



CERN's Platform for Data Analysis with Spark

Enric Tejedor, Prasanth Kothuri, CERN

#SAISExp1



Outline

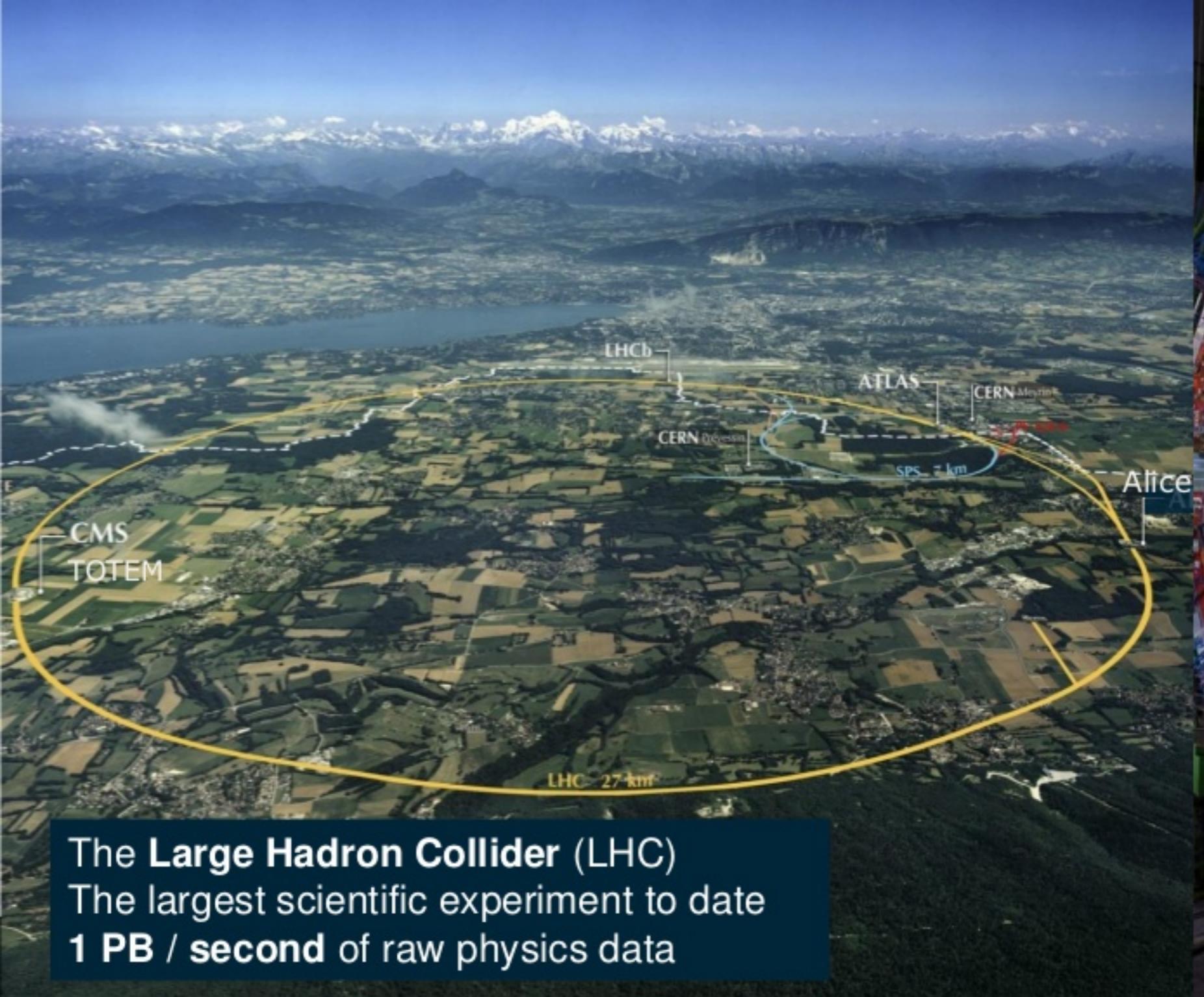
1. What we do at CERN
2. Interactive data analysis with Spark
3. Use cases
 1. Physics data
 2. Controls data

Founded in 1954

Mission: fundamental research in Physics



Accelerating Science

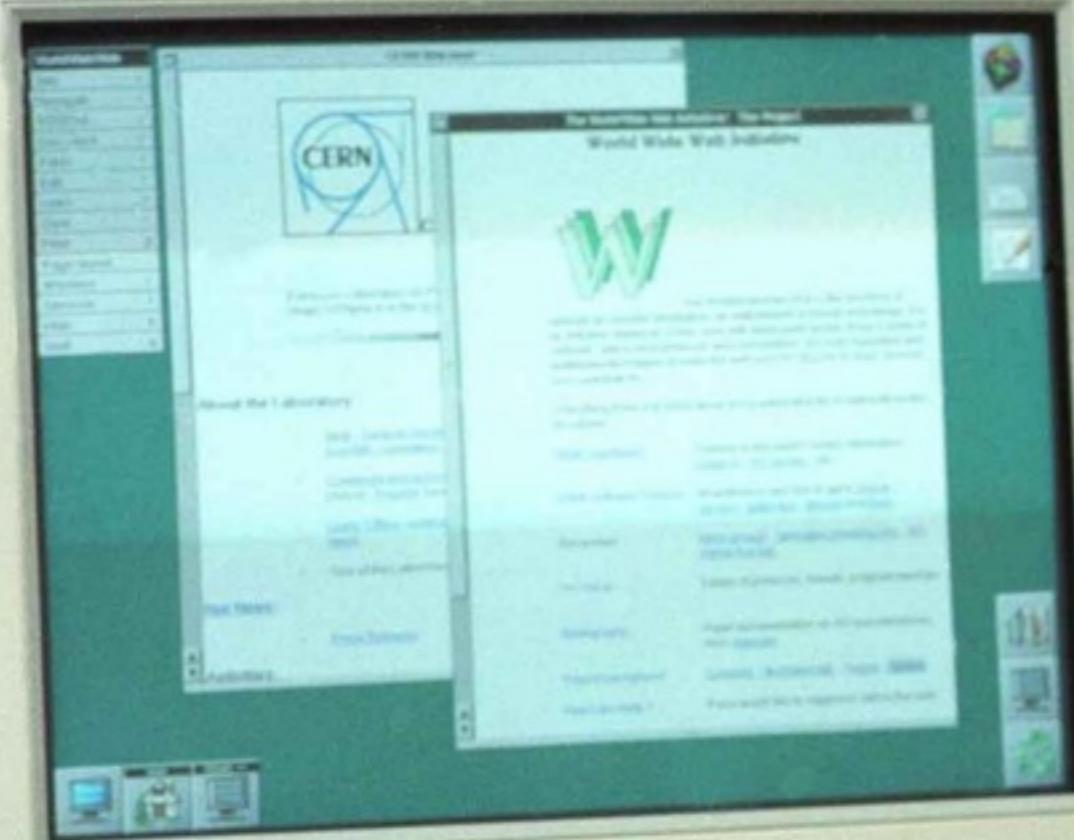


The Large Hadron Collider (LHC)
The largest scientific experiment to date
1 PB / second of raw physics data

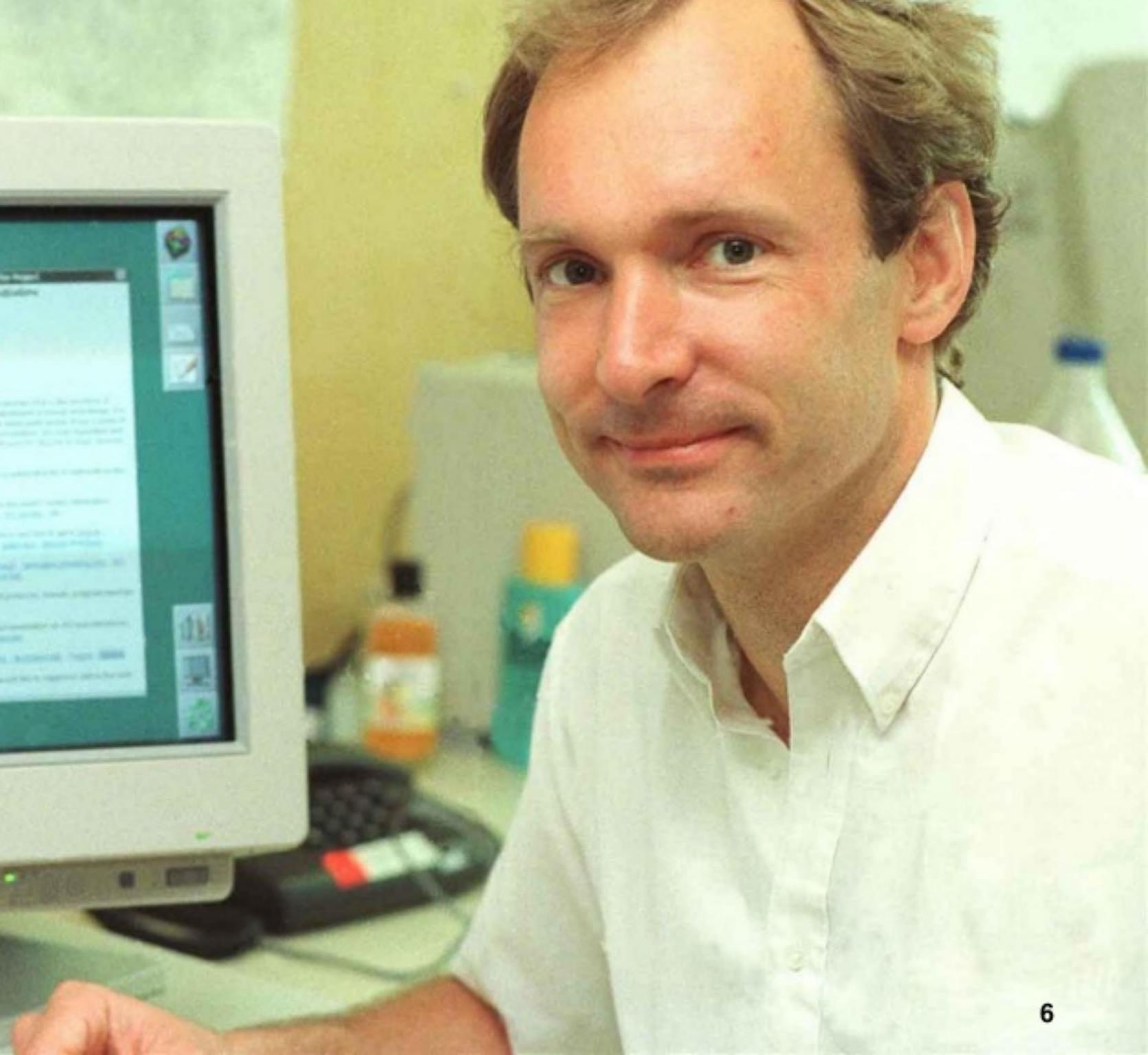


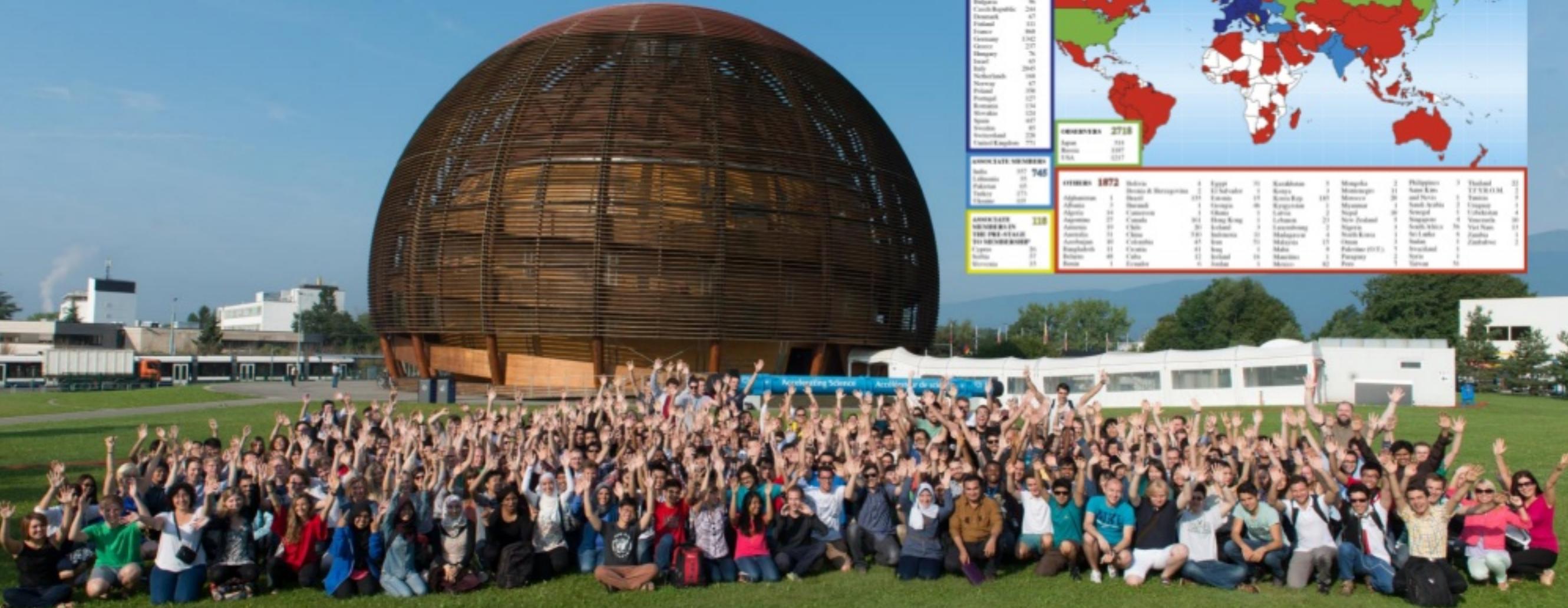


Discovery of the Higgs boson
ATLAS & CMS 2012



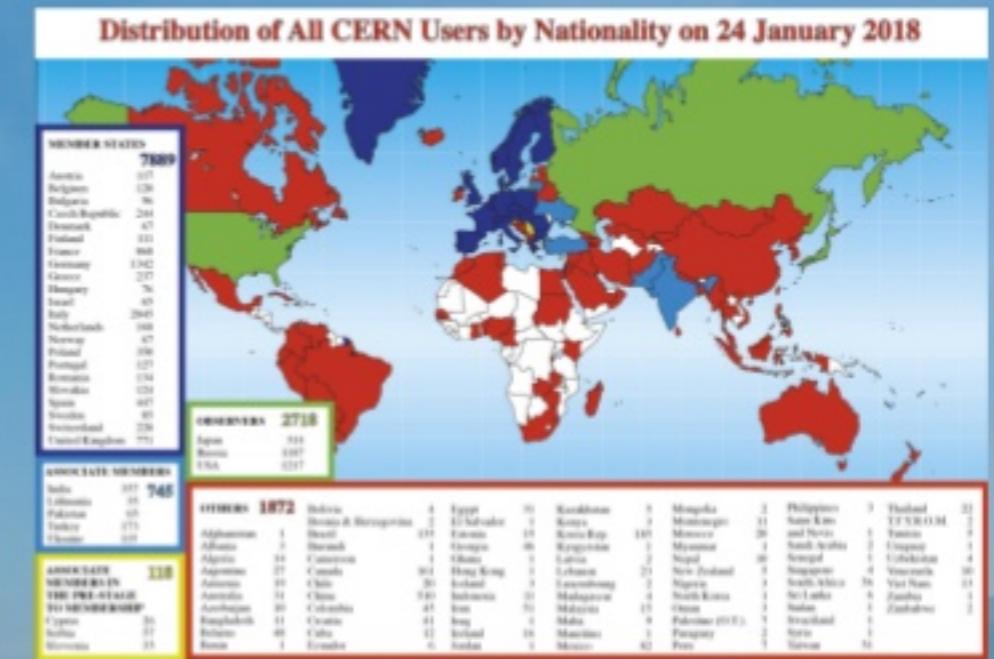
Invention of the WWW
Sir Tim Berners-Lee 1989
<http://info.cern.ch>





>15,000 scientists
>100 nationalities

Distribution of All CERN Users by Nationality on 24 January 2018

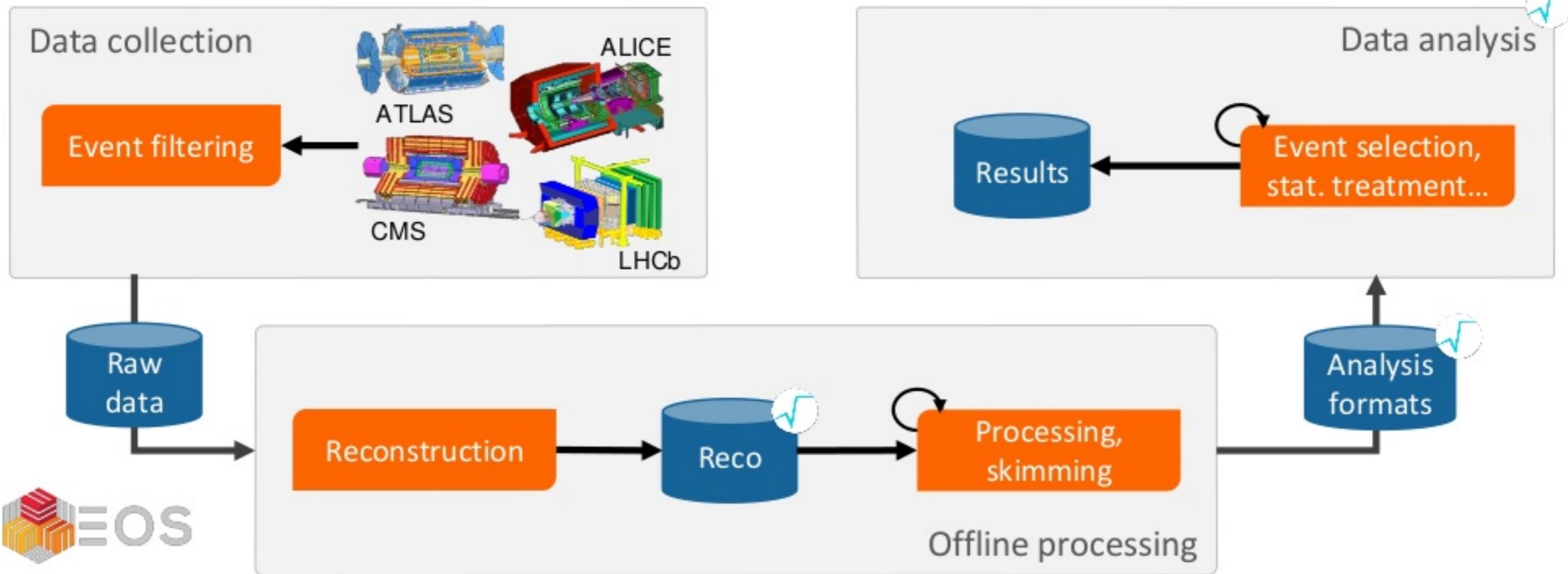




Physics Data Processing and Analysis at CERN

#SAISExp1

LHC Data Pipeline at CERN



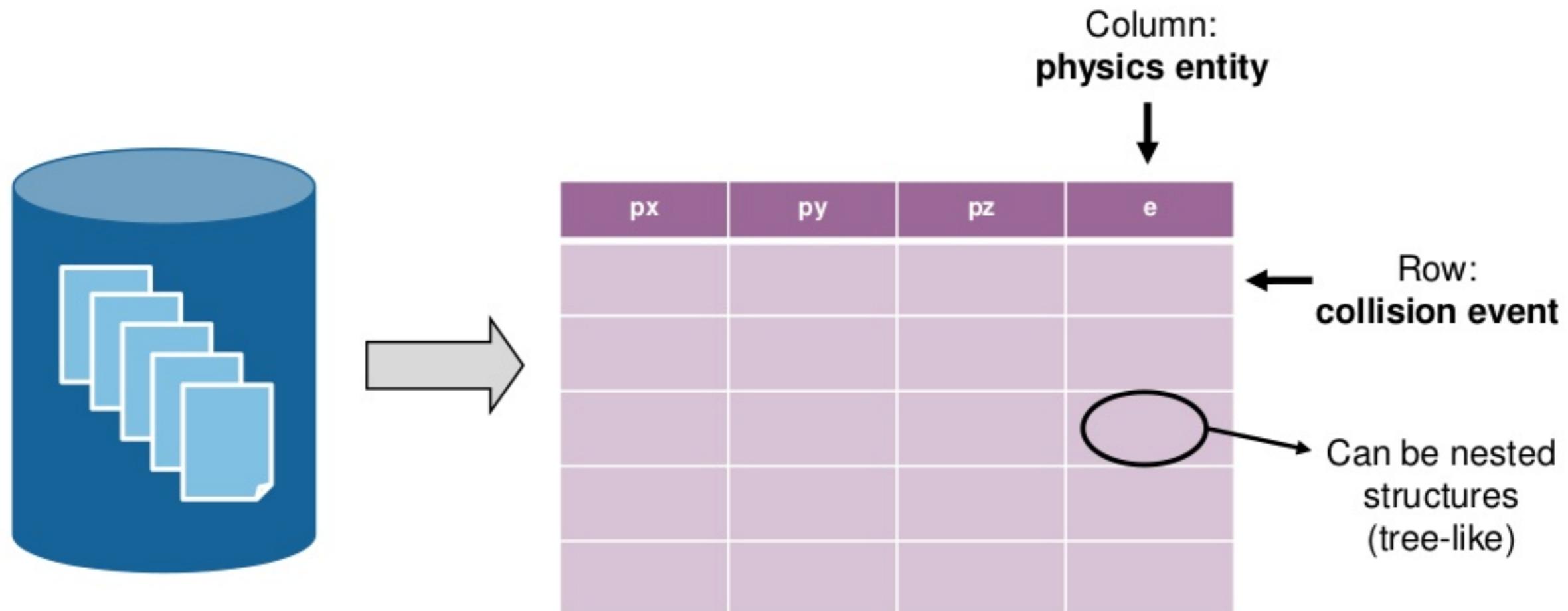
ROOT

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



<https://root.cern>

ROOT Dataset



ROOT dataset stored in
one or multiple files

The LHC Computing Grid

Started in 2002

Provide processing power and data
access to physicists

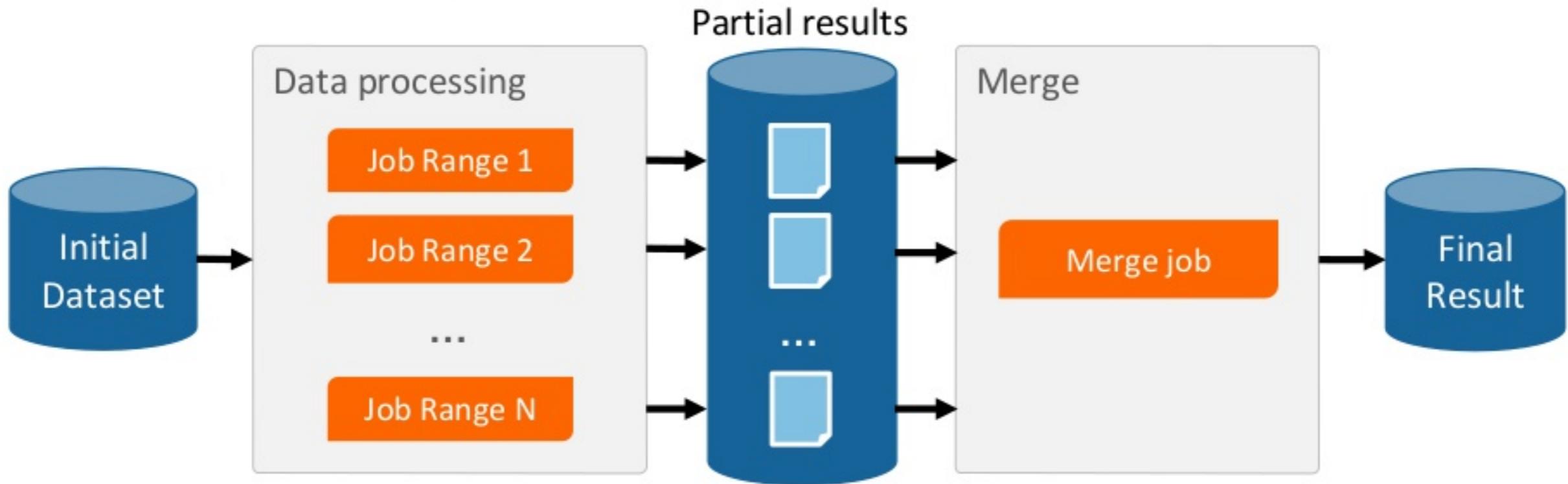
~170 centres in 42 countries

Running 24/7/365



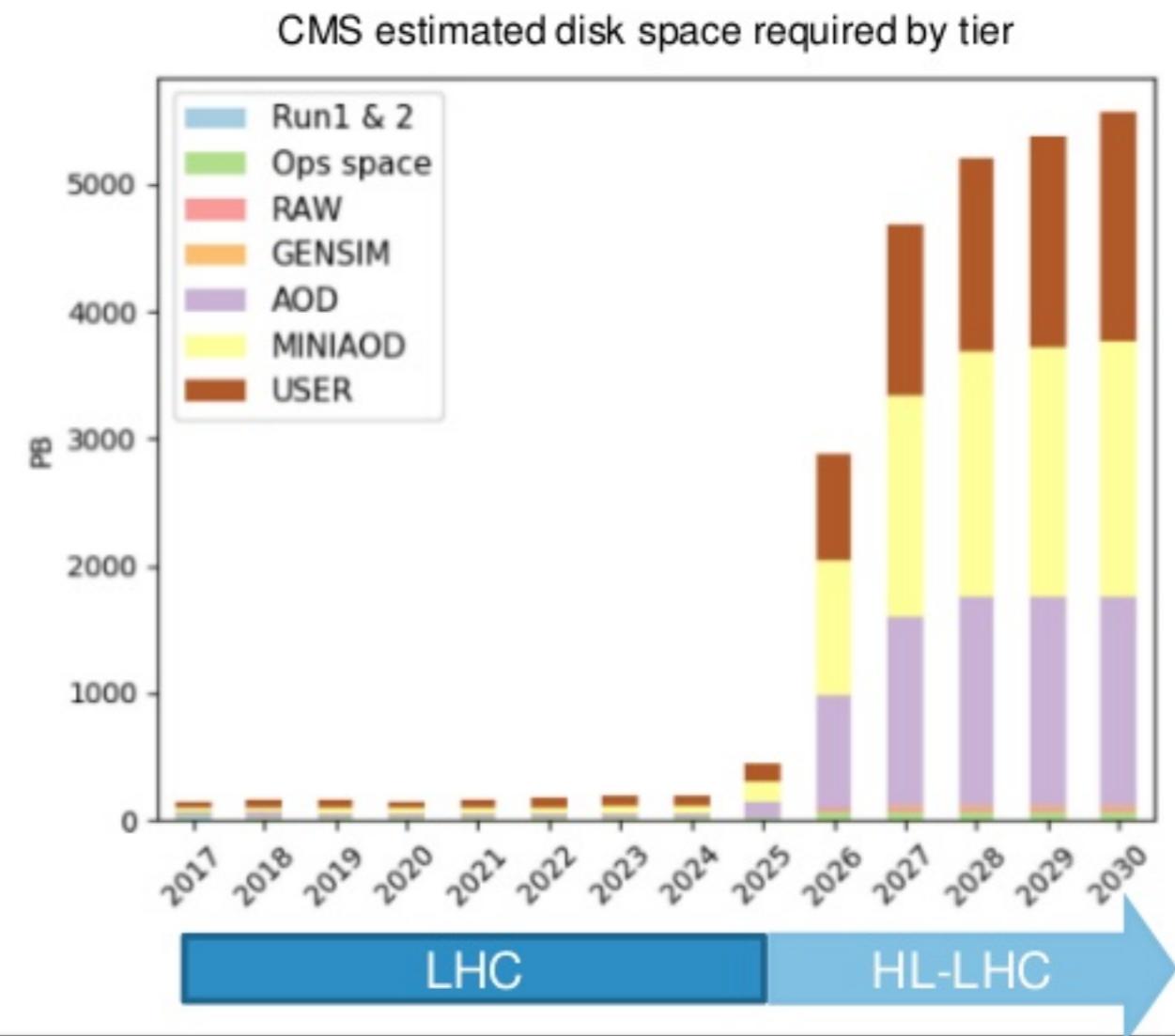
Distributed Computing in HEP

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



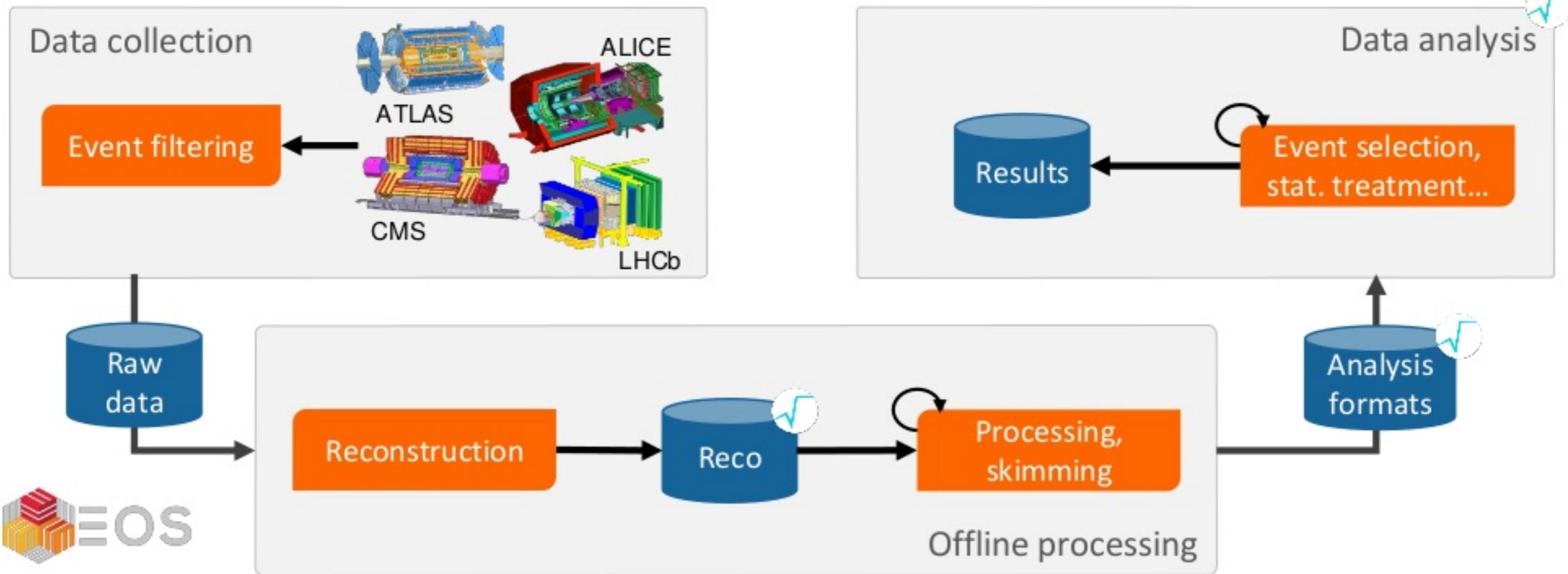
HL-LHC: Even More Data!

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

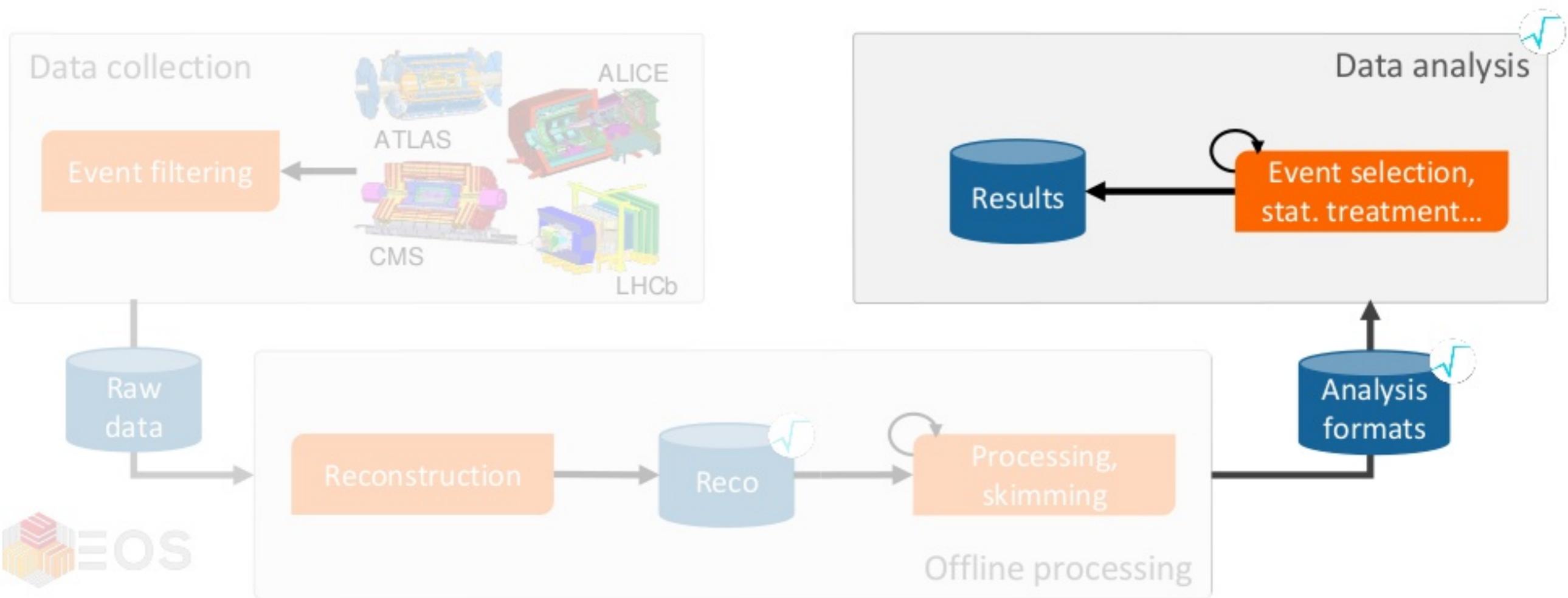


Source: HSF community white paper

LHC Data Pipeline at CERN



Final Steps of Data Analysis





Interactive Data Analysis

##SAISExp1

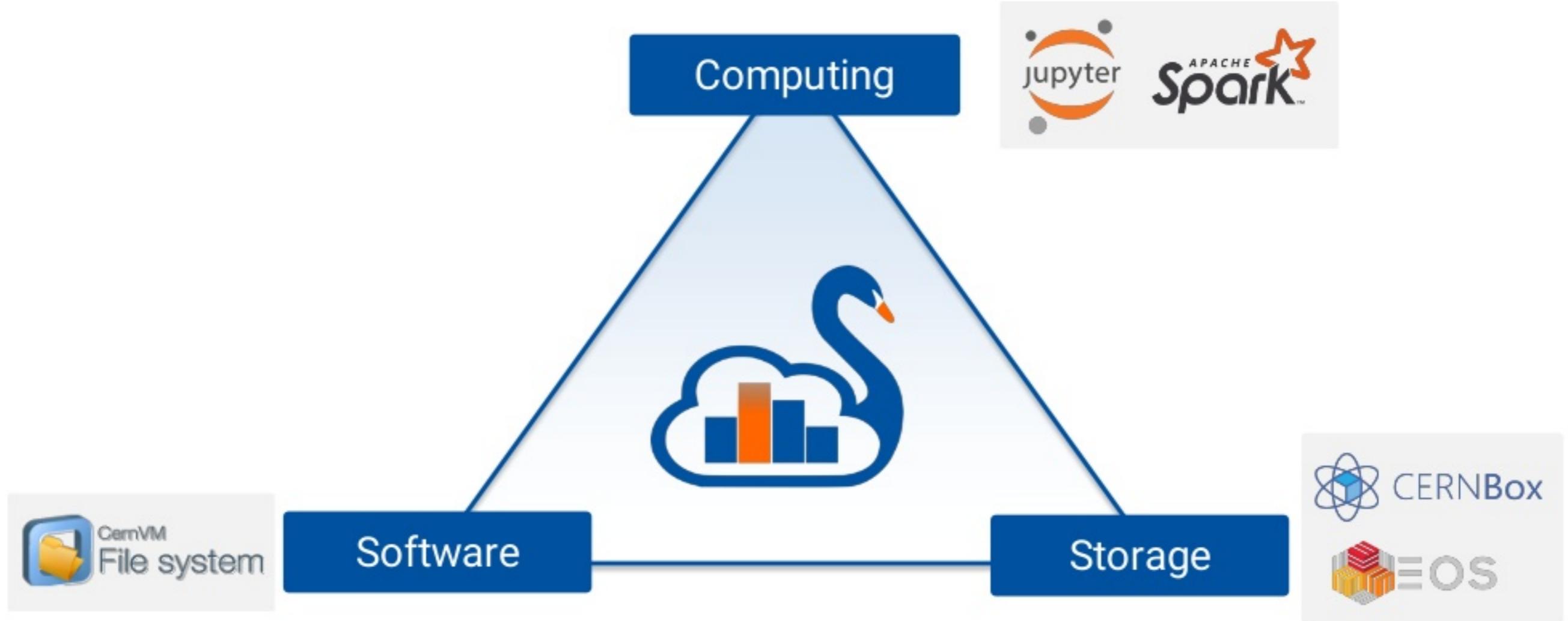
The SWAN Service

- **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

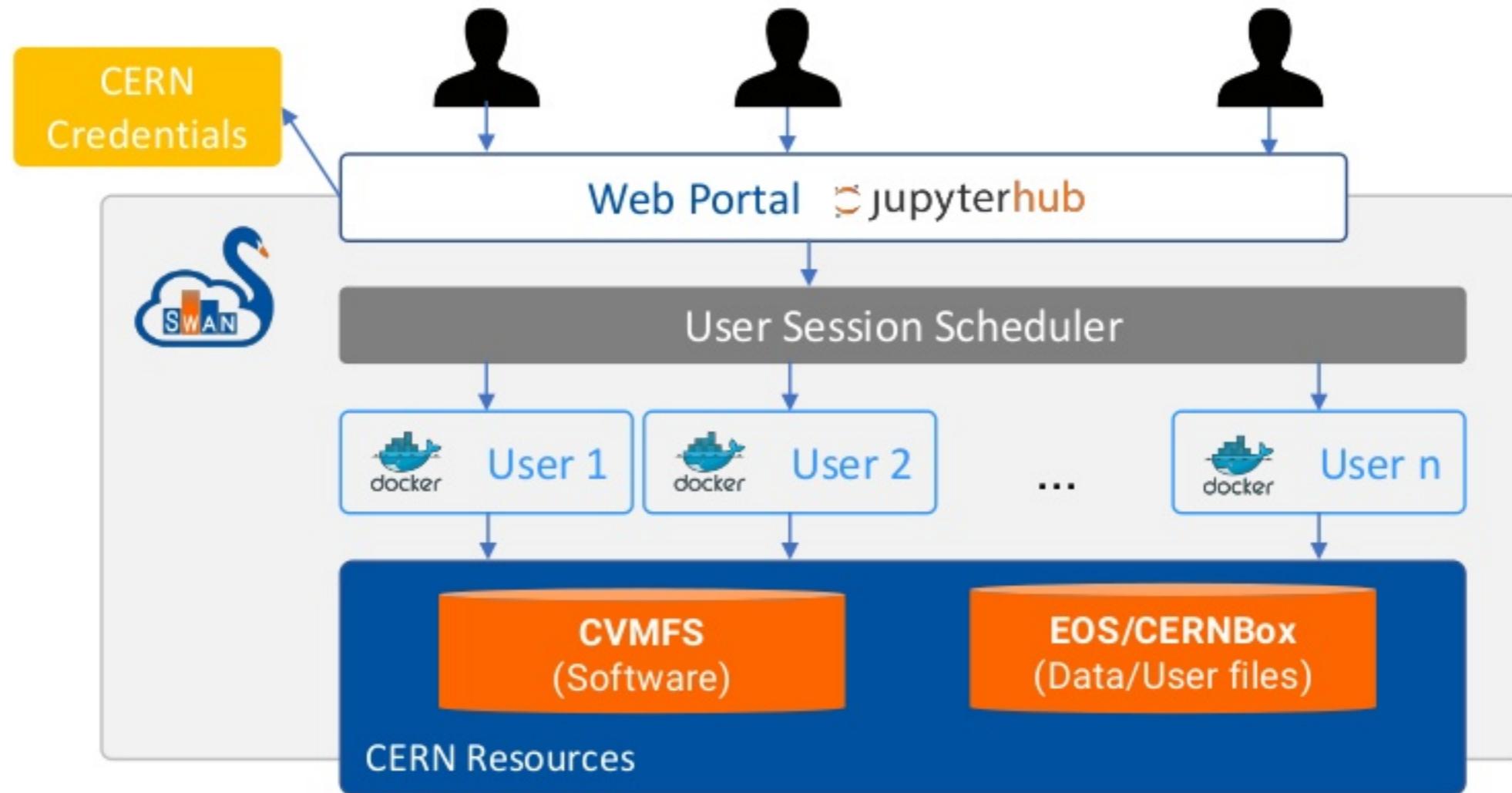


<https://swan.web.cern.ch>

SWAN Pillars



SWAN Architecture



SWAN Interface: Notebooks

Projects		Share	CERNBox	
SWAN > My Projects				
<h2>My Projects</h2>				
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	

Simple_ROOTbook_0pp.ipynb
In [1]:

Simple ROOTbook (C++)

This simple ROOTbook shows how to create a `TH1F` and `TH1D`. The language chosen is C++.

In order to activate the interactive visualisation we can use the `%ROOT` magic:

```
In [1]: %jupyter os
```

Now we will create a `TH1F` specifying its title and axes titles:

```
In [2]: TH1F h("myHisto","My Histotitle axisY axisX",64, -4, 4)
```

```
In [2]: h; h.GetName() == "myHisto" & h.GetTitle() == "My Histotitle axisY axisX": 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 rndGenerator;
for (auto i = 0; i < 1000; i+=1) {
    auto rnd = rndGenerator.Gaus();
    h.Fill(rnd);
}
```

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

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

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

My Histo

myHisto
Entries 1000
Mean 0.02680
Std Dev 1.038

Sharing in SWAN

The screenshot shows the 'Share Project' dialog box over a background of the SWAN interface. The dialog box title is 'Share Project'. It displays the message 'You are sharing: Super Real Analysis with TOTEM data'. Below this is a search bar with the placeholder 'Start typing to add names...'. Under the heading 'Shared with', two users are listed: 'Danilo Piparo (danilo)' and 'Enric Tejedor Saavedra (enric)'. At the bottom of the dialog are 'Stop Sharing' and 'Update' buttons.

SWAN > My Projects > Super Real Analysis with TOTEM data

Super Real Analysis with TOTEM data

NAME ▾

DistilDistribution.ipynb

dataset.root

You are sharing:
Super Real Analysis with TOTEM data

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

Start typing to add names...

Shared with

Danilo Piparo (danilo)

Enric Tejedor Saavedra (enric)

Stop Sharing

Update

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

The screenshot shows the SWAN interface with the 'Share' tab selected. The top navigation bar includes 'Projects', 'Share', 'CERNBox', and other icons. The main content area has two sections: 'Projects shared with me' and 'Projects shared by me'. Both sections have headers with dropdown menus for 'NAME', 'SIZE', 'SHARED BY', and 'DATE'. The 'Projects shared with me' section lists one item: 'UP2University Pilot' (shared by 'Empty' at 7 minutes ago). The 'Projects shared by me' section lists two items: 'Higgs Boson discovery' (shared with '2 people/groups' at 18 hours ago) and 'Super Real Analysis with TOTEM data' (shared by 'diogo' at 19 hours ago).

SWAN > Share

Projects shared with me

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

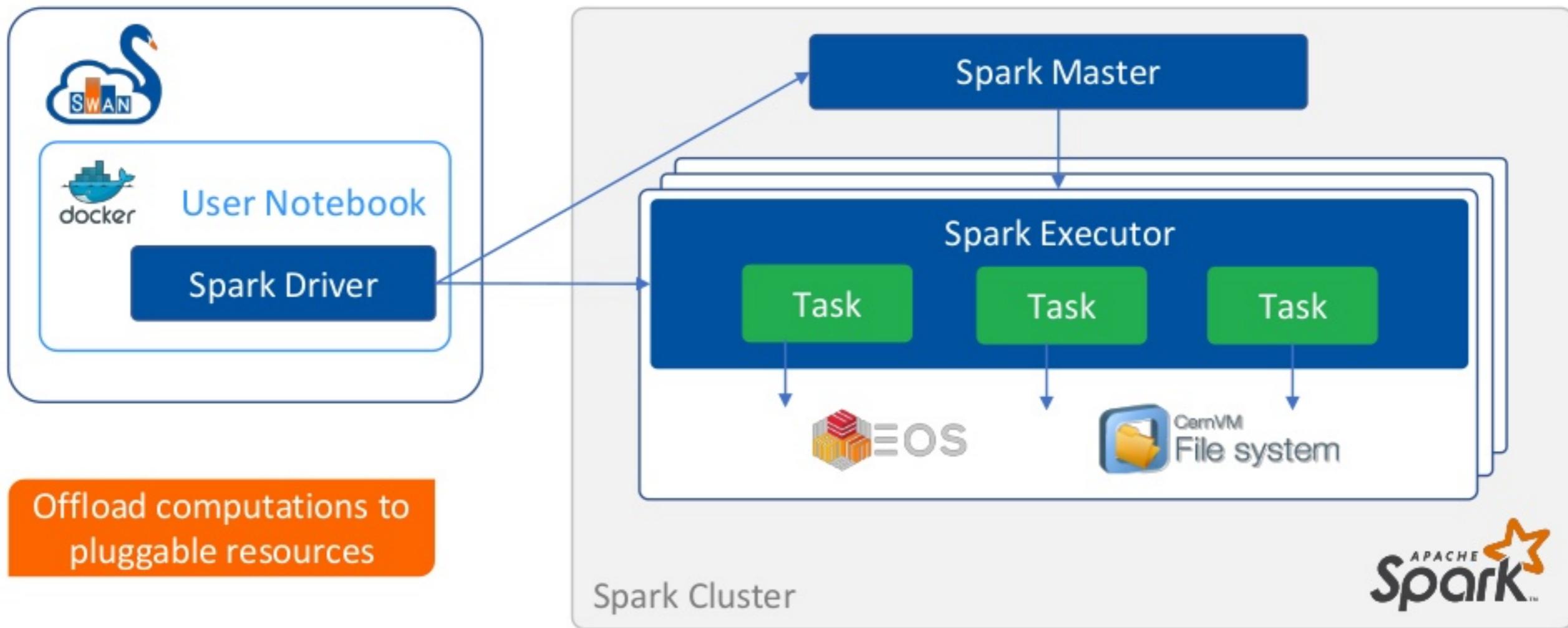
Projects shared by me

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

CERN

Integration with Spark



Spark Connector

The screenshot shows a Jupyter Notebook interface with the title "Spark > physics_analysis_using_swam_spark_template (success)". The notebook contains sections for "Integration of SWAN with Spark clusters" and "Step 1 - Acquire the necessary credentials to access the Spark cluster". It includes a code cell (In [1]) with Python code for generating credentials:

```
In [1]:  
import getpass  
import os, sys, re  
  
print("Please enter your password")  
ret = os.system("echo \\$s | xinit" + re.escape(getpass.getpass()))  
  
if ret == 0: print("Credentials created successfully")  
else:     sys.stderr.write("Error creating credentials, return code: %s\n" % ret)
```

The screenshot shows a Jupyter Notebook interface with the title "Spark > Spark_Simple (success)". A configuration dialog box titled "Spark clusters connection" is open on the right, showing options for connecting to a cluster named "hadalytic". The dialog includes fields for "Add a new option" (with placeholder "Write the option name..."), "Bundled configurations" (with an unchecked checkbox for "Include NXCALB options"), and a "Selected configuration" section with three radio button options:

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

A large orange callout box with the text "Configure Spark and connect to cluster with a click" points to the "Connect" button at the bottom right of the dialog.

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		Unknown	-
	7	foreach	COMPLETED	32 / 32	5 minutes ago	1m 20s

A decorative graphic on the left side of the slide consists of numerous thin, diagonal lines of varying lengths and colors, ranging from dark purple to light blue. These lines are arranged in a way that suggests motion or a data visualization.

Physics Data Use Case

#SAISExp1

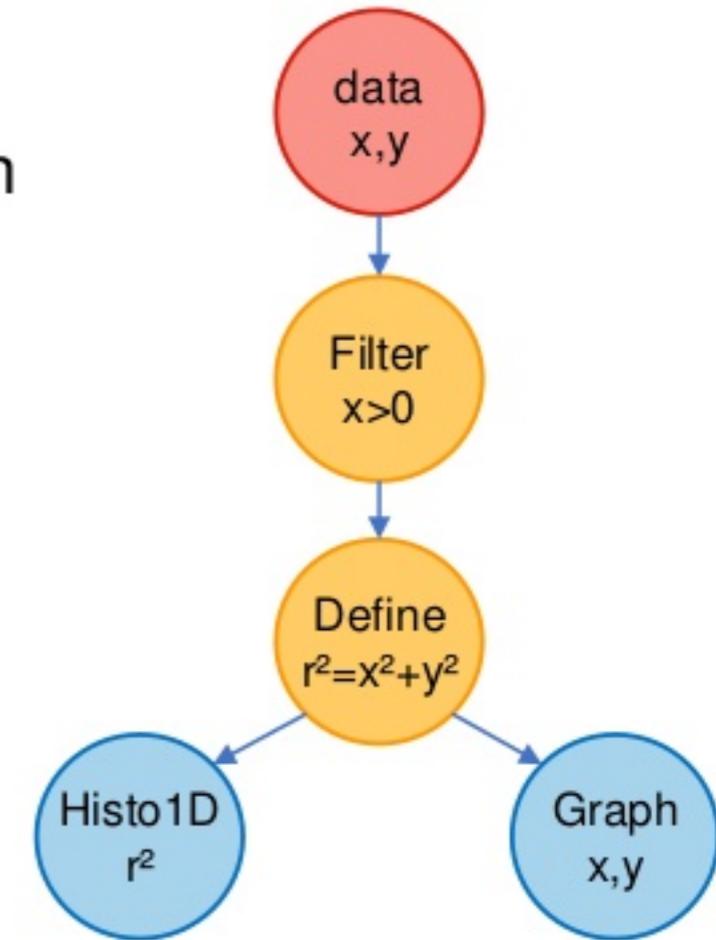
The HEP DataFrame

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

```
df = RDataFrame(dataset)

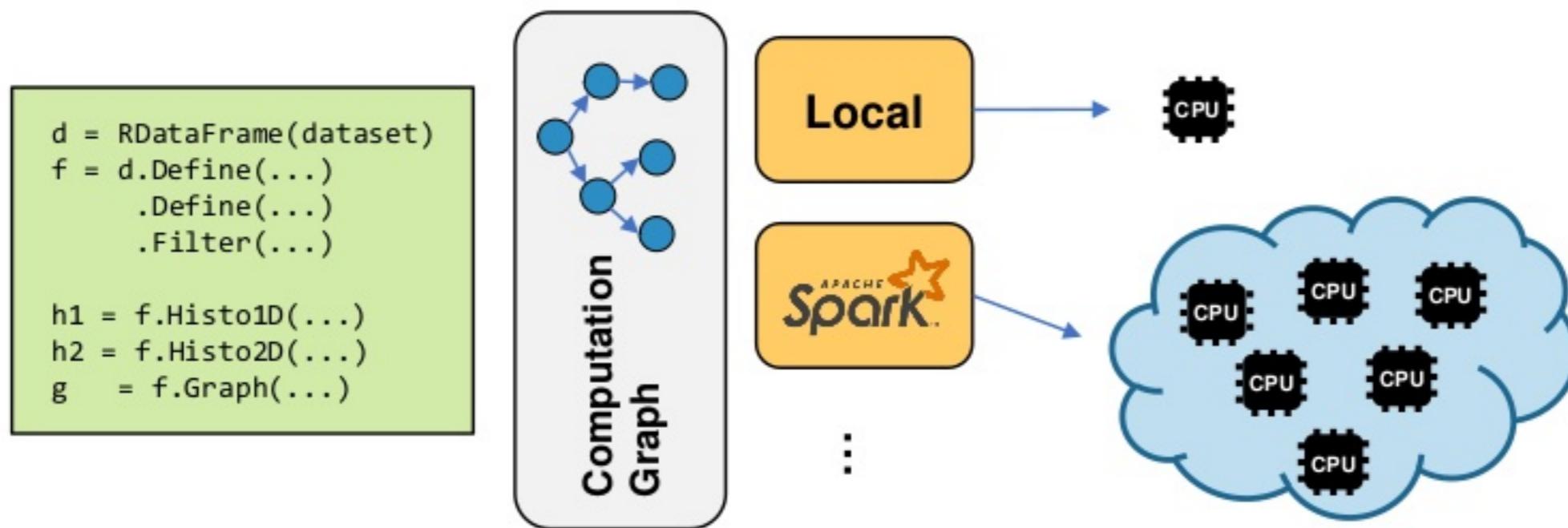
df2 = df.Filter('x > 0')
      .Define('r2', 'x*x + y*y')

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



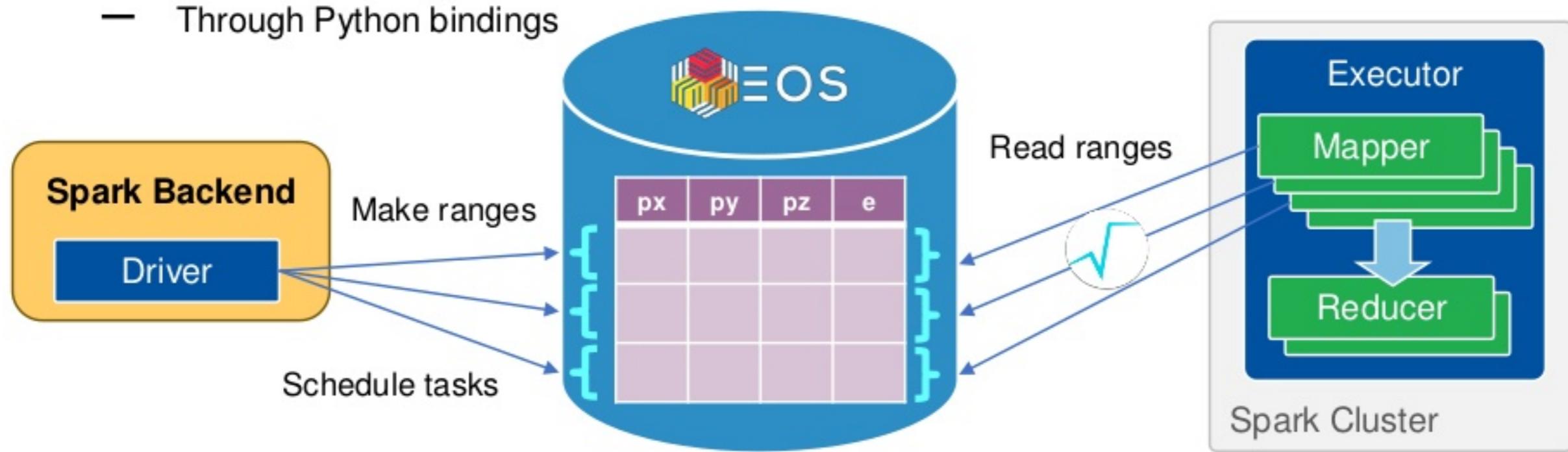
Distributed RDataFrame

- Exploratory work to parallelise RDataFrame computations with multiple backends



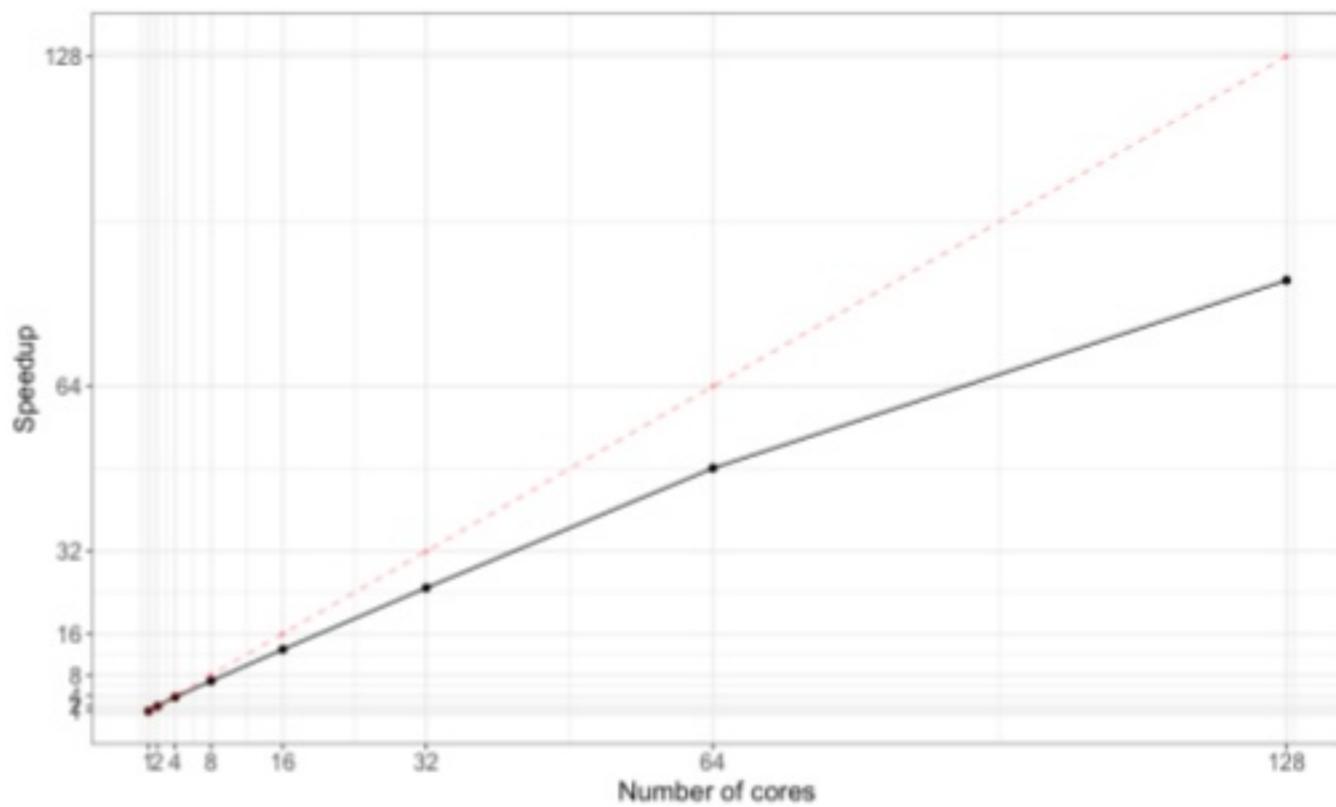
Spark Backend for RDataFrame

- **Map-reduce** workflow where every mapper runs the RDataFrame computation graph on a **range** of collision events
- Run **analysis in C++ with Spark**
 - Through Python bindings



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
- **Get to physics results faster!**





Controls Data Use Case

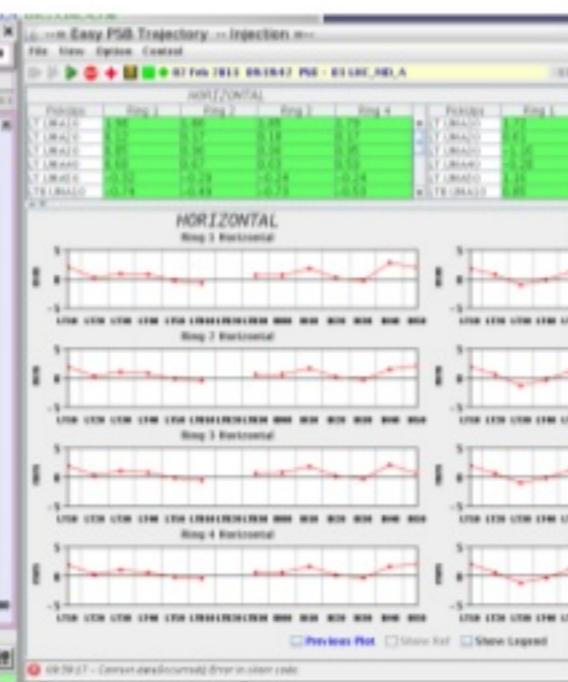
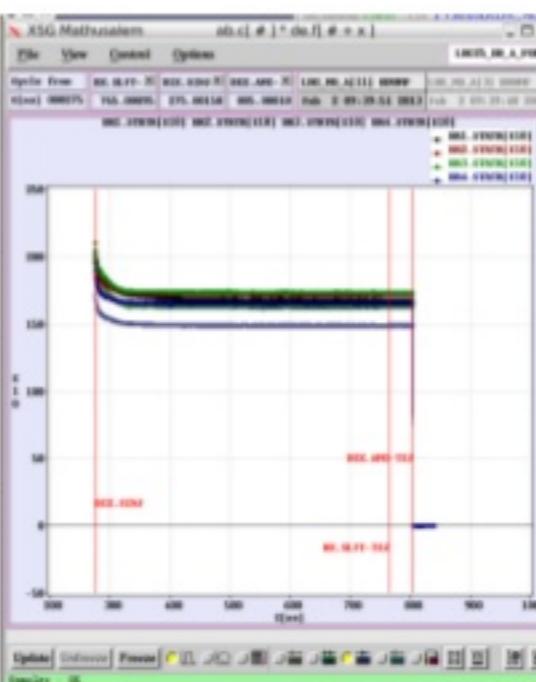
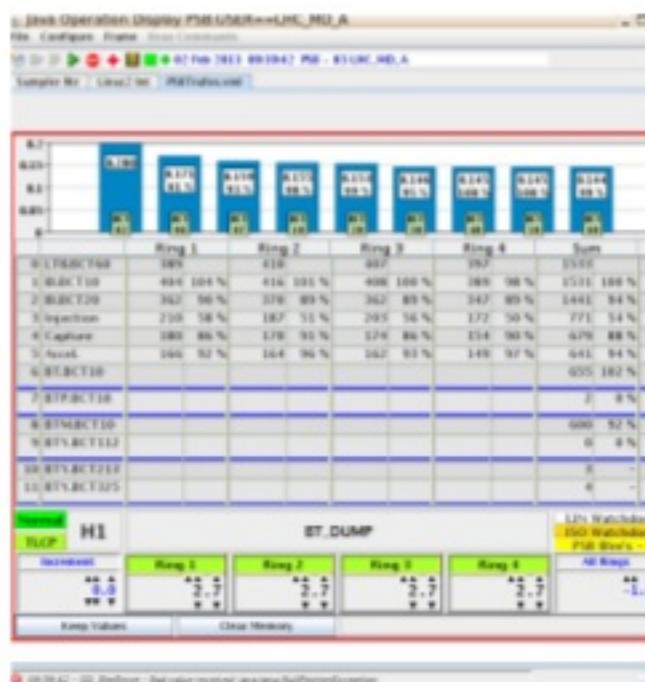
#SAISExp1



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

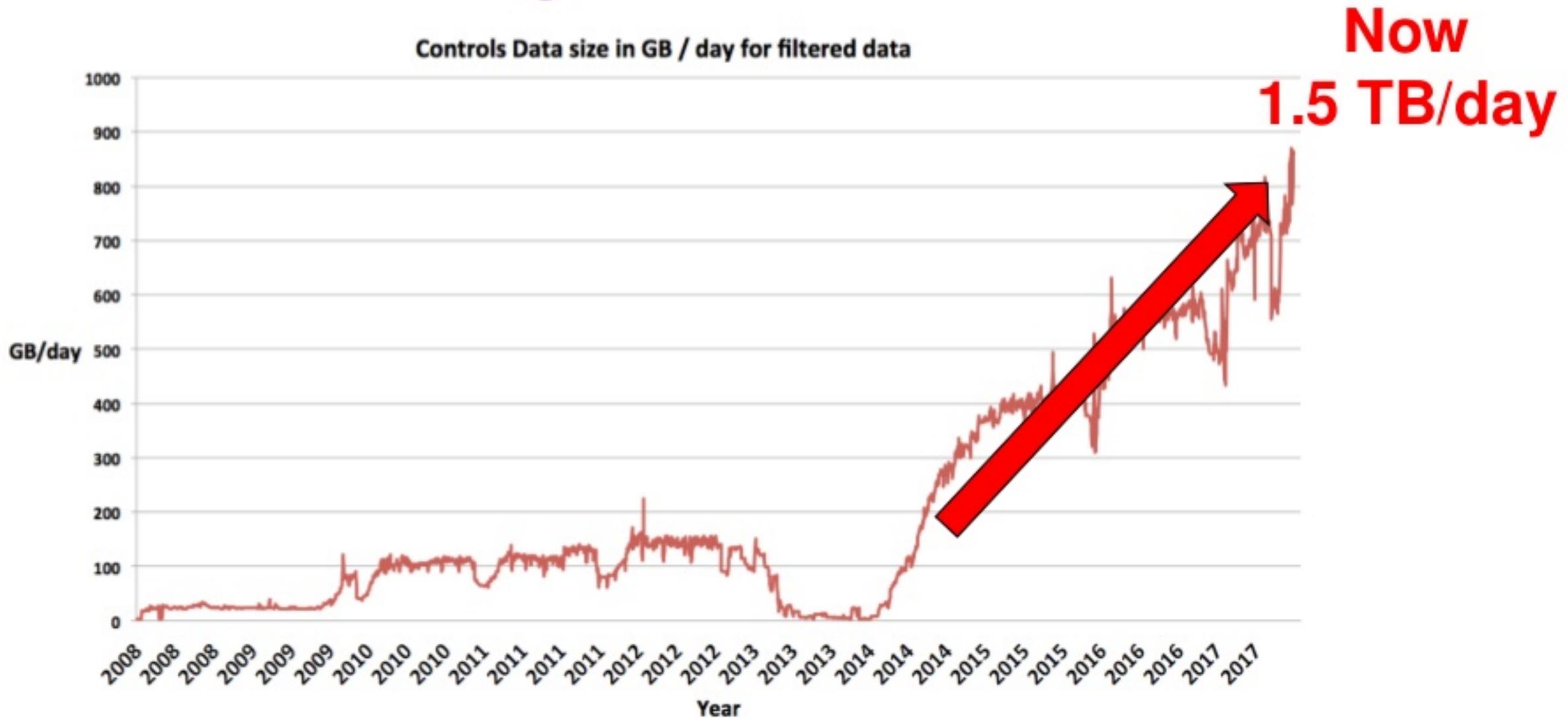
Hardware Controls

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



PSB:MULTIP_OP - PSB.USER.LHC.MD_A					
File Edit View References Commands Control Programs Help					
25 May 2011 15:02:22 PSB - ZERO, B3					
Simple view					
POW	Status	CCV	ACN	Unit	
BTR.BNO412L3	0e	1.07	0.14	A	
BTR.BNO412L5	0e	-0.54	-0.47	A	
BTR.BNO412L3	0e	-1.09	-0.94	A	
BTR.BNO412L3	0e	1.31	0.99	A	
BTR.BNO816L3	0e	-2.63	-1.85	A	
BTR.BNO816L3	0e	-3.69	-3.26	A	
BTR.BNO816L3	0e	-2.69	-1.66	A	
BTR.BNO816L3	0e	+1.77	-0.66	A	
POW	Status	CCV	ACN	Unit	
BTR.BNO412	0e	0.11	0.15	A	
BTR.BNO413	0e	0.91	0.01	A	
BTR.BNO413	0e	7.46	7.43	A	
BTR.BNO413	0e	7.04	6.29	A	
BTR.BNO413	0e	1.06	0.77	A	
BTR.BNO1213	0e	6.03	5.97	A	
BTR.BNO1213	0e	6.83	10.40	A	
POW	Status	CCV	ACN	Unit	
BTR.BSK214	0e	-5.26	-4.89	A	
BTR.BSK214	0e	0.51	0.35	A	
BTR.BSK214	0e	-3.07	-2.66	A	
BTR.BSK214	0e	2.99	2.66	A	
BTR.BSK413	0e	-2.19	-1.92	A	
BTR.BSK413	0e	-0.26	-0.15	A	
BTR.BSK413	0e	16.68	16.75	A	
BTR.BSK413	0e	12.77	12.39	A	
BTR.BSK413	0e	9.96	9.16	A	
BTR.BSK413	0e	17.99	18.10	A	
BTR.BSK413	0e	7.12	6.85	A	
BTR.BSK414	0e	1.93	1.69	A	
BTR.BSK414	0e	2.92	2.85	A	
BTR.BSK414	0e	0.99	1.23	A	
POW	Status	CCV	ACN	Unit	
BTR.DSK40	0e	-0.68	0.00	A	
BTR.DSK40	0e	0.65	0.03	A	
BTR.DSK40	0e	0.63	0.06	A	
BTR.DSK40	0e	0.78	0.00	A	
PEIM-V	Pulse	CCV	ADN	Start	Train
BTR.AQUAD	Enabled	30	10	BTR.MIO-ST	1-KHz
BTR.ASEXT	Enabled	25	25	BTR.MIO-ST	1-KHz
BTR.ACST	Enabled	50	10	BTR.MIO-ST	1-KHz
BTR.ACSC	Enabled	356	356	BTR.SCY-ST	1-KHz

Controls Log 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 rely 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()[:5]
```

```
Out[2]:
```

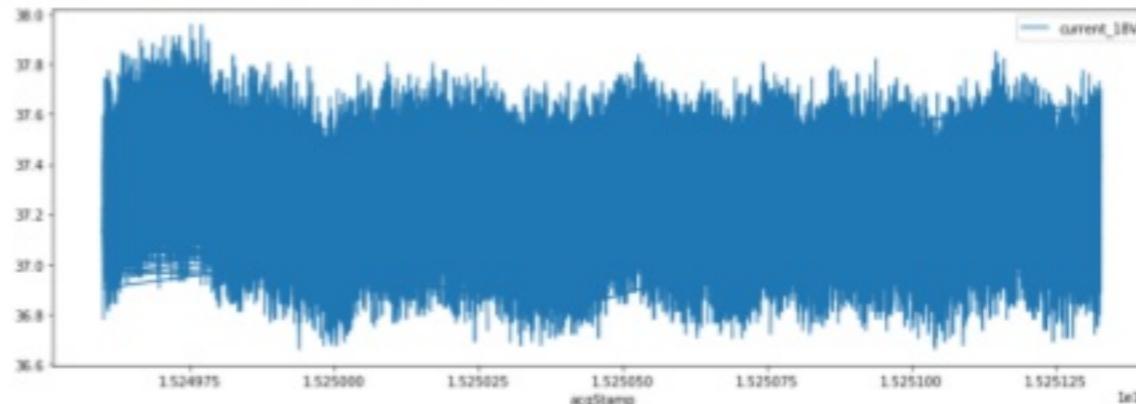
	acqStamp	voltage_18V	current_18V	device	pt100Value
0	1524960103132865000	NaN	37.301794	RADMON.PS-10	106.578911
1	1524960254134564000	NaN	NaN	RADMON.PS-10	107.246742
2	1524960322134942000	NaN	37.360940	RADMON.PS-10	106.504707
3	1524960353135244000	20.099066	NaN	RADMON.PS-10	107.069654
4	1524960911140548000	20.111261	37.888135	RADMON.PS-10	108.578911

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>
```





Summary

#SAISExp1

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
- 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 **easy access** to more processing power
 - Higher-level programming models
 - Pluggable computing resources

More on CERN and Spark: stay tuned for [Luca's](#) and [Prasanth's presentations](#)



Backup Slides

#SAISExp1

JupyterLab

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

The screenshot shows the JupyterLab interface. On the left is a sidebar with tabs for Files, Running, Commands, Cell Tools, Tabs, and Help. The Files tab shows a list of notebooks: Data.ipynb (an hour ago), Fasta.ipynb (a day ago), Julia.ipynb (a day ago), Lorenz.ipynb (seconds ago), R.ipynb (a day ago), iris.csv (a day ago), lightning.json (9 days ago), and lorenz.py (3 minutes ago). The Running tab shows a terminal window titled "Terminal 1" and a console window titled "Console 1". The main area contains a notebook cell with the title "Lorenz.ipynb". The cell displays text about the Lorenz system of differential equations, followed by a code cell:

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

Below the code cell is an "Output View" panel showing sliders for parameters sigma (10.00), beta (2.07), and rho (28.00). To the right is a code editor window titled "lorenz.py" containing the following Python code:

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim([-25, 25])
    ax.set_ylim([-35, 35])
    ax.set_zlim([5, 55])

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random((N, 3))
```