

---

# Interactive Data Analysis for End Users on HN Science Cloud

---

## Status Report on TOTEM Test

**Jakub T. Mościcki**  
CERN IT, Storage



Helix Nebula Science Cloud Pilot Phase  
29 November 2018, CERN, Geneva

# Contributors

---

- **TOTEM Experiment // AGH Krakow**

V. Avati, M. Blaszkiewicz, L. Grzanka, J. Kaspar  
M. Malawski, A. Mnich



- **EP, Software Development for Experiments**

D. Piparo, E. Tejedor, J. Cervantes

- **IT, Database Services**

L. Canali, P. Kothuri



- **IT, Storage**

E. Bocchi, D. Castro, J.T. Moscicki

# HN Deployment Test (2017)

- Automated deployment of “Science Box” on a single VM



- ✓ Streamlined installation – No configuration required
- ✓ Automated functional tests for validation

- ✓ IBM
- ✓ RHEA
- ✓ T-Systems

# HN Totem Test (2018) = “T3 in the cloud”

---

## ■ Goals

- ✓ Enable physicists to run data analysis interactively on bigger datasets, exploring state-of-the-art Big Data software
- ✓ Experiment with ability to move transparently between infrastructures
  - User experience on HN **identical** with the environment at CERN

## ■ Application

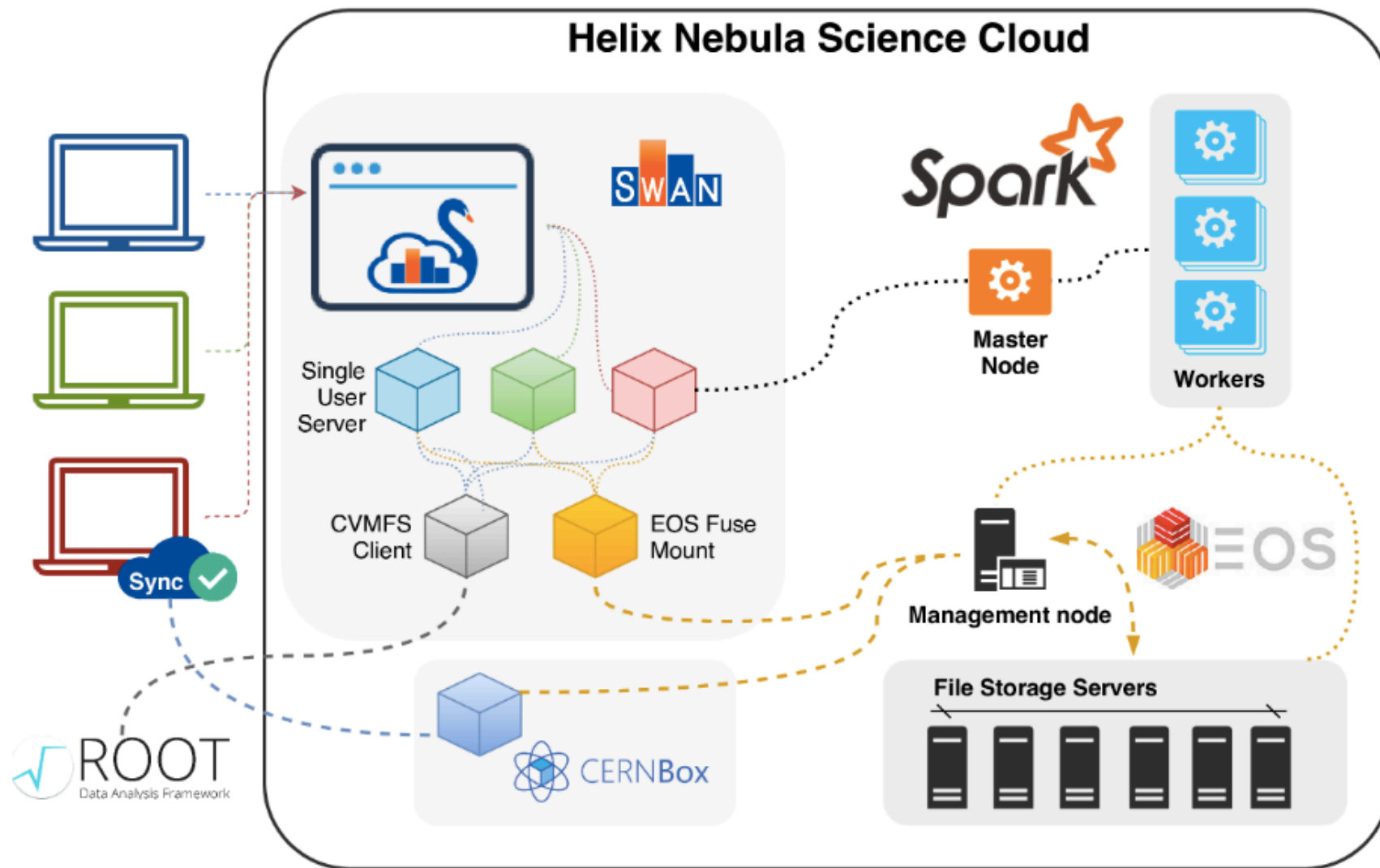
- ✓ TOTEM proton-proton elastic scattering analysis
- ✓ Use standard analysis framework: ROOT
  - Explore new RDataFrame interface

## ■ Data:

- ✓ Data transfer: import relevant datasets (CERN -> HN, ~5TBs)
- ✓ Using original data “as is”, no data transformation needed

## ■ Services:

- ✓ Infrastructure-agnostic service layer: ScienceBox
  - EOS, CERNBox, SWAN + SPARK



# Deployment + infrastructure feedback

---

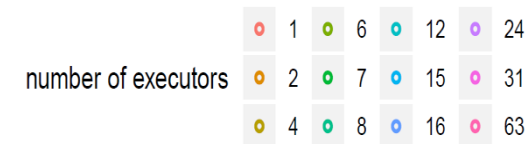
- ✓ All resources managed as sub-tenant: *TOTEM\_Test*
  - ScienceBox installed on plain VMs (Kubernetes)+Block Devices
  - Hadoop and Spark installed via yum on plain VMs (Cloudera Manager)
- ✓ Resources
  - Compute (Spark): 250 cores, 1.1TB RAM
  - Storage (EOS): 15TB used space
- ✓ Feedback
  - + Good baseline performance of raw resources (could not saturate in app)  
e.g. 350MB/s FIO sequential read and xrdcp
  - + Stability: good, comparable to experience with other infrastructures
  - + Good selection of VM flavours (e.g. 8GB RAM/core available)
  - + Responsive support
  - Some interventions non-transparent: Spectre fix
  - Missing DNS
  - Tenant separation (GPU resource created by another user)
  - Issues with VMs CC7 images, freeze&kernel issues with Spark executors

# What we achieved (1)

- Fully functional system & **validated physics results**
  - ✓ ROOT Map-Reduce analysis with Apache Spark driver
    - EOS dataset: 4.7 TB, 1148 ROOT files
  - ✓ Able to bring the execution time down to 7 minutes (240 cores)
    - >13 hours on a single core



Execution time:  
distill+analysis phases



## What we achieved (2)

---

- Flexibility
  - ✓ Users may run the very same analysis notebooks unmodified back in production environment at CERN
  - ✓ Getting the results back easily in the same place via CERNBox
  - ✓ Smooth experience with access to data in “virtualized” EOS instance
- Pushed several improvements for the in-house services, too
  - ✓ Optimization of RDataFrame (via new ROOT releases)
    - This model is also interesting for multicore and batch jobs at CERN
  - ✓ Spark at CERN to follow ScienceBox model => available as Kubernetes cluster with our OpenStack service



# What's next

---

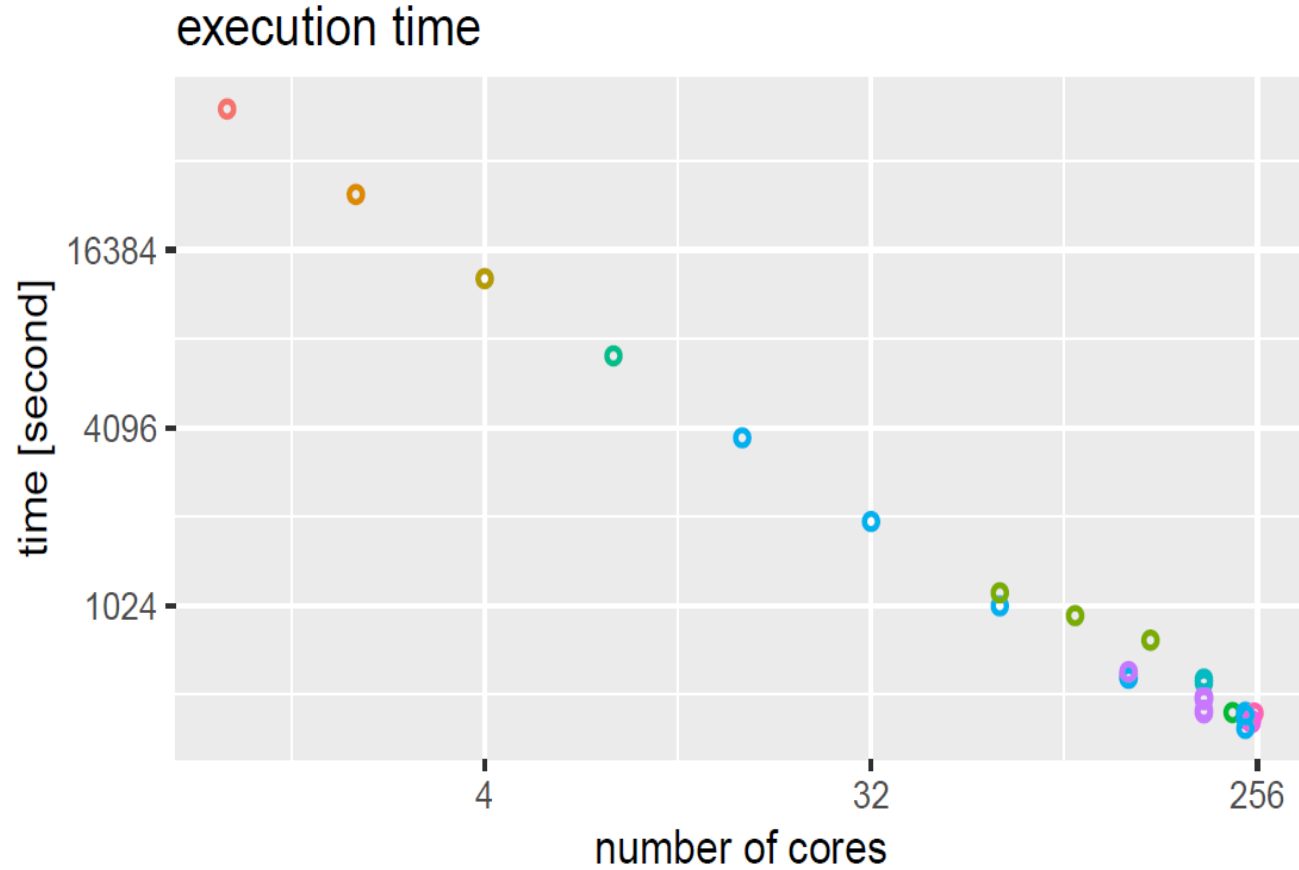
- Could not finish the tests because last 6 weeks lost
  - Abrupt (premature) termination of access mid-October
  - (if) resource extension granted then we will do scale tests x10 cores
- Looking forward possible future collaboration and testing in the next phases
- Thanks to all involved collaborators!

---

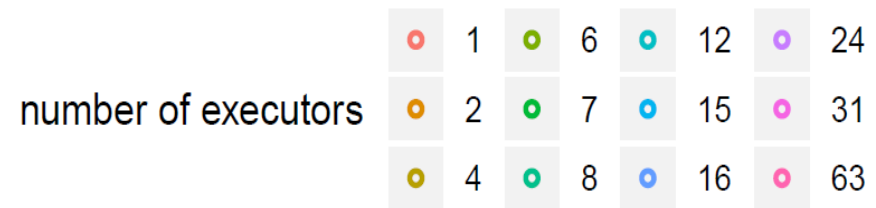
# Backup Slides

---

# Execution time of notebook encapsulating whole (distill + distribution) analysis



time	cores	speedup
13:39:00	1	1
07:00:49	2	1,95
03:38:16	4	3,75
...	...	...
00:09:41	128	84,58
00:06:33	240	125,03



## Original (C++)

```
// explicit event loop
for (int ev_idx = 0; ev_idx < ch_in-
>GetEntries();
    ev_idx += evIdxStep)
{
    ch_in->GetEntry(ev_idx);

    // check time
    if (anal.SkipTime(ev.timestamp))
        Continue;

    // diagonal cut
    bool allDiagonalRPs =
        (ev.h.L_2_N.v &&
ev.h.L_2_F.v && ev.h.R_2_N.v &&
ev.h.R_2_F.v);
    if (!allDiagonalRPs)
        continue;
    h_timestamp_dgn->Fill(ev.timestamp);
}
```

## RDataFrame (Py)

```
#Read all branches
rdf = RDF(treename, input_file)

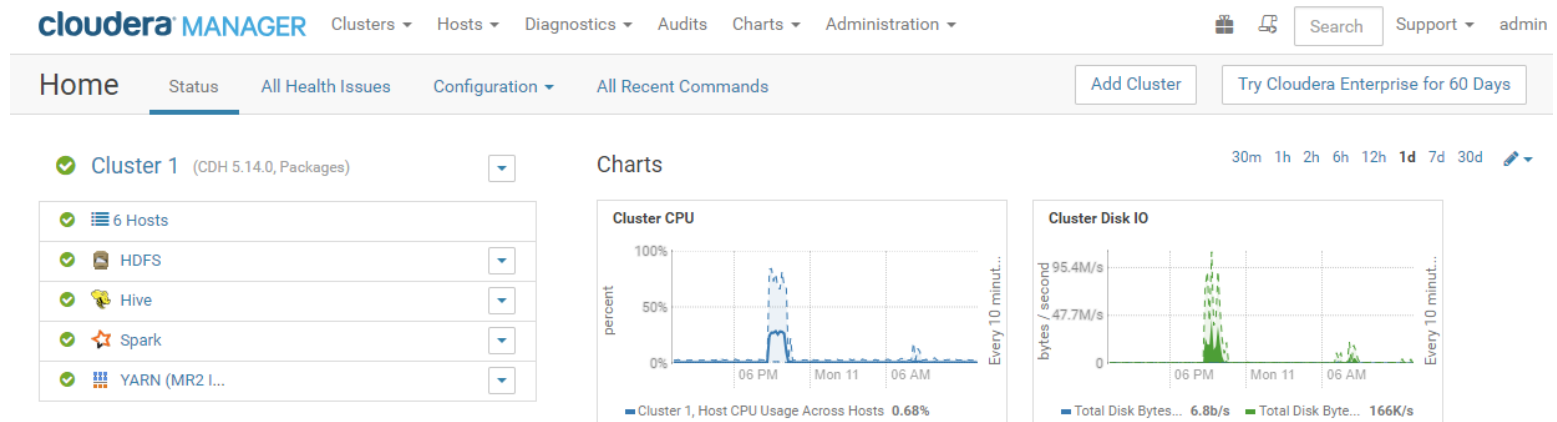
# check time
f1 = rdf.Filter("! SkipTime( timestamp )")

# diagonal cut
f2 = f1.Filter("v_L_2_F && v_L_2_N &&
v_R_2_F && v_R_2_N")
h_timestamp_dgn = f2.Histo1D(model,
"timestamp")
```

Code

# Current Tests: Spark Cluster

- ✓ Hadoop and Spark installed via yum on plain VMs
- ✓ Configuration via Cloudera Manager
- ✓ Validation with industry standard TPC-DS benchmark + user workloads



## Current Installation

- ✓ 6 nodes
- ✓ ECS Type: s2.xlarge.8
- ✓ Hadoop: CDH\_5.14
- ✓ Spark: 2.2.1
- ✓ JAVA: 1.8.0\_161-b12

# Deployment Status in HN Cloud

- Deployment **OK** on OTC
- All resources managed as sub-tenant “*TOTEM\_Test*”
- Small scale for now
  - ✓ 22 VMs, 41 Block Devices
  - ✓ 128 CPUs, 488GB Memory
  - ✓ 22 System Volumes (950GB)
  - ✓ 19 Data Volumes (~12TB)
- Functional tests successful
  - ✓ Storage, Sync, Analysis
  - ✓ SWAN to Spark connection
  - ✓ Import small dataset

The screenshot shows the Open Telekom Cloud (OTC) dashboard. The top navigation bar includes the OTC logo, the region 'eu-de(T...)', and links for 'Homepage', 'Service List', and 'Favorites'. A dropdown menu is open for the 'eu-de' region, showing the sub-tenant 'TOTEM\_Test'. The main content area is divided into 'Computing' and 'Storage' sections. The 'Computing' section shows 'Elastic Cloud Server (22)' and 'Cloud Server Backup Service (0)'. The 'Storage' section shows 'Elastic Volume Service (41)' and 'Volume Backup Service (0)'. A table below lists the ECS instances:

<input type="checkbox"/>	Name/ID	AZ	Status	Specifications/Image
<input type="checkbox"/>	<a href="#">eos-mgm</a> 02fdd392-0b2b-41f0-82...	eu-de-02	Running	16 vCPUs   32 GB Standard_CentOS_7_latest
<input type="checkbox"/>	<a href="#">swan</a> 05d69a83-5426-4349-a...	eu-de-02	Running	32 vCPUs   128 GB Standard_CentOS_7_latest
<input type="checkbox"/>	<a href="#">spark-0000</a> 1fe61b77-e8e8-4e72-97...	eu-de-02	Running	4 vCPUs   32 GB Standard_CentOS_7_latest

# Experience with OTC as a User

## ■ Incidents:

- ✓ May 7<sup>th</sup> Outage:
  - 3 containers lost due to VMs reboot
  - Manual intervention required
- ✓ May 14<sup>th</sup>: VM reported IO errors on system disk – VM was rebuilt
- ✓ Early June: OTC web console unavailable for short periods

Dear user of the Open Telekom Cloud,

We do apologize for the partial downtime of our Open Telekom Platform on 7<sup>th</sup> may and we would like to provide you the final error analysis of Open Telekom Cloud's partial outage.

**Date / time:** 2018-05-07, 21:25 - 22:24 CEST (19:25 - 20:24 UTC)

## ■ Requests:

- ✓ Streamline VM name <==> hostname <==> IP address resolution
  - E.g., Internal DNS to resolve VM name to its private IP address
  - Worked-around with entries in /etc/hosts + dnsmasq

# Next Steps

---

- Application layer optimization for TOTEM use-case
  - ✓ Performance tuning, caching of frequently-used software packages, ...
  - ✓ Focus on relevant metrics for users, e.g., total execution time
- Scale-out storage and computing resources
  - ✓ Leverage on scalable architecture design (EOS, Swan, Spark)
  - ✓ Storage:
    - Import larger dataset (10-50TB) → Additional file servers and block devices
    - Investigate caching layer for Spark for improved performance
  - ✓ Computing:
    - Experiment and identify best VM flavor for Spark computing nodes
    - Additional spark workers to crunch bigger datasets



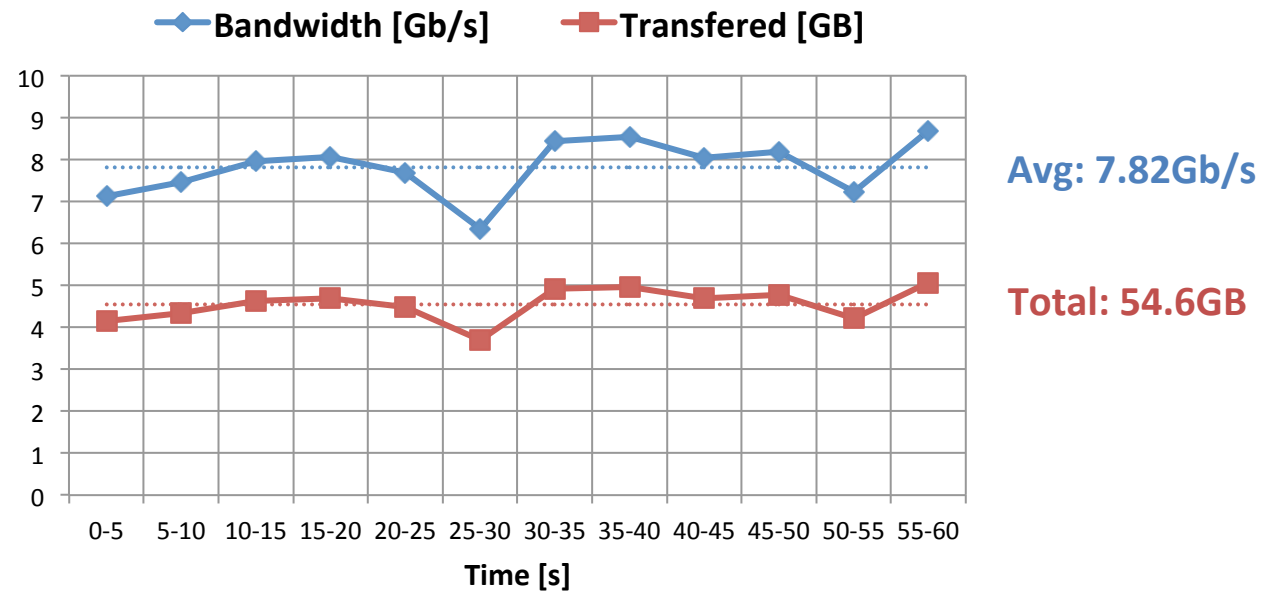
# Project Description

---

- This is a deployment test of the ScienceBox IT services (EOS, SWAN, CERNBox, and Spark) in a particular analysis scenario for TOTEM experiment (conducted jointly with SFT group and interested TOTEM collaborators).
- Phase 1 (April-September) is a feasibility study: (a) verification of the deployment of the services; (b) performance tuning and testing of a TOTEM data analysis example. If feasibility tests are successful then Phase 2 (September-November) is to use available resources opportunistically to understand the scaling limitations of the system and, if possible, to actually perform analysis for use-cases defined by the TOTEM collaborators.
- The amount of data currently foreseen is ~10-50TB. HN resource is treated as an extension of CERN resource. Any data stored in HN cloud will be automatically erased at the end of 2018.
- This project does not provide guarantee of service but will make opportunistic use of resources if feasible.

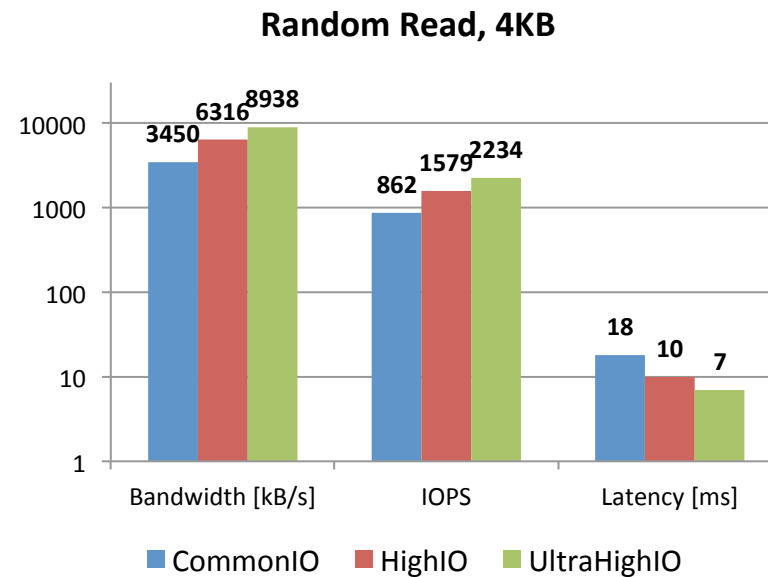
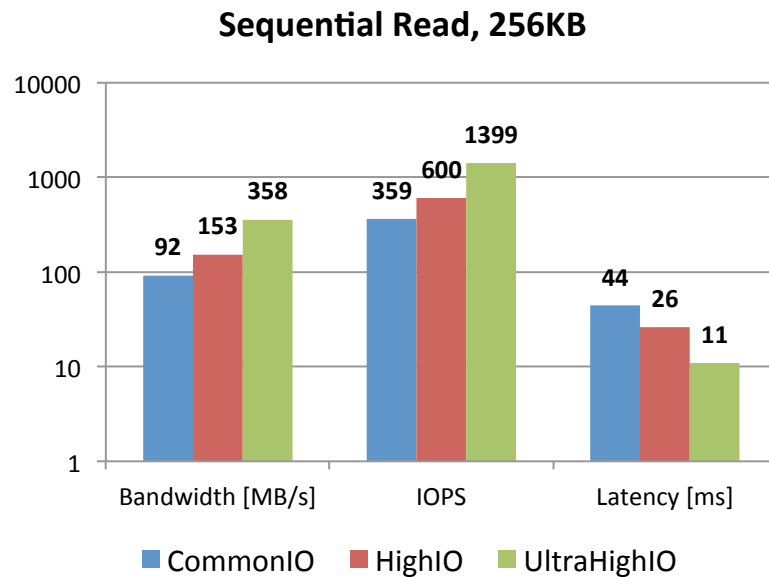
# Synthetic Benchmarks

- Network – iperf, ping
  - ✓ TCP connection, 128K buffer, 60s test (5s binning)
  - ✓ Average bandwidth: ~7.8 Gb/s (both directions)
  - ✓ Very low latency (~0.3ms) and jitter (sdev ~0.07ms)



# Synthetic Benchmarks (cont'd)

- Block Devices – fio
  - ✓ Three types of VDBs: Common-IO, High-IO, and UltraHigh-IO
  - ✓ No disk encryption, No disk sharing
  - ✓ Size form 10GB to 1TB (not related to performance)



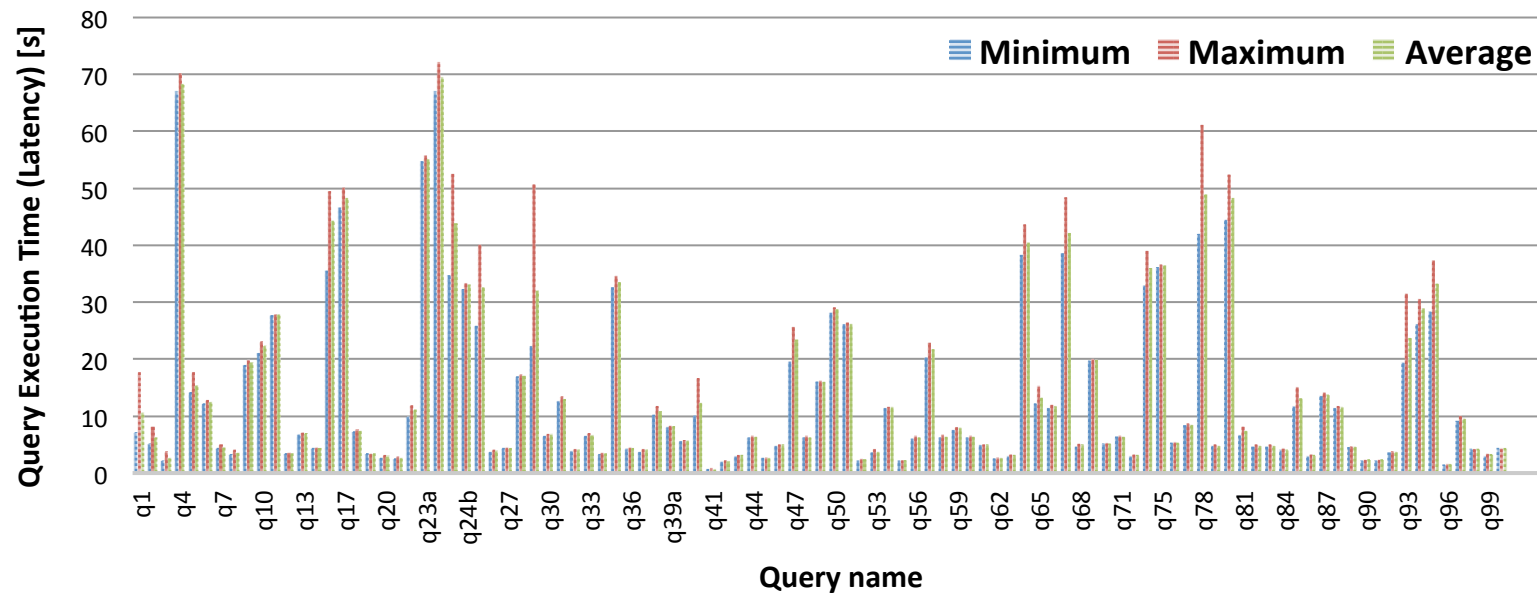
# Synthetic Benchmarks (cont'd)

---

- CPU and Memory – sysbench
  - ✓ CPU - Total events: 10'000
    - Avg. time per event: 1.11ms
    - 95<sup>th</sup> percentile: 1.12ms
  - ✓ Memory – Transfer (write) 10GB
    - IOPS: 1'614'325
    - Throughput: 1576.49 MB/s
- Two (main) flavor of VMs:
  - ✓ 4CPUs, 8GB memory – General purpose VM
  - ✓ 4CPUs, 32GB memory – Spark workers
  - ✓ Performance is consistent across VM sizes

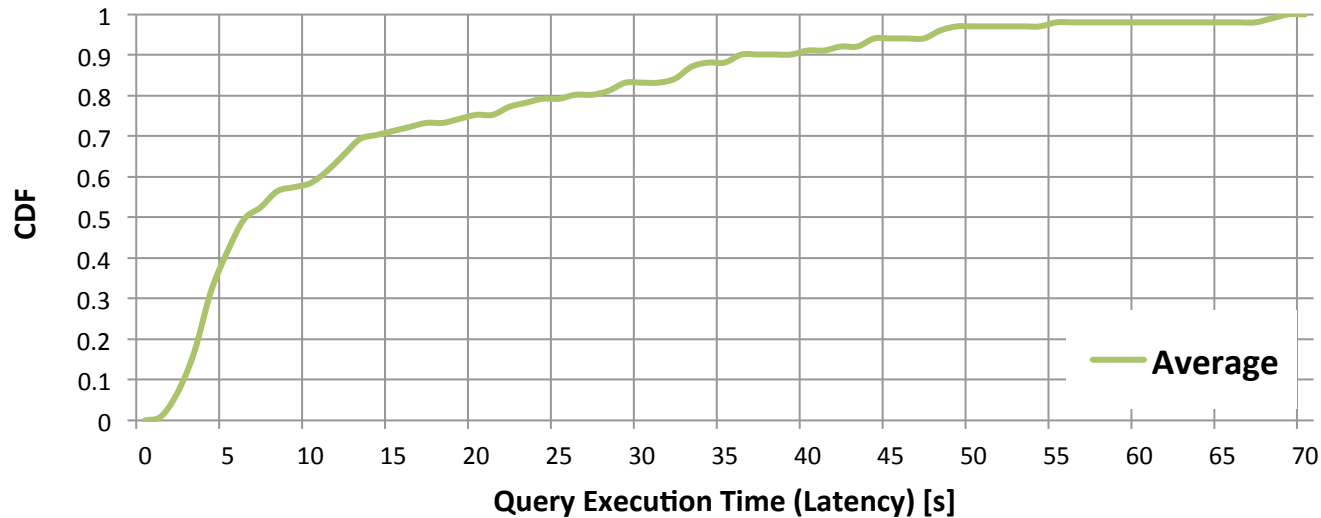
# Synthetic Benchmarks (cont'd)

- Spark Cluster – TPC-DS Benchmark
  - ✓ Small case test: Dataset size 20GB
  - ✓ 5 executors (2 cores, 7GB memory), Query Set v1.4, Spark 2.2.1



# Synthetic Benchmarks (cont'd)

- Spark Cluster – TPC-DS Benchmark
  - ✓ Small case test: Dataset size 20GB
  - ✓ 5 executors (2 cores, 7GB memory), Query Set v1.4, Spark 2.2.1



# Data Analysis with RDataFrame

```
df = RDataFrame('t', 'f.root')
hist = df.Filter('theta > 0') \
        .Histo1D('pt')
hist.Draw()
```

← - - - - Build data-frame object

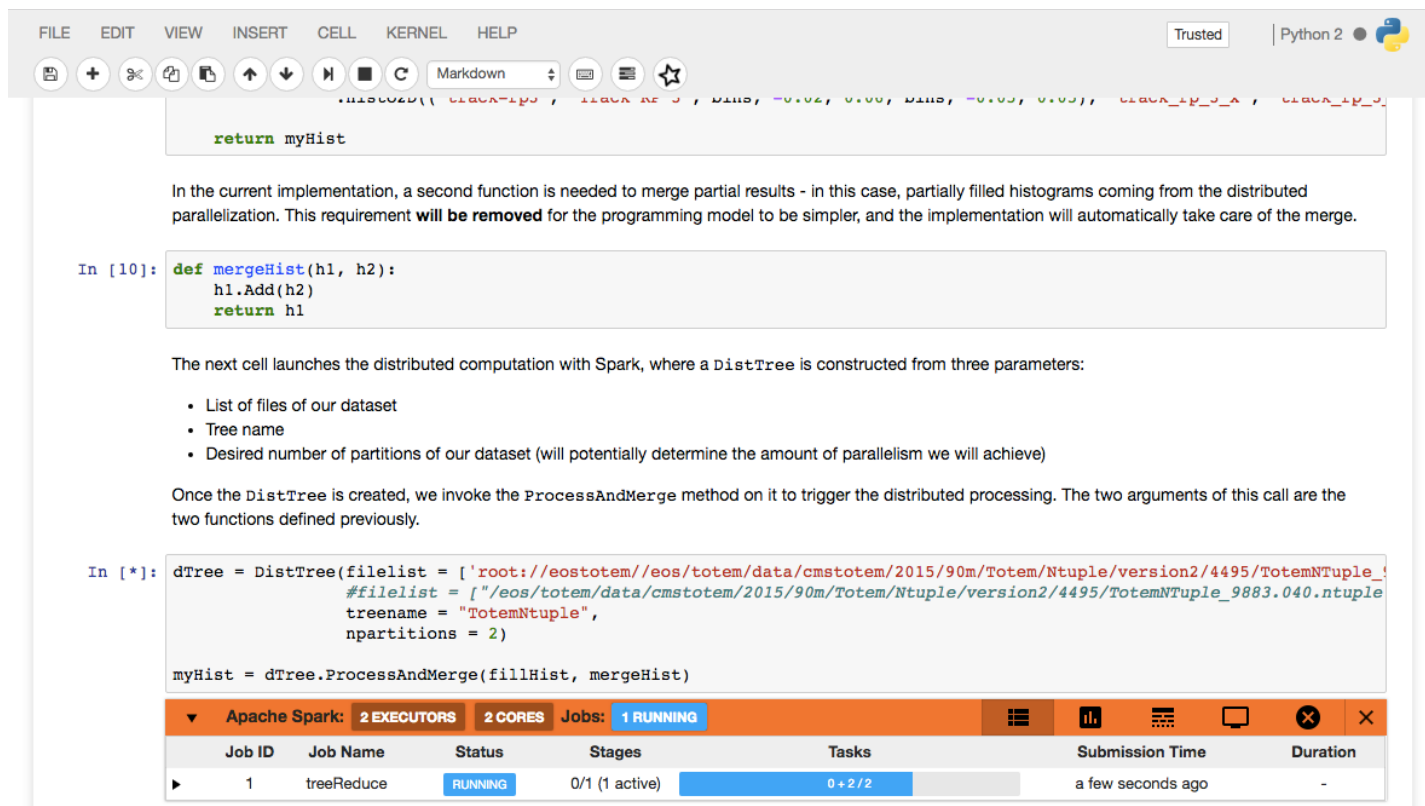
← - - - - Apply transformation

← - - - - Apply actions

- Columnar dataset: ROOT, other formats
- Declarative analysis: **what**, not how
  - ✓ Transformations: cuts, definition of new columns
  - ✓ Actions: return a result (e.g., histogram)
- Implicit parallelisation
  - ✓ Multi-threaded, distributed

# RDataFrame: Distributed Execution

- RDataFrame supports distributed execution on top of Spark
  - ✓ No changes in the app will be required to go distributed!



The screenshot shows a Jupyter Notebook interface with a menu bar (FILE, EDIT, VIEW, INSERT, CELL, KERNEL, HELP) and a toolbar. The notebook content includes:

```
return myHist
```

In the current implementation, a second function is needed to merge partial results - in this case, partially filled histograms coming from the distributed parallelization. This requirement **will be removed** for the programming model to be simpler, and the implementation will automatically take care of the merge.

```
In [10]: def mergeHist(h1, h2):
         h1.Add(h2)
         return h1
```

The next cell launches the distributed computation with Spark, where a `DistTree` is constructed from three parameters:

- List of files of our dataset
- Tree name
- Desired number of partitions of our dataset (will potentially determine the amount of parallelism we will achieve)

Once the `DistTree` is created, we invoke the `ProcessAndMerge` method on it to trigger the distributed processing. The two arguments of this call are the two functions defined previously.

```
In [*]: dTree = DistTree(filelist = ['root://eosstotem//eos/totem/data/cmstotem/2015/90m/Totem/Ntuple/version2/4495/TotemNtuple_
#filelist = ["/eos/totem/data/cmstotem/2015/90m/Totem/Ntuple/version2/4495/TotemNtuple_9883.040.ntuple
treename = "TotemNtuple",
npartitions = 2)

myHist = dTree.ProcessAndMerge(fillHist, mergeHist)
```

At the bottom, the Apache Spark job status is shown:

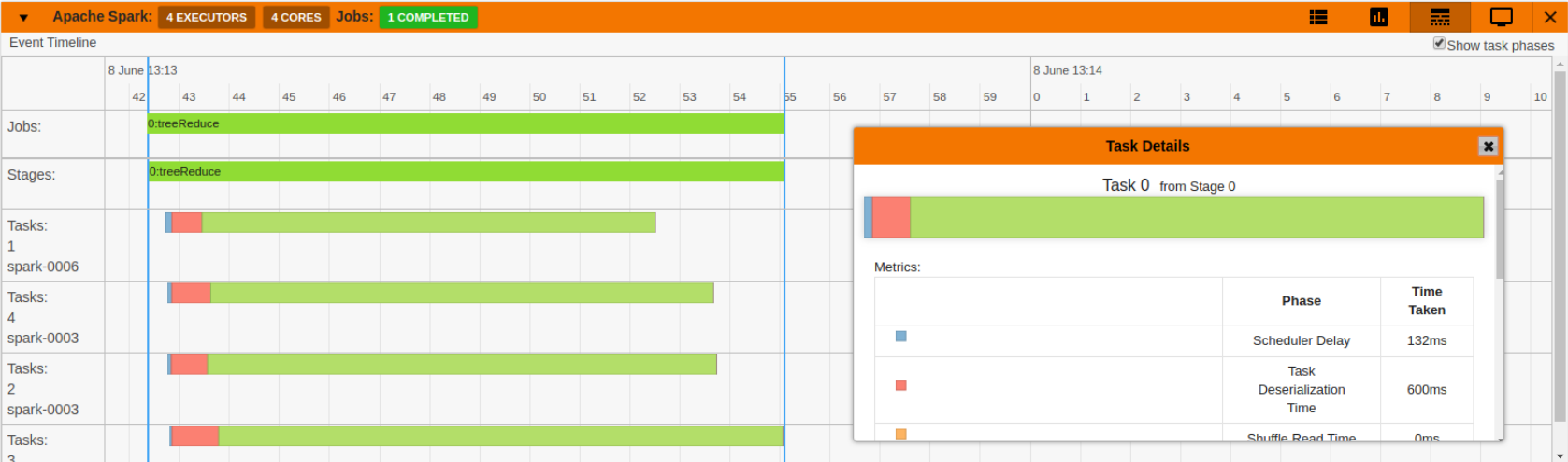
Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
1	treeReduce	RUNNING	0/1 (1 active)	0 + 2/2	a few seconds ago	-



# RDataFrame: Spark Monitoring

- Automatic monitoring when an RDataFrame Spark job is spawned
- Job progress, task distribution, resource utilization
  - ✓ Dataset split in ranges under the hood, one task per range

```
In [4]: dTree = DistTree(filelist = ['root://eostotem/eos/totem/data/cmstotem/2015/90m/Totem/Ntuple/version2/4495/TotemNTuple_9883.040.ntuple.root'],  
                        treename = "TotemNtuple",  
                        npartitions = 4)  
  
myHist = dTree.ProcessAndMerge(fillHist, mergeHist)
```



Apache Spark: 4 EXECUTORS 4 CORES Jobs: 1 COMPLETED

Event Timeline  Show task phases

8 June 13:13 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 8 June 13:14 0 1 2 3 4 5 6 7 8 9 10

Jobs: 0:treeReduce

Stages: 0:treeReduce

Tasks: 1 spark-0006

Tasks: 4 spark-0003

Tasks: 2 spark-0003

Tasks: 3

**Task Details**

Task 0 from Stage 0

Metrics:

Phase	Time Taken
Scheduler Delay	132ms
Task Deserialization Time	600ms
Shuffle Read Time	0ms