

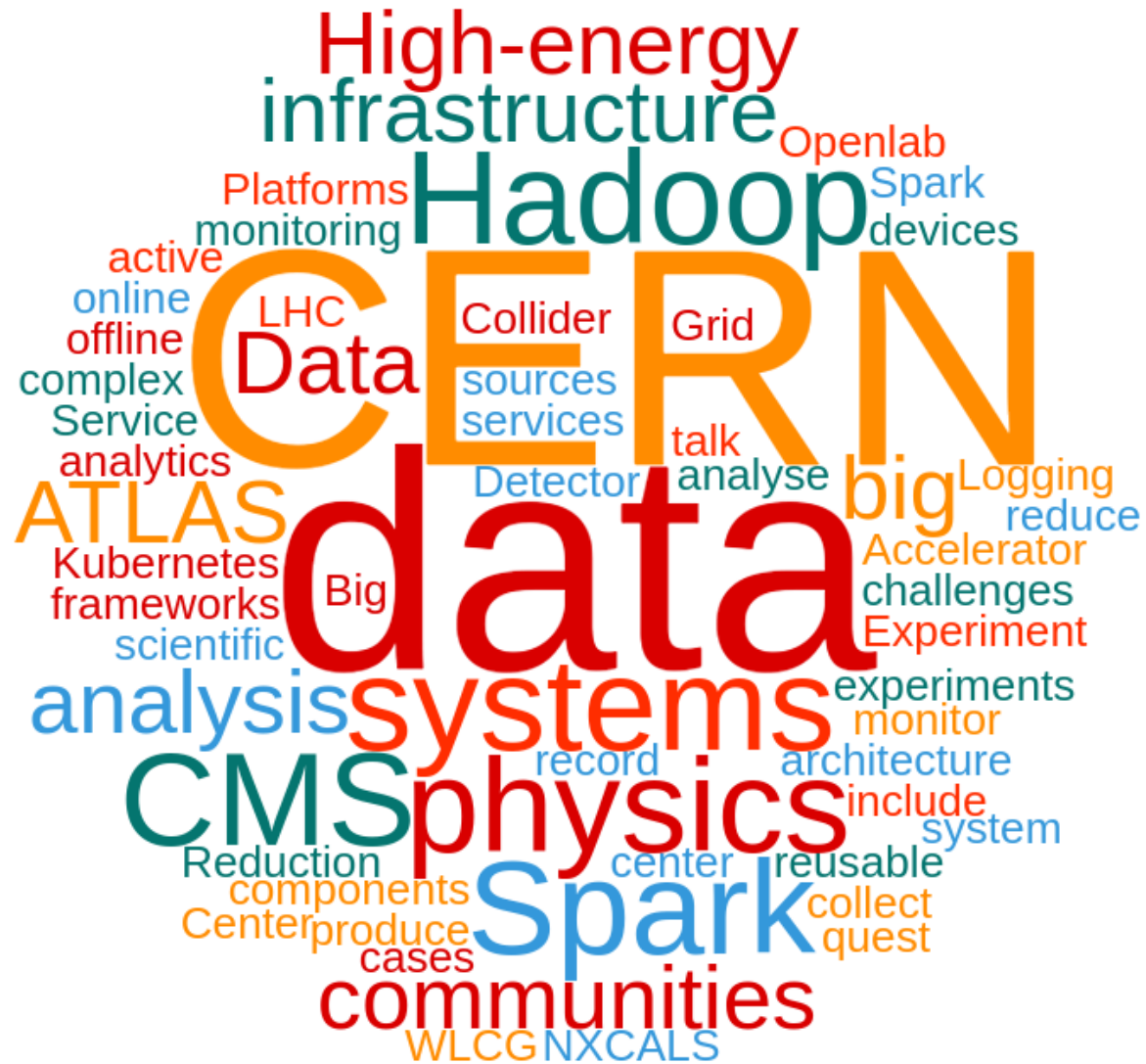


Big Data at CERN

2nd International PhD School on Open Science Cloud

Vaggelis Motesnitsalis

20 September 2018



Who am I?

I am...



Big Data Engineer as openlab Intel
Research Fellow



Former Big Data Devops Support Engineer
and Escalation Engineer @AWS



Research Fellow from 2017 - today



MSc Distributed Systems @Imperial College London



4+ years of Big Data Experience



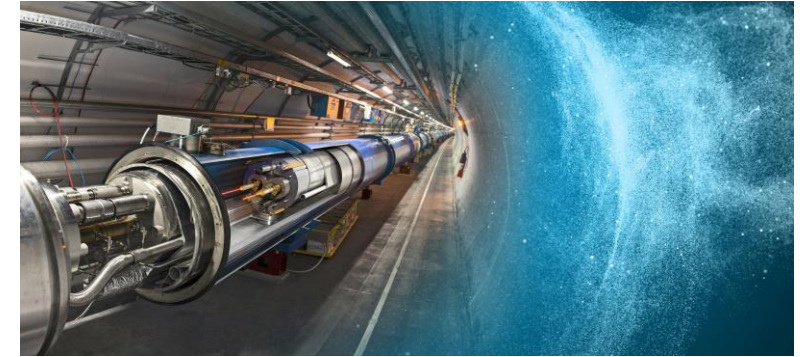
Studies at King's College London and Aristotle
University of Thessaloniki

emotes@cern.ch

Physics Analysis at CERN

Physics Analysis at CERN

Until today, the vast majority of high energy physics analysis is done with the **ROOT Framework** which uses physics data that are stored in ROOT format files, within the **EOS Storage Service** of CERN.



ROOT Data Analysis Framework

A modular scientific software framework which provides all the functionalities needed to deal with big data processing, statistical analysis, visualization and file storage. It is mainly written in C++ but also integrated with Python and R.



EOS Storage Service

A disk-based, low-latency storage service with a highly-scalable hierarchical namespace, which enables data access through the XRootD protocol. It provides storage for both physics and user use cases via different service instances such as EOSPUBLIC, EOSATLAS, EOSCMS.



Operating the LHC

Operating the LHC



2 Beams



Only 2 nanograms of Hydrogen per day



7 TeV per proton



Ultra-high Vacuum



350 MJ per beam – equivalent to a
TGV train travelling with 150 km/h



11245 circuits every second

How many collisions?

Collisions



2 beams



100 billions protons per bunch



4 collision points – 4 experiments



20 – 40 collisions per bunch interaction



2808 bunches (max)



1 collision every 25 nanoseconds

4 Collision Points

1 Billion collisions per second
(at each collision point)

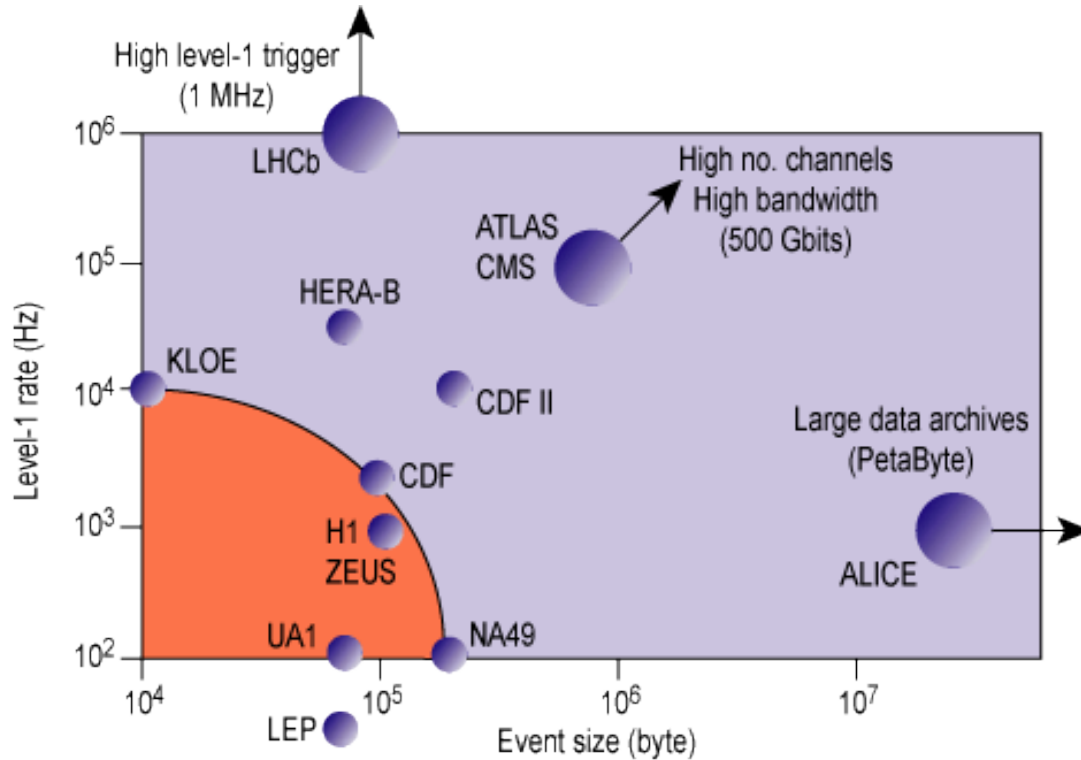
1-2 MB to store a collision

What is the data production rate of the LHC?

4 – 8 PBs

Filtering and Triggers

Filtering and Triggers



Trigger Level 1: 100K collisions out of 1B



Trigger Level 2: 10K events out of 100K



Trigger Level 3: 100-200 events out of 10K

Data

Data



12.3 PB in a single month
(October 2017)



Different types of data:

physics,
accelerator metrics,
infrastructure logging, etc.



>250 PB at CERN Data Center



Stored on a **custom storage** service [EOS]



Data **formats** include:

ROOT for Physics
Parquet, JSON, AVRO in Hadoop



Tapes are used for long-term storage

The Worldwide LHC Computing Grid

WLCG

Worldwide LHC Computing Grid



The Worldwide LHC Computing Grid (WLCG) is a global collaboration of more than 170 institutions in 42 countries which provide resources to store, distribute and analyse the PBs of LHC Data



WLCG
Worldwide LHC Computing Grid

EOS Storage Service

EOS Storage Service

Service Characteristics



Disk-based



Based on the XRootD Protocol



Low Latency



Storage for both physics and user use cases



Highly Scalable



Different service instances such as
EOSPUBLIC, EOSATLAS, EOSCMS

Hadoop Service at CERN

Hadoop and Spark Service at CERN

The Service in Numbers



6 Clusters



20+ TB Memory



110+ Nodes



3100+ Cores



14+ PBs Storage



4 TBs/day Data Growth

Hadoop and Spark Service at CERN

Running the service



“vanilla” Apache Hadoop Distribution



High Availability



Puppet for Administration



Daily backups to Tapes



Kerberos for Authentication

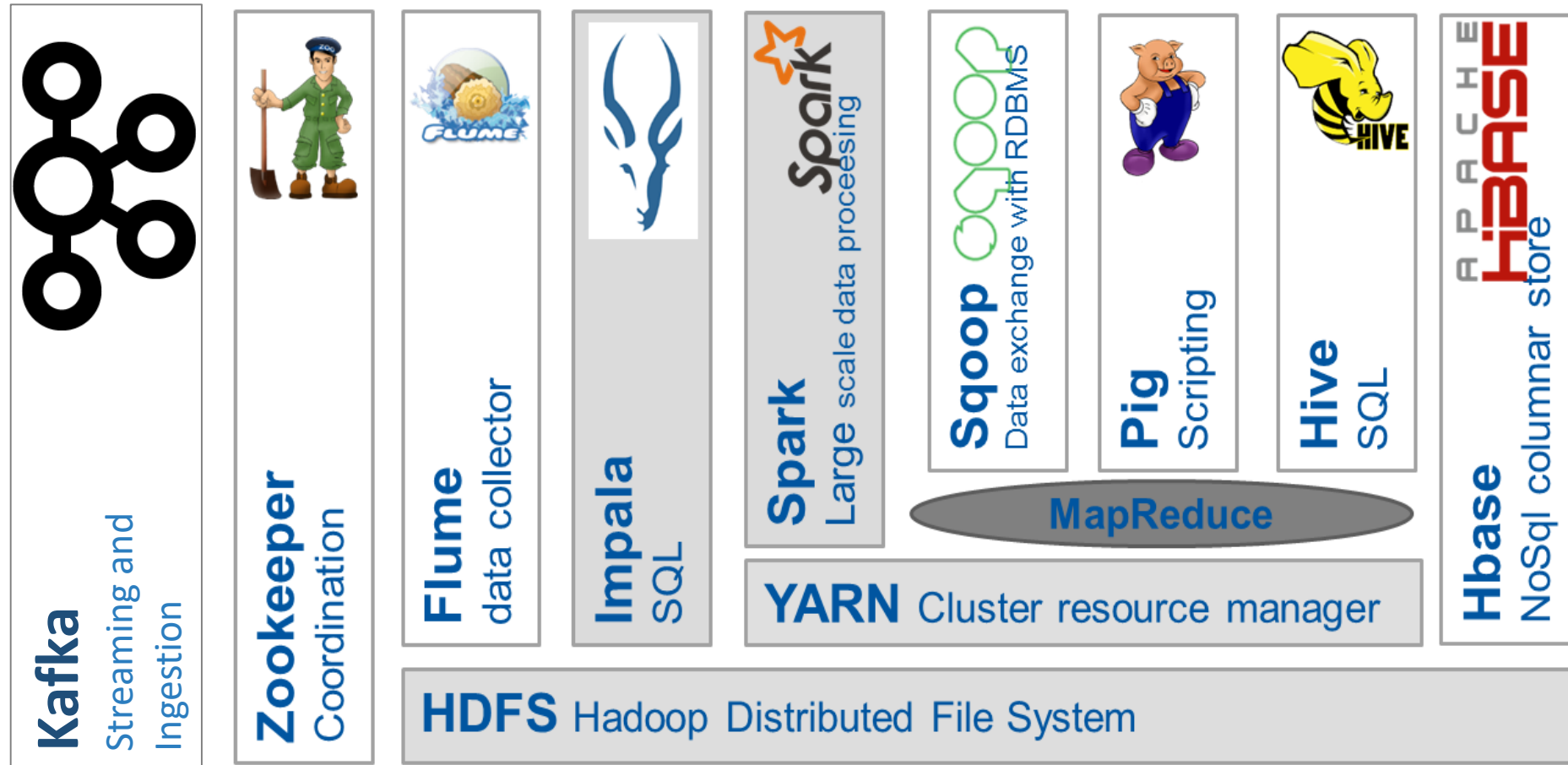


Accessible by client nodes and Jupyter Notebooks



Hadoop and Spark Service at CERN

Service Components



SWAN Service at CERN

SWAN Service

Hosted Jupyter Notebooks for Data Analysis



Web-based interactive analysis
using PySpark in the cloud



Collaboration between
EP-SFT, IT-ST, and IT-DB



No need to install software



Combines code, equations, text and
visualisations



Direct access to the EOS and HDFS



Fully Integrated with IT Spark and
Hadoop Clusters

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

SWAN Service

FILE EDIT VIEW INSERT CELL KERNEL HELP Trusted Python 2

Add column as create temporary table view

```
In [17]: sls_metrics = sls_raw.withColumn('status', when(col('data.service_status')== 'available', 2).when(col('data.service_status')== 'degraded', 1).otherwise(0))
        sls_metrics.createOrReplaceTempView("sls_metrics")
```

Do the heavylifting in spark and collect aggregated view to panda DF

```
In [19]: sls_pandas = spark.sql("select hour(from_unixtime(metadata.timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')) as hr, metadata.producer, avg(status) as avg from sls_metrics group by hour(from_unixtime(metadata.timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')), metadata.producer").toPandas()
```

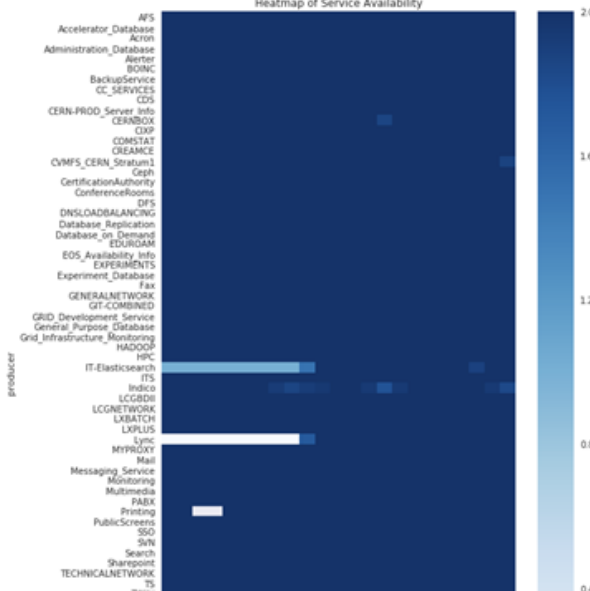
Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
9	toPandas	COMPLETED	2/2	218 / 218	6 minutes ago	14s

Visualize with seaborn

```
In [22]: # heatmap of service availability
        plt.figure(figsize=(9, 15))
        ax = sns.heatmap(sls_pandas.pivot(index='producer', columns='hr', values='avg'), cmap="Blues")
        ax.set_title("Heatmap of Service Availability")

Out[22]: Text(0.5,1,u'Heatmap of Service Availability')
```

Heatmap of Service Availability



Other Tools

Bridging the Gap

Problem Definition



Hadoop – XRootD Connector

Connecting XrootD-based Storage Systems with Hadoop and Spark



A Java library that connects to the XRootD client via JNI



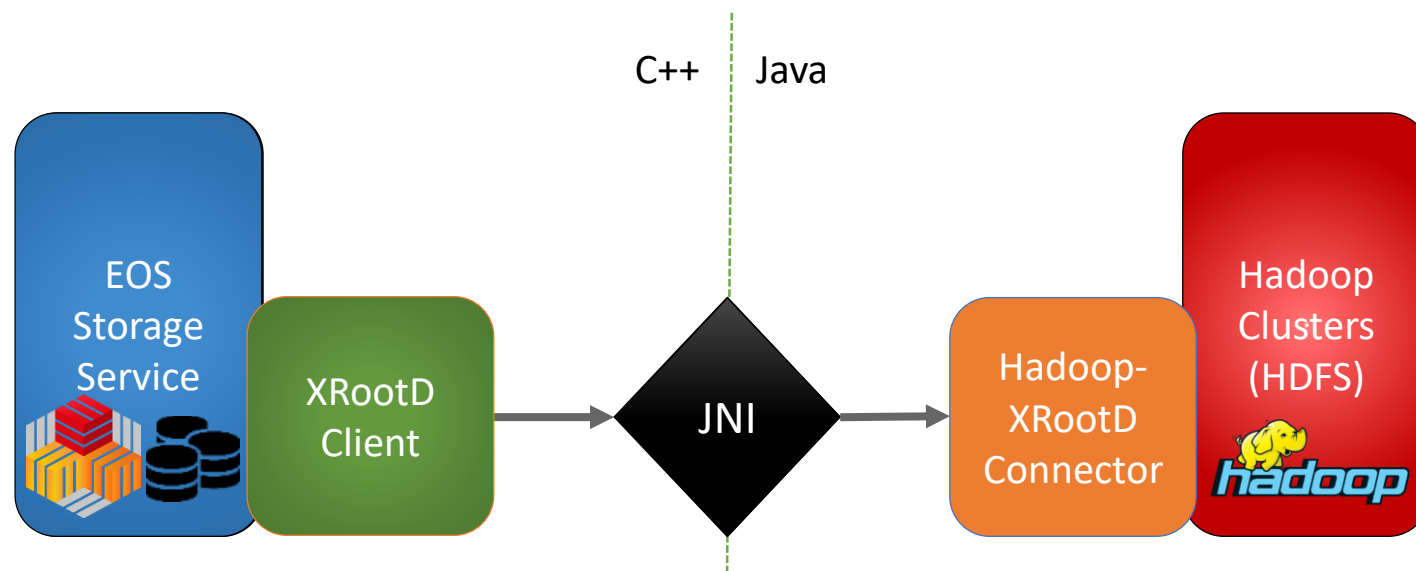
It reads files from the EOS Storage Service directly



Slower Read – Broader Data Access



Supports Kerberos and GRID Certificate Authentication



<https://github.com/cerndb/hadoop-xrootd>

spark-root

Importing Root files in Apache Spark



A Scala library which implements DataSource for Apache Spark



Spark can read ROOT TTrees and infer their schema



Root files are imported to Spark Dataframes/Datasets/RDDs



Developed by DIANA-HEP

<https://github.com/diana-hep/spark-root/>

Spark on Kubernetes Service

Spark on Kubernetes Service

Leveraging the Kubernetes support in Spark 2.3



Prototype of Spark on Kubernetes over OpenStack and built spark images and tooling



Under active development

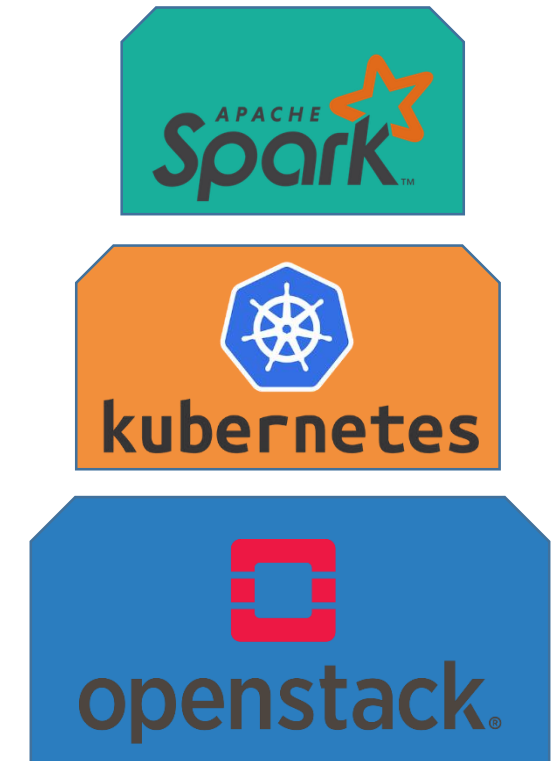


Based on already separated storage solution (EOS)

Storage

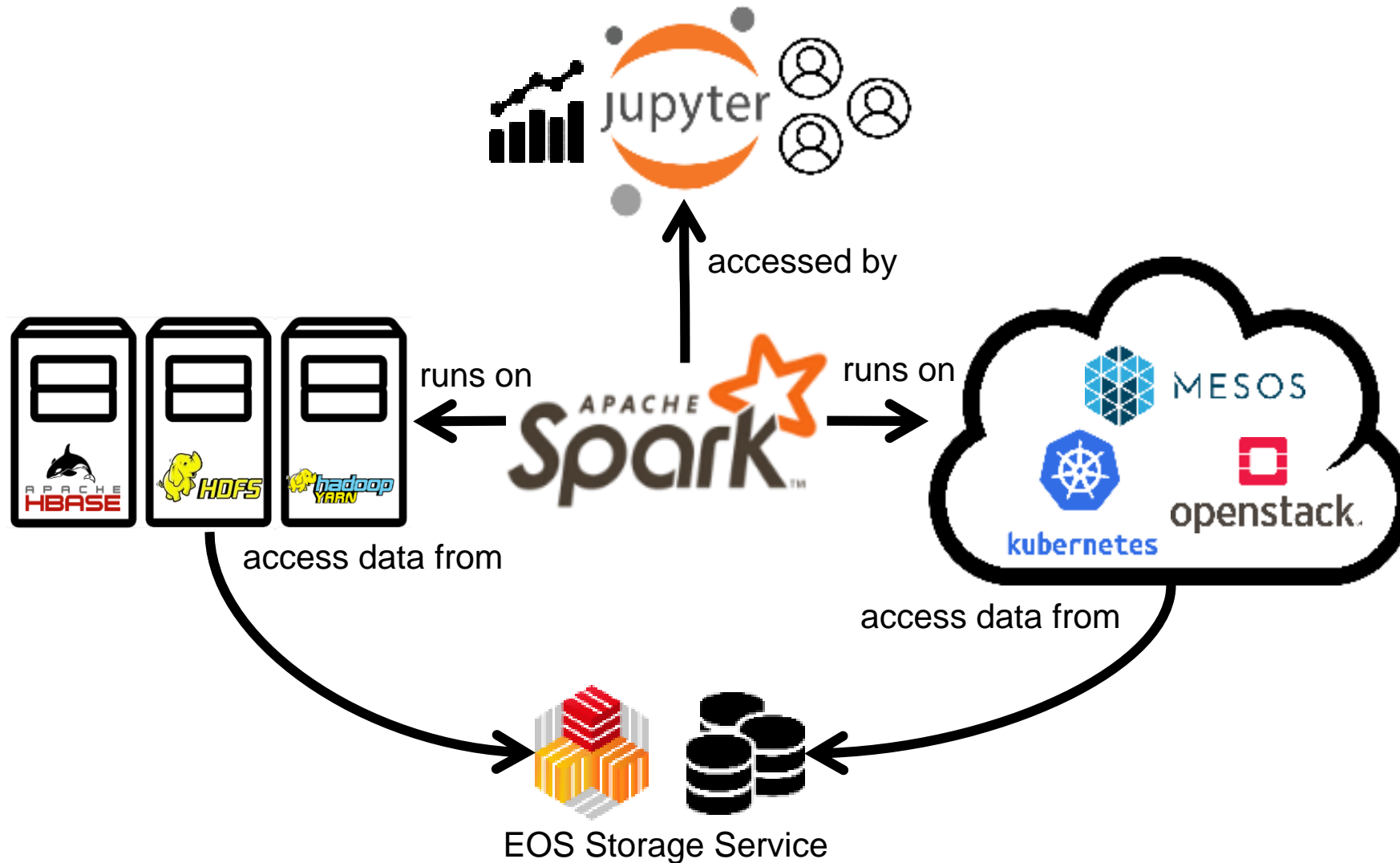


Compute



Future Plans

Analytics Platform



Selected Projects

Next Accelerator Logging (NXCALs)



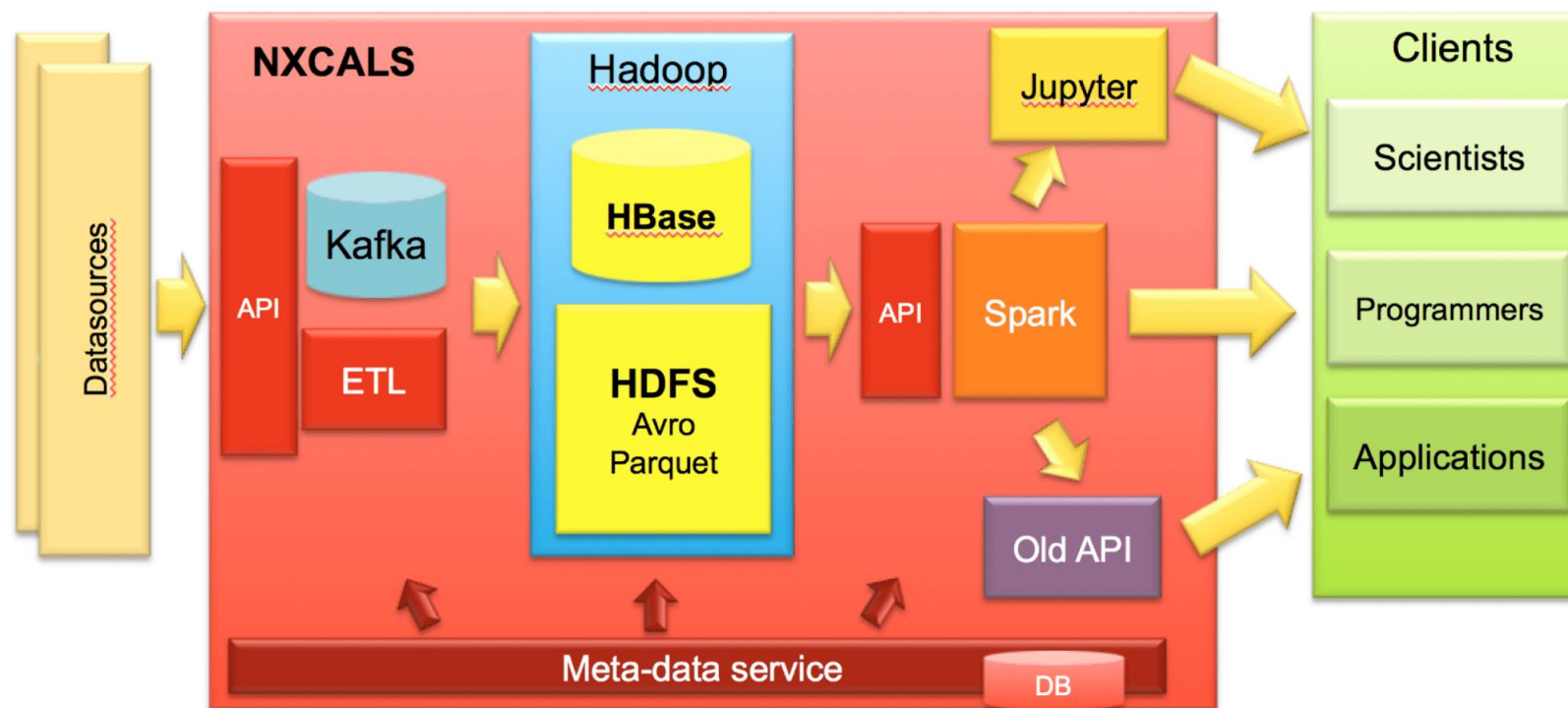
A control system with:
Streaming
Online System
API for Data Extraction



Critical for LHC
Operations



Runs on a dedicated
cluster



Credits: BE-CO-DS

Data Center and WLCG Monitoring Systems



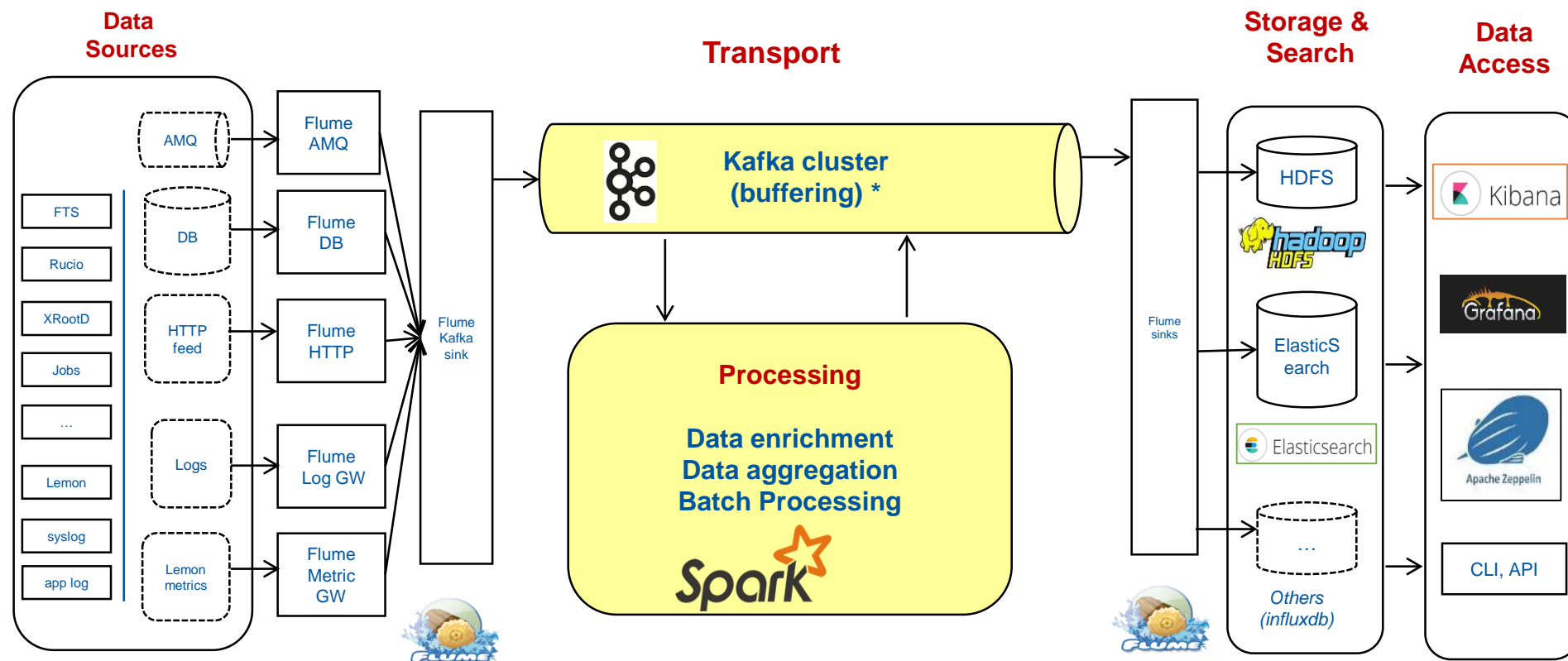
Critical for Data Center operations and WLCG



200M events/day
500 GB/day

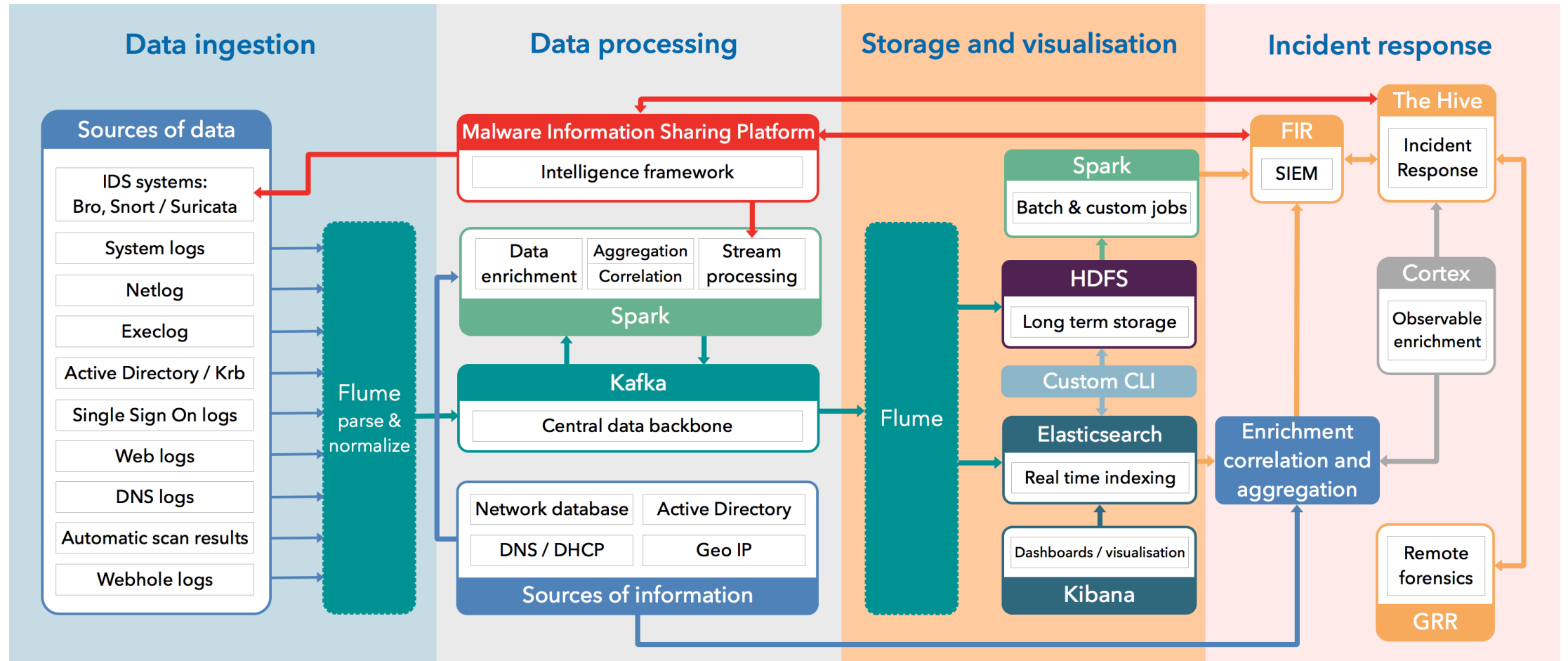


Proved effective in several occasions



Credits: IT-CM-MM

Computer Security Intrusion Detection



Credits: CERN security team, IT-DI

Physics Analysis with Spark

The CMS Data Reduction Facility

CMS Data Reduction Facility

Performing Physics Analysis and Data Reduction with Apache Spark



A Project by openlab, Intel, Fermilab and the CMS Experiment



We started scaling – goal is 1 PB in 5 hours



Investigate new ways to analyse physics data and improve resource utilization and time-to-physics



Root files are imported with « spark-root »

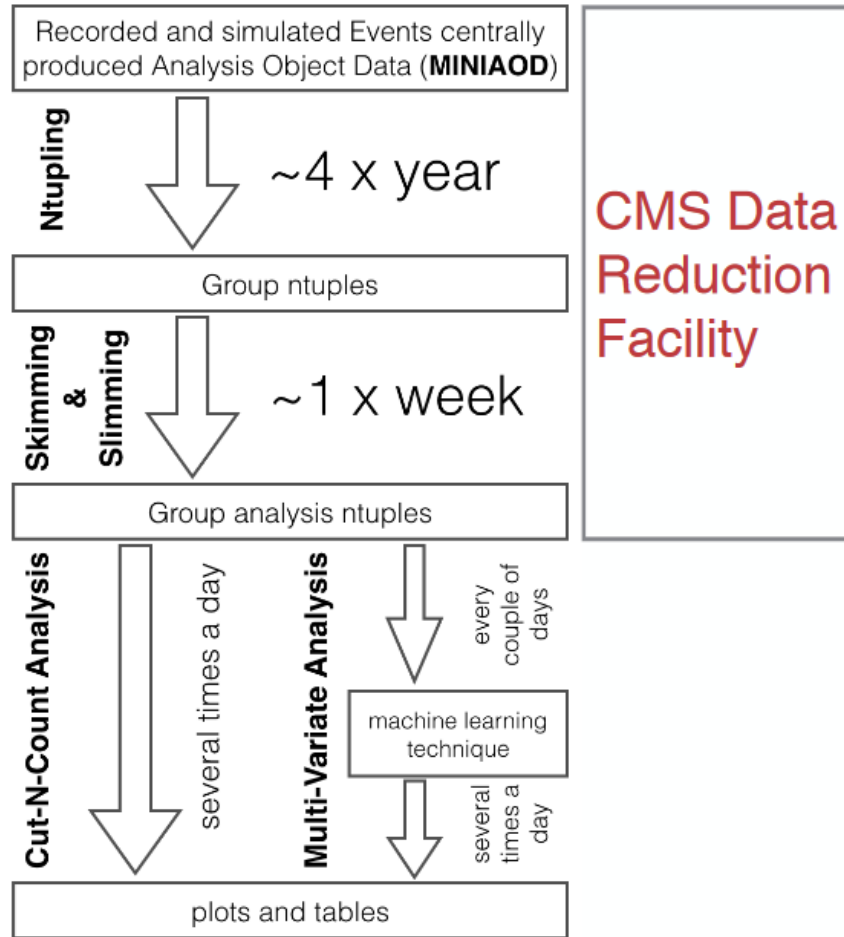


We now have fully functioning Analysis and Reduction examples tested over CMS Open Data



Files are accessed from the EOS Storage Service with the « Hadoop-XRootD Connector »

CMS Data Reduction Facility



Produce reduced data based on potential complicated user queries

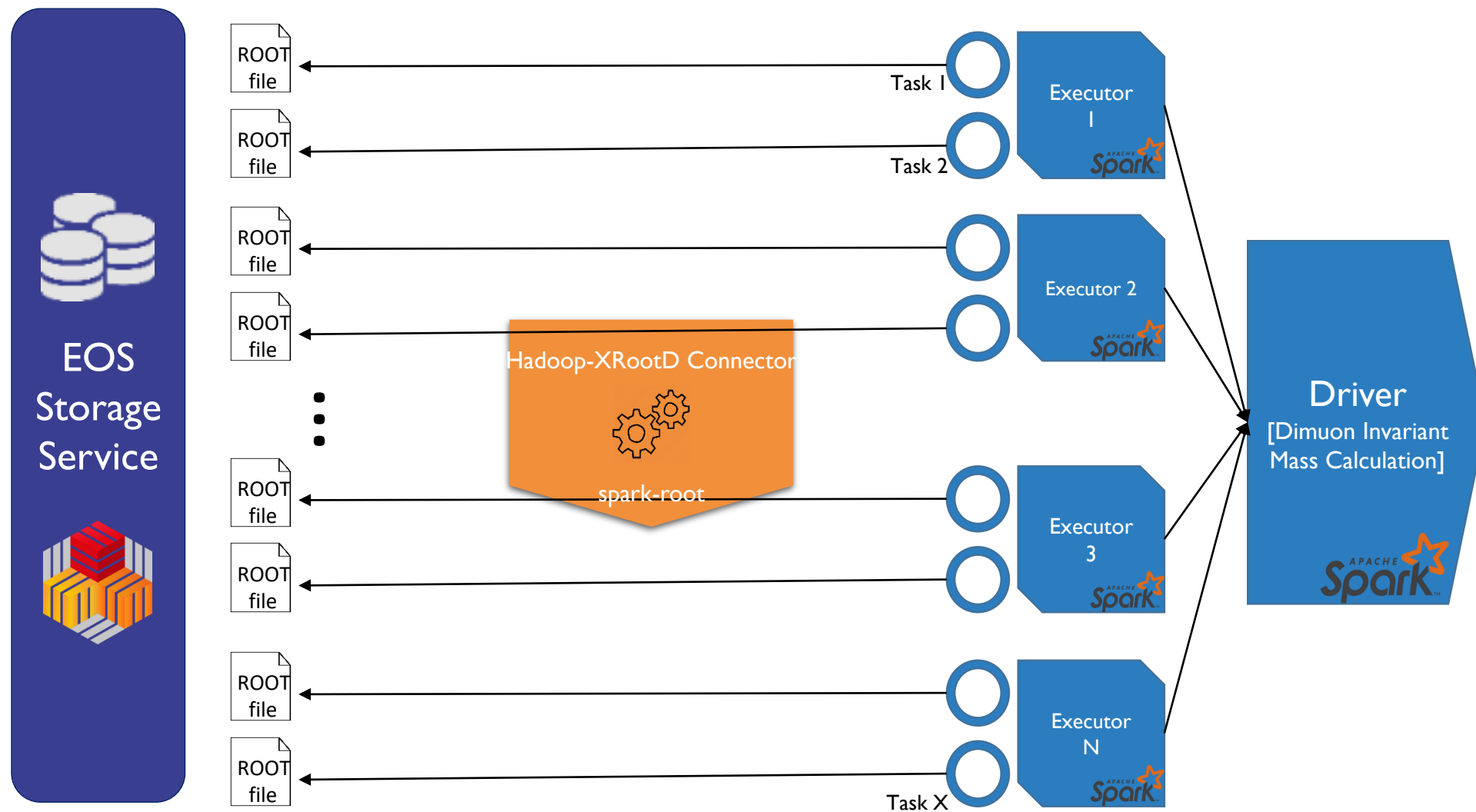


If successful, this type of facility could be a big shift for High Energy Physics



Make High Energy Physics more open to the Big Data community

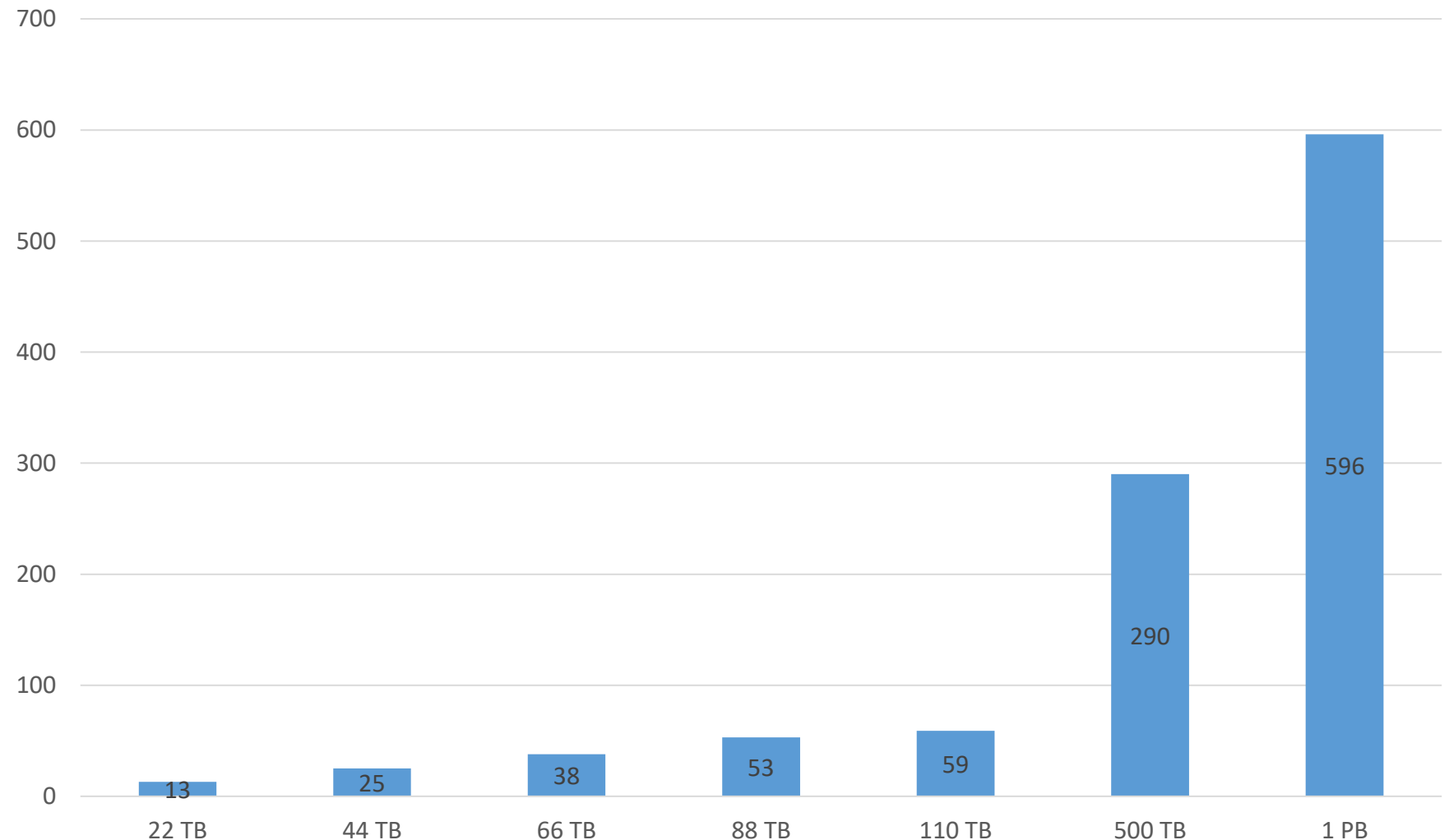
CMS Data Reduction Facility



CMS Data Reduction Facility





Results for different input size in minutes for Dimuon Mass Calculation


Input Size	Time
22 TB	13m
44 TB	25m
66 TB	38m
88 TB	53m
110 TB	59m
500 TB	4h50m
1 PB	9h54m

















Example

Example on Physics Analysis with SWAN

 DimuonMass_NanoAOD_Demo > DimuonMass_NanoAOD_Demo (autosaved)   

FILE EDIT VIEW INSERT CELL KERNEL HELP Not Trusted Python 2 

            Markdown  

Set python for pyspark

this is fixed in the next lcg release

```
In [1]: import os
os.environ["PYSPARK_PYTHON"]="/cvmfs/sft.cern.ch/lcg/views/LCG_93/x86_64-centos7-gcc62-opt/bin/python"
```

Setting up Spark

Start a pySpark session including Diana-Hep spark-root and histogrammar APIs

```
In [2]: %reset
import pyspark.sql
session = pyspark.sql.Session.builder \
    .master('spark://10.64.22.198:7077') \
    .appName('Demo') \
    .config('spark.jars.packages', 'org.diana-hep:spark-root_2.11:0.1.16,org.diana-hep:histogrammar-sparksql_2.11:1.0.4') \
    .config('spark.driver.extraClassPath', '/opt/hadoop/share/hadoop/common/lib/EOSfs.jar') \
    .config('spark.executor.extraClassPath', '/opt/hadoop/share/hadoop/common/lib/EOSfs.jar') \
    .config('py-files', 'helper.py') \
    .getOrCreate()

sqlContext = session

print 'SparkSQL session created'
```

Example on Physics Analysis with SWAN

Loading root files stored remotely on EOS via xrootd

Loading root files (NanoAOD CMS format) from CERN public EOS area via xrootd. Trees are read using the spark-root

```
In [3]: from pyspark.sql.functions import lit
        from samples import *

        DFList = []

        for s in samples:
            print 'Loading {} sample from EOS file'.format(s)
            dsPath = "root://eospublic.cern.ch/eos/opstest/cmspd-bigdata/"+samples[s]['filename']
            tempDF = sqlContext.read \
                .format("org.dianahep.sparkroot") \
                .option("tree", "Events") \
                .load(dsPath)\
                .withColumn("sample", lit(s))
            DFList.append(tempDF)
```

Loading SingleMuon sample from EOS file
Loading DYJetsToLL sample from EOS file
Loading WZ sample from EOS file

Access DataFrame content

Get list of columns in the DataFrame ("branches" of the equivalent ROOT TTree).

```
In [4]: DFList[0].printSchema()

root
 |-- run: integer (nullable = true)
 |-- luminosityBlock: integer (nullable = true)
 |-- event: long (nullable = true)
 |-- CaloMET_phi: float (nullable = true)
 |-- CaloMET_pt: float (nullable = true)
 |-- CaloMET_sumEt: float (nullable = true)
 |-- nElectron: integer (nullable = true)
 |-- Electron_deltaEtaSC: array (nullable = true)
 |    |-- element: float (containsNull = true)
 |-- Electron_dr03EcalRecHitSumEt: array (nullable = true)
```

Final Result

```
scala> val empty = Bin(40, 0, 100, {x: Float => x})
empty: org.dianahep.histogrammar.Binning[Float,org.dianahep.histogrammar.Counting,org.dianahep.histogrammar.Counting,org.dianahep.histogrammar.Counting] = <Binni
ng num=40 low=0.0 high=100.0 values=Count underflow=Count overflow=Count nanflow=Count>

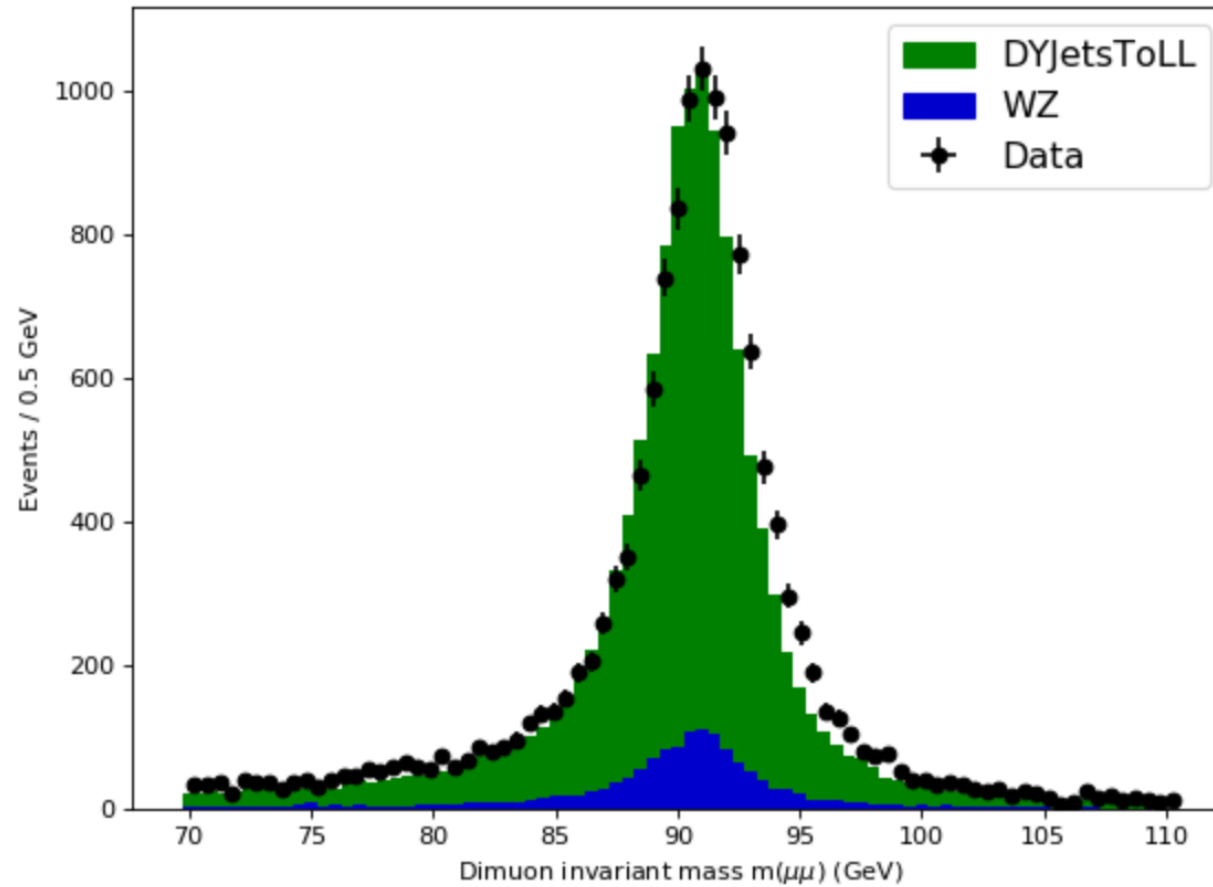
scala> val histo = muons.as[Seq[Float]].flatMap({case x => x}).rdd.aggregate(empty)(new Increment, new Combine)
recoMuons_muons RECO
[Log.apache.spark.sql.sources.Filter:@798f8f25
17/09/20 15:06:45 WARN ClosureCleaner: Expected a closure; got org.dianahep.histogrammar.Increment
17/09/20 15:06:45 WARN ClosureCleaner: Expected a closure; got org.dianahep.histogrammar.Combine
histo: org.dianahep.histogrammar.Binning[Float,org.dianahep.histogrammar.Counting,org.dianahep.histogrammar.Counting,org.dianahep.histogrammar.Counting] = <Binni
ng num=40 low=0.0 high=100.0 values=Count underflow=Count overflow=Count nanflow=Count>

scala> histo.println
0 8869.30
+-----+
underflow 0 |
[ 0 , 2.5 ) 8063 |*****|
[ 2.5 , 5 ) 5173 |*****|
[ 5 , 7.5 ) 1629 |*****|
[ 7.5 , 10 ) 379 |***|
[ 10 , 12.5 ) 130 |*|
[ 12.5 , 15 ) 60 |
[ 15 , 17.5 ) 26 |
[ 17.5 , 20 ) 19 |
[ 20 , 22.5 ) 8 |
[ 22.5 , 25 ) 5 |
[ 25 , 27.5 ) 4 |
[ 27.5 , 30 ) 7 |
[ 30 , 32.5 ) 2 |
[ 32.5 , 35 ) 3 |
[ 35 , 37.5 ) 2 |
[ 37.5 , 40 ) 1 |
[ 40 , 42.5 ) 1 |
[ 42.5 , 45 ) 3 |
[ 45 , 47.5 ) 1 |
[ 47.5 , 50 ) 0 |
[ 50 , 52.5 ) 0 |
[ 52.5 , 55 ) 1 |
[ 55 , 57.5 ) 2 |
[ 57.5 , 60 ) 0 |
[ 60 , 62.5 ) 0 |
[ 62.5 , 65 ) 0 |
[ 65 , 67.5 ) 0 |
[ 67.5 , 70 ) 0 |
[ 70 , 72.5 ) 1 |
[ 72.5 , 75 ) 0 |
[ 75 , 77.5 ) 0 |
[ 77.5 , 80 ) 0 |
[ 80 , 82.5 ) 0 |
[ 82.5 , 85 ) 0 |
[ 85 , 87.5 ) 0 |
[ 87.5 , 90 ) 1 |
[ 90 , 92.5 ) 1 |
[ 92.5 , 95 ) 0 |
[ 95 , 97.5 ) 0 |
[ 97.5 , 100 ) 0 |
overflow 7 |
nanflow 0 |
+-----+
```

OK, OK, one with better graphics

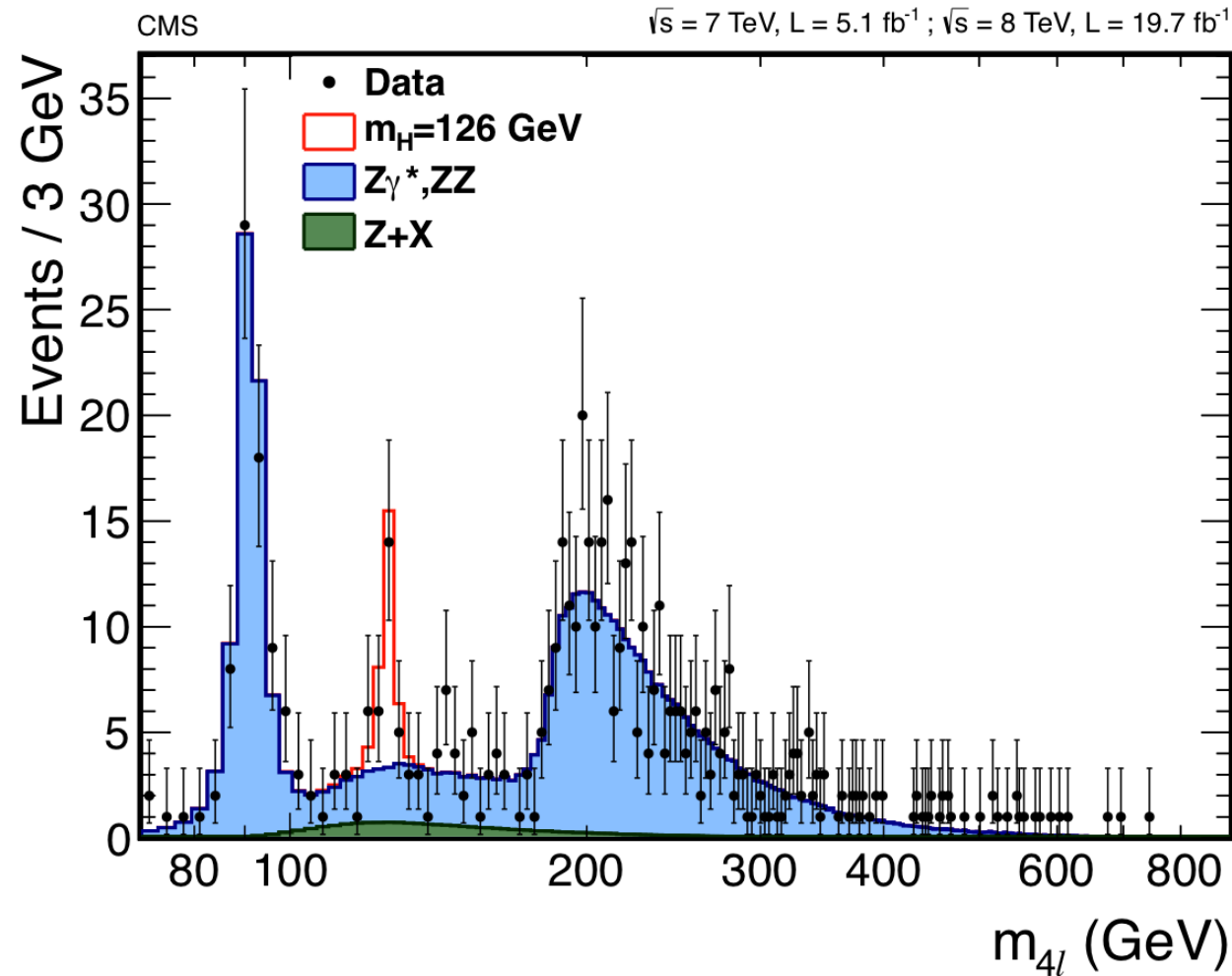
Final Result

```
Out[16]: <matplotlib.legend.Legend at 0x7fd0a7f308d0>
```



More on: <https://swan.web.cern.ch/content/apache-spark>

Final Result



The observed mass distribution of two Z bosons in four leptons events. The points are the data, while the blue histogram is the distribution expected from direct ZZ production in the Standard Model. The red line with a central value around 125 GeV is the Higgs boson signal.

Summary

Conclusions



CERN deals with an unprecedented amount of data:

- Several PBs at the collisions
- 12+ PBs of filtered data in a single month
- 250 PBs capacity at the Data Center



CERN utilizes a variety of frameworks from the Hadoop Ecosystem:

- Apache Spark
- Apache Hadoop/YARN
- Kafka
- Jupyter Notebooks



We have a number of critical and data intensive use cases:

- CMS Data Reduction Facility
- Next Accelerator Logging Service
- WLCG and Data Center Monitoring System
- Security Intrusion Detection



We now have fully functioning Analysis and Reduction examples tested over **1 PB** of CMS Open Data



Thank you

emotes@cern.ch

www.linkedin.com/in/evangelos-motesnitsalis