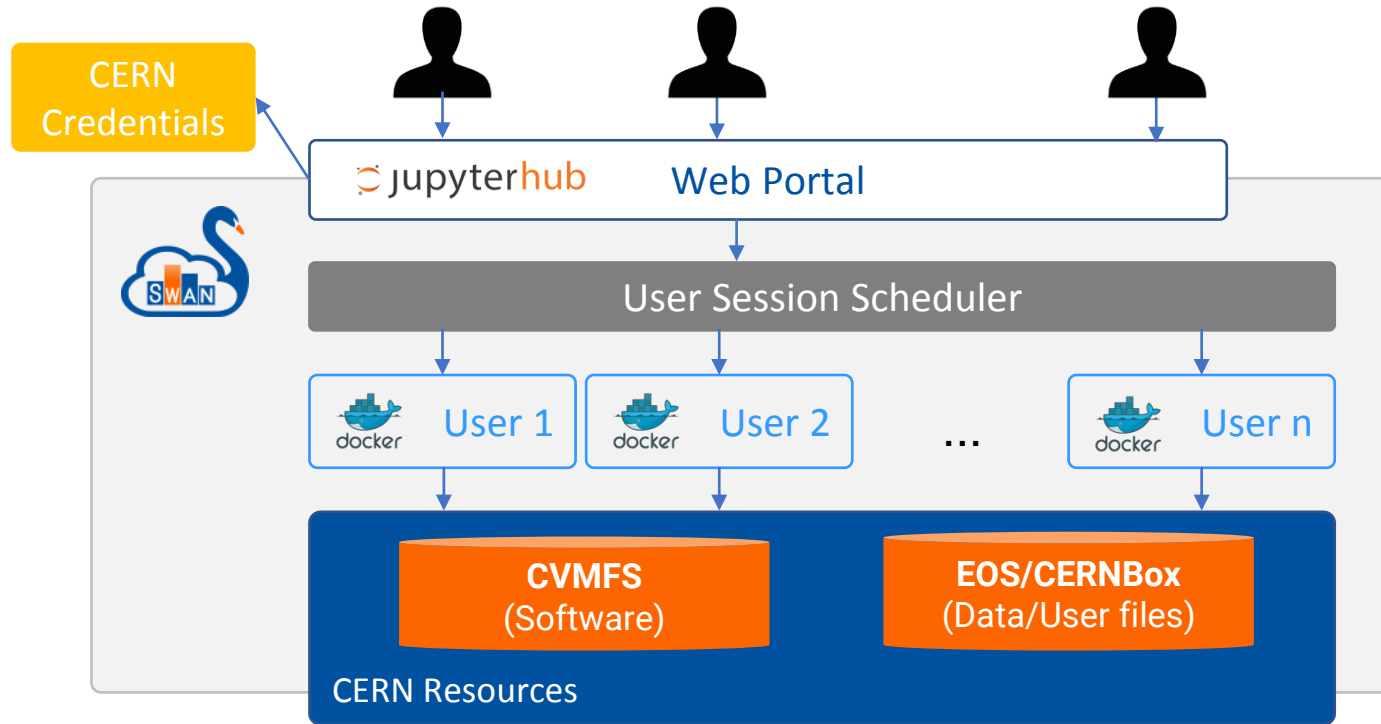
- **SWAN**: Service for Web-based Analysis
- **Interactive computing** platform for scientists
 - Based on Jupyter notebooks
- Analysis with only a web browser
- Easy **sharing** of results
- Integrated with CERN resources
 - Storage, software and computing



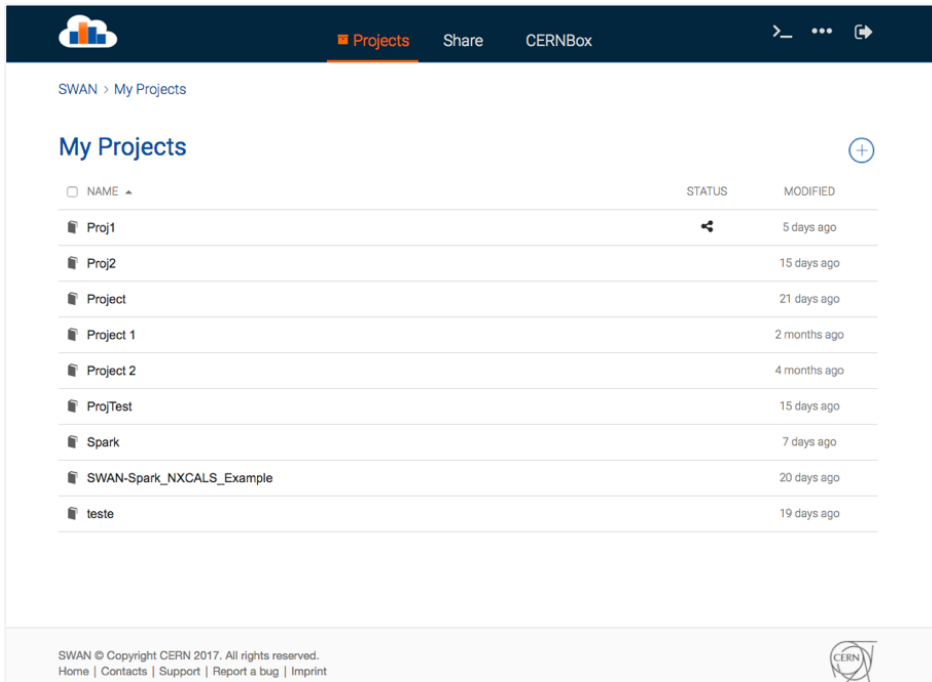
SWAN Pillars



SWAN Architecture



SWAN Interface: Notebooks

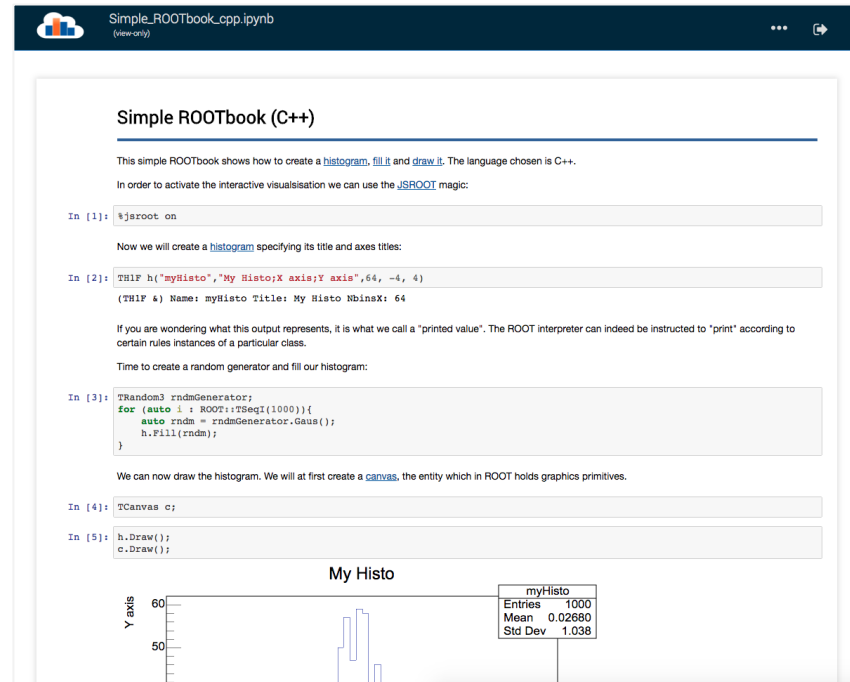


SWAN > My Projects

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



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]: %jarroot 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::ESeq(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();
```

My Histo



myHisto	
Entries	1000
Mean	0.02680
Std Dev	1.038

Sharing in SWAN

SWAN > My Projects > Super Real Analysis with TOTEM data

Super Real Analysis with TOTEM data

NAME

- DistillDistribution.ipynb
- dataset.root

Share Project

You are sharing:
Super Real Analysis with TOTEM data

Search by name or username.
Use "a:" for secondary accounts.

Shared with

- Danilo Piparo (danilo)
- Enric Tejedor Saavedra (enric)

Stop Sharing Update

SWAN > Share

Projects shared with me

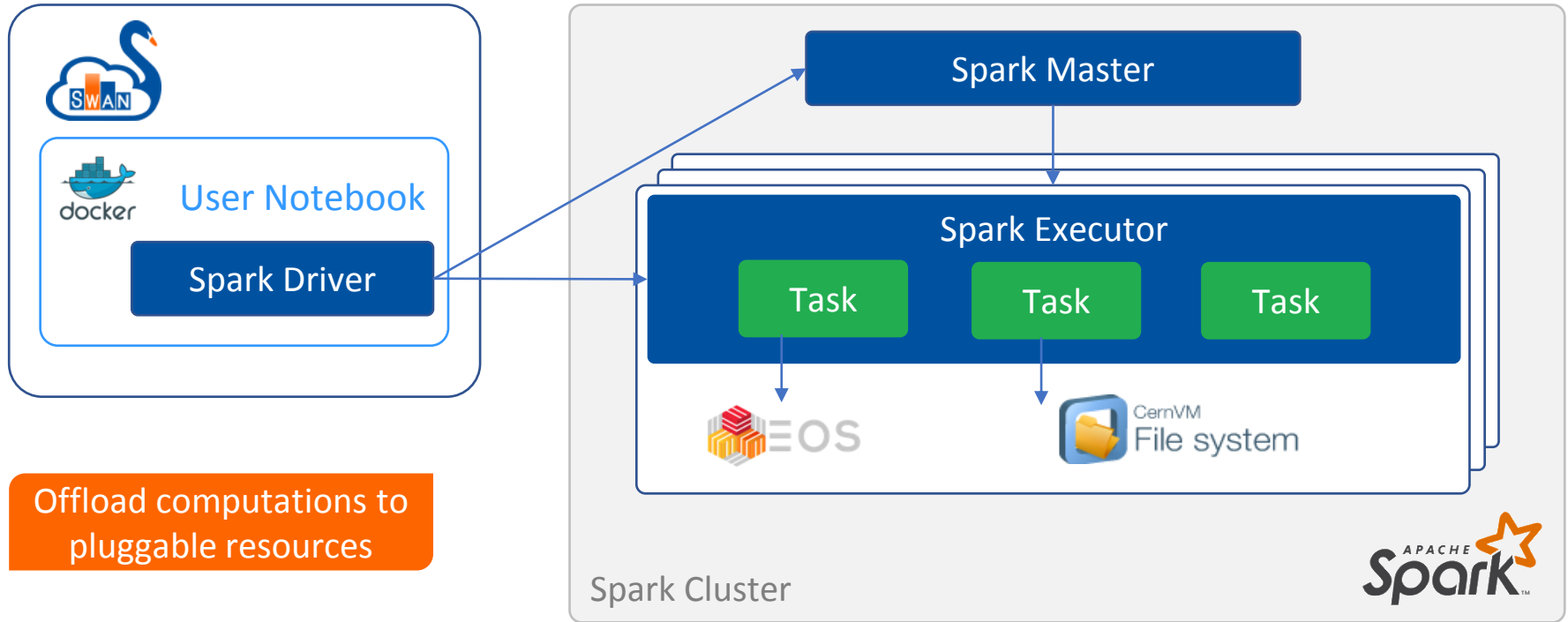
NAME	SIZE	SHARED BY	DATE
UP2University Pilot	Empty	jupytercon2	7 minutes ago

Projects shared by me

NAME	SHARED WITH	DATE
Higgs Boson discovery	2 people/groups	18 hours ago
Super Real Analysis with TOTEM data	diogo	19 hours ago

SWAN © Copyright CERN 2016-2018. All rights reserved.
Home | Contact | Support | Report a bug

Integration with Spark



Spark Connector

Spark > physics_analysis_using_swan_spark_template

FILE EDIT VIEW INSERT CELL KERNEL HELP Not Trusted Python 2

Integration of SWAN with Spark clusters

This notebook demonstrates the functionality provided by a SWAN prototype machine that allows to offload computations to an external Spark cluster. The Spark version we are going to use is 2.1.0 and we are going to connect to the analytics cluster (as previously selected in the SWAN web form).

Step 1 - Acquire the necessary credentials to access the Spark cluster.

```
In [1]: import getpass
import os, sys, re

print("Please enter your password")
ret = os.system("echo \"%s\" | kinit" % re.escape(getpass.getpass()))

if ret == 0: print("Credentials created successfully")
else: sys.stderr.write("Error creating credentials, return code: %s\n" % ret)
```

Spark > Spark_Simple

Spark clusters connection

You are going to connect to:
hadalytic

You can configure the following [options](#).
Environment variables can be used via (ENV_VAR_NAME).

Add a new option

Bundled configurations

Include NXCAL options

Selected configuration

- spark.shuffle.service.enabled: false
- spark.driver.memory: 2g
- spark.executor.instances: 4

Connect

Configure Spark and connect to cluster with a click

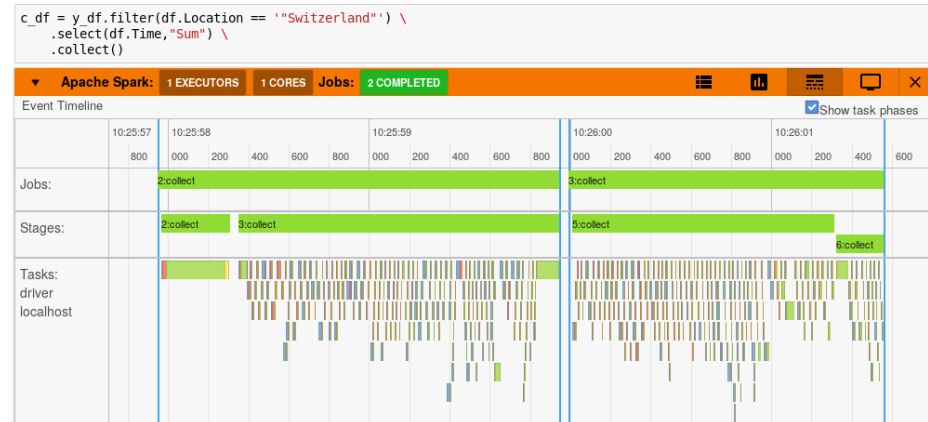
Spark Monitor



[Code here!](#)

Google Summer of Code

- 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



Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
2	reduce	COMPLETED	2/2	48 / 48	5 minutes ago	3s
Stage Id	Stage Name	Status	Tasks	Submission Time	Duration	
5	reduce	COMPLETED	32 / 32	5 minutes ago	2s	
4	coalesce	COMPLETED	16 / 16	5 minutes ago	0s	
3	foreach	COMPLETED	1/1 (1 skipped)	32 / 32	5 minutes ago	1m:20s
Stage Id	Stage Name	Status	Tasks	Submission Time	Duration	
6	coalesce	SKIPPED	0 / 16	Unknown	-	
7	foreach	COMPLETED	32 / 32	5 minutes ago	1m:20s	



SPARK+AI
SUMMIT EUROPE

Physics Data Use Case

##SAISExp1

The HEP DataFrame

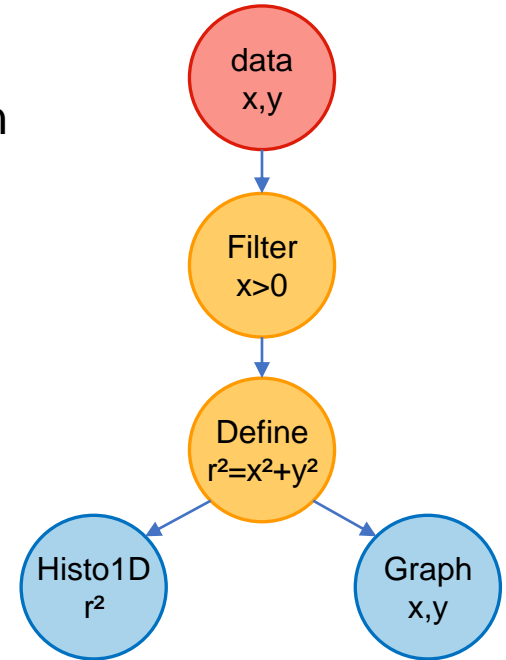
- **RDataFrame**

- Implemented in C++, interfaced also to Python
- Tailored for ROOT and HEP

```
df = RDataFrame(dataset)

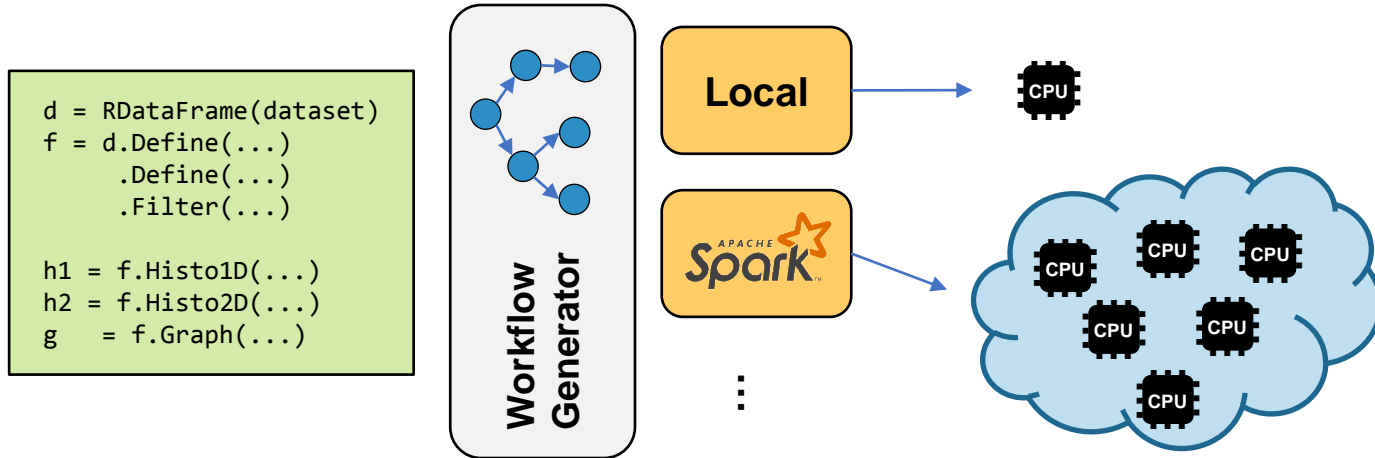
df2 = df.Filter('x > 0')
        .Define('r2', 'x*x + y*y')

h = df2.Histo1D('r2')
g = df2.Graph('x', 'y')
```



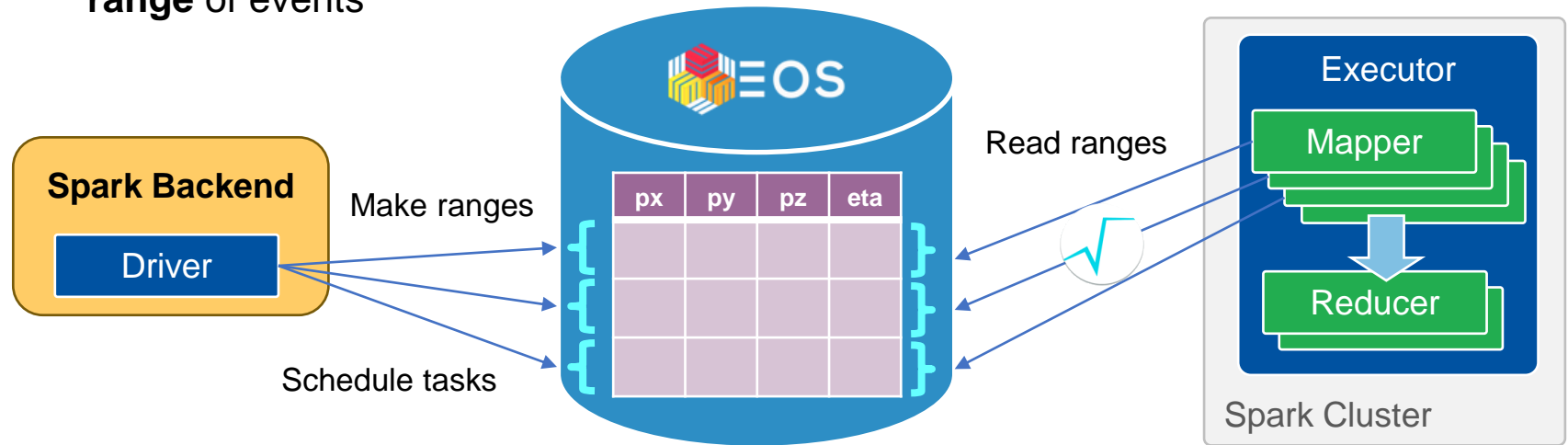
Distributed RDataFrame

- Exploratory work to parallelise RDataFrame computations with multiple backends



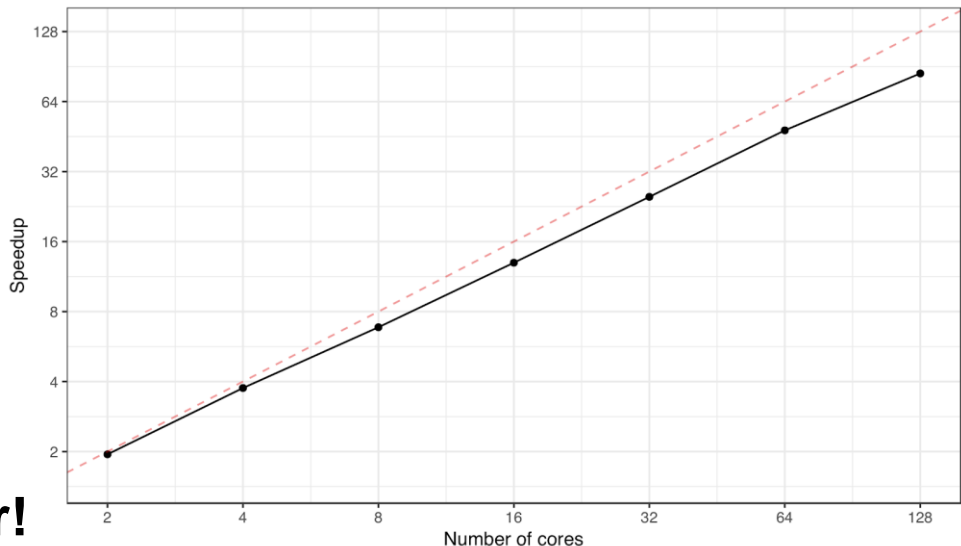
Spark Backend for RDataFrame

- Input: ROOT physics dataset consisting of n **collision events** (rows) with k **properties** (columns)
- **Map-reduce** workflow where every mapper runs the RDataFrame computation on a **range** of events



Real Example: TOTEM Analysis

- TOTEM experiment analysis coded with RDataFrame
- **Spark** backend
- **4.7 TB** dataset on EOS
- Launched from **SWAN** to a dedicated Spark cluster
 - 16 cores per executor
- **Get to physics results faster!**





SPARK+AI
SUMMIT EUROPE

Controls Data Use Case

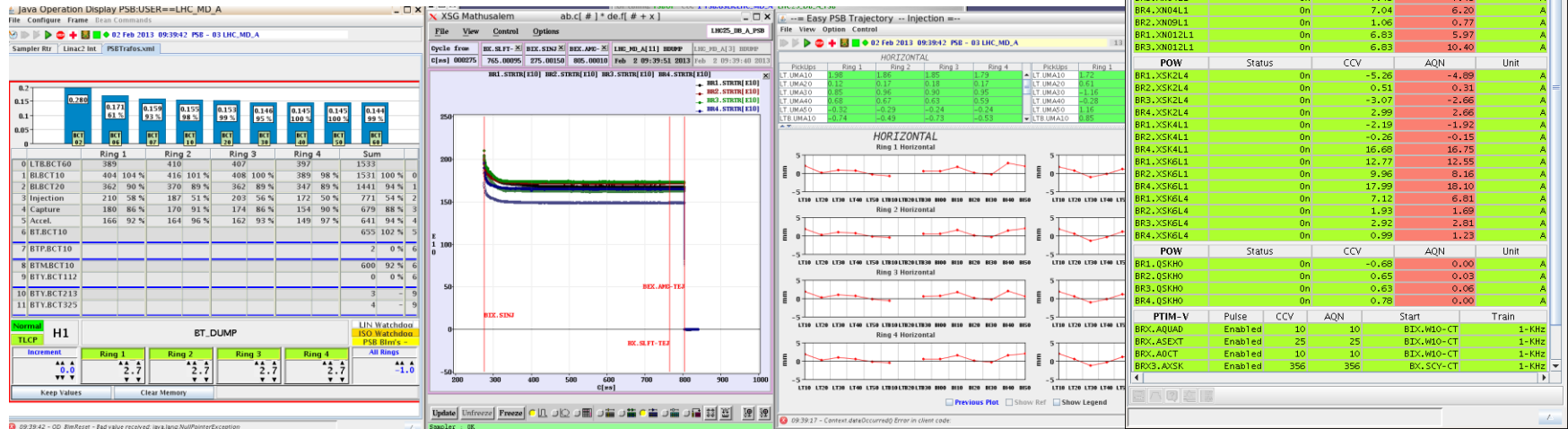
##SAISExp1



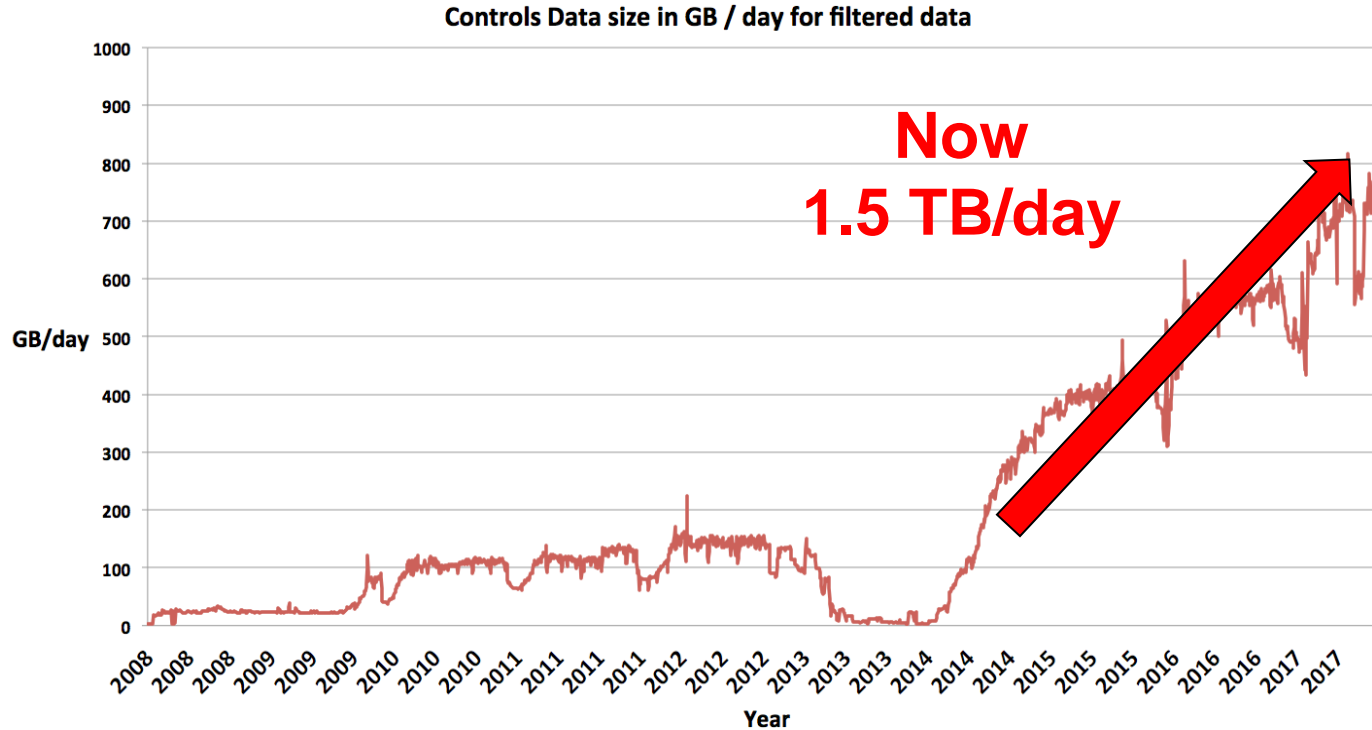
LHC: Huge machine, highly sophisticated
Control and monitoring are crucial

Hardware Controls

- Complex control system for monitoring
- 1000s of devices, 100s of properties each



Controls Data Growth



CERN Accelerator Logging Service

- Old system based on SQL databases
 - Hard to scale horizontally
 - Slow data extraction
- New system (NXCALS)
 - Data pumped into HBase and HDFS (Parquet)
 - **Spark** to extract and process data
 - **SWAN** to visualise + analyse



NXCALS Data Analysis in SWAN

- NXCALS bet on SWAN as their data analysis platform
- Connection to Spark clusters
- Access to software (data science Python ecosystem)

Inspect data

```
In [2]: df1.select('acqStamp', 'voltage_18V', 'current_18V', 'device', 'pt100Value').toPandas()[0:5]
```

```
Out[2]:
```

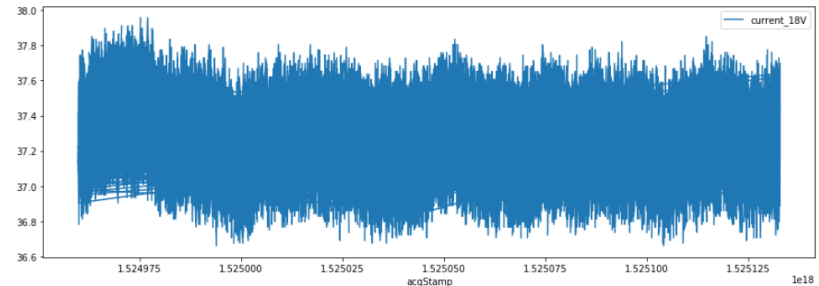
	acqStamp	voltage_18V	current_18V	device	pt100Value
0	1524960103132865000	NaN	37.301794	RADMON.PS-10	106.578911
1	1524960284134584000	NaN	NaN	RADMON.PS-10	107.246742
2	1524960322134942000	NaN	37.560940	RADMON.PS-10	106.504707
3	1524960353135244000	20.099066	NaN	RADMON.PS-10	107.068654
4	1524960911140548000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





SPARK+AI
SUMMIT EUROPE

Summary

##SAISExp1

Challenges

- The increase in physics and controls data volumes 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
- Changing the mindset of programmers takes time
 - Declarative analysis
 - Pushing computations to data

Future Directions

- Bridge the gap between data processing needs and technology evolution
 - Complement traditional ways with new strategies
- Combine interactive analysis with access to more processing power
 - Provide higher-level programming models
 - Facilitate the access to computing resources
- More on CERN and Spark: stay tuned for [Luca's](#) and [Prasanth's](#) presentations



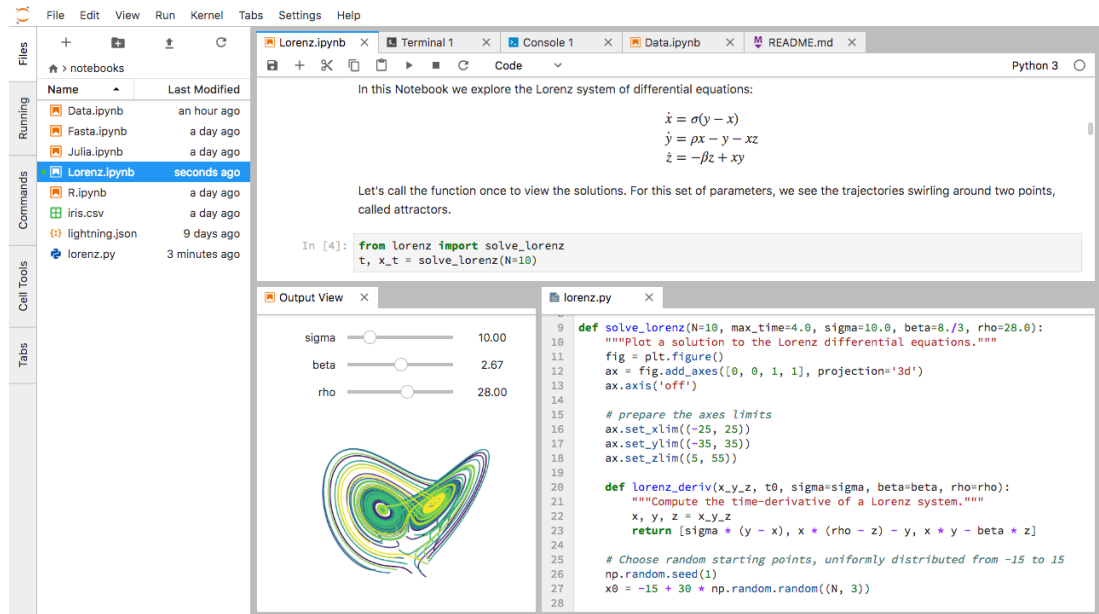
SPARK+AI
SUMMIT EUROPE

Backup Slides

##SAISExp1

JupyterLab

- Jupyter is evolving towards a desktop-like environment
 - Notebook, terminal, file browser, editors, ...
 - Highly customisable



The screenshot displays the JupyterLab desktop environment. On the left, a sidebar contains a file browser showing a list of files and folders, including 'Data.ipynb', 'Fasta.ipynb', 'Julia.ipynb', 'Lorenz.ipynb' (selected), 'R.ipynb', 'iris.csv', 'lightning.json', and 'lorenz.py'. Below the file browser are sections for 'Running', 'Commands', 'Cell Tools', and 'Tabs'. The main workspace is divided into several panes: a code editor for 'Lorenz.ipynb' containing text and mathematical equations, a 'Terminal 1' pane, a 'Console 1' pane, and an 'Output View' pane. The code editor shows the following content:

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

The 'Output View' pane displays a 3D plot of the Lorenz attractor, a complex, swirling trajectory in a 3D space. To the right of the plot are three sliders for parameters: sigma (set to 10.00), beta (set to 2.67), and rho (set to 28.00). The 'lorenz.py' file is open in a separate pane, showing the following Python code:

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x,y,z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```