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
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
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
More Data Incoming!

HL-LHC will bring $\sim 30x$ 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

Source: HSF Community WhitePaper and CMS for HOW 2019
Complementing the existing approaches
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**

- **Hadoop**
- **Spark**

**Interactive Data Analysis**

- **Jupyter**
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

Hadoop/Spark

- HBase
- YARN Resource Manager
- HDFS Hadoop Distributed File System

On-Premise Bare metal Infrastructure

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

Spark on Kubernetes over Openstack

- External Storage (EOS, S3, HDFS)

- Spark-on-Kubernetes only compute
- Kubernetes Resource Manager
- Openstack Project 1 Resources

- Spark-on-Kubernetes only compute
- Kubernetes Resource Manager
- Openstack Project 2 Resources

Openstack Cloud Infrastructure

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**: 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](https://swan.web.cern.ch)
## My Projects

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>MODIFIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proj1</td>
<td></td>
<td>5 days ago</td>
</tr>
<tr>
<td>Proj2</td>
<td></td>
<td>15 days ago</td>
</tr>
<tr>
<td>Project</td>
<td></td>
<td>21 days ago</td>
</tr>
<tr>
<td>Project 1</td>
<td></td>
<td>2 months ago</td>
</tr>
<tr>
<td>Project 2</td>
<td></td>
<td>4 months ago</td>
</tr>
<tr>
<td>ProjTest</td>
<td></td>
<td>15 days ago</td>
</tr>
<tr>
<td>Spark</td>
<td></td>
<td>7 days ago</td>
</tr>
<tr>
<td>SWAN-Spark_NXCALS_Example</td>
<td></td>
<td>20 days ago</td>
</tr>
<tr>
<td>teste</td>
<td></td>
<td>19 days ago</td>
</tr>
</tbody>
</table>

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:

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

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

```plaintext
In [2]: TH1 h("myHisto","My Histo;X axis;Y axis",64, -4, 4)
```

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:

```plaintext
In [3]: TRandom3 rndmGenerator;
   for (auto i : ROOT::R Boo se{1000}){
      auto rdm = rndmGenerator.Gaus();
      h.Fill(rdm);
   }
```

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

```plaintext
In [4]: TCanvas c;
In [5]: h.Draw();
   c.Draw();
```

![My Histo](attachment:image.png)

- **Entries**: 1000
- **Mean**: 0.02880
- **Std Dev**: 1.038
Software Releases: CVMFS

- **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

- 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

Spark Cluster

Spark Master

Spark Executor

Task

Task

Task

Offload computations to pluggable resources
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!

```python
# Code here!
```

---

### Spark Monitor

**Event Timeline**

<table>
<thead>
<tr>
<th>Event Time</th>
<th>0h 0min 57s 000</th>
<th>0h 0min 58s 000</th>
<th>0h 0min 59s 000</th>
<th>0h 0min 59s 600</th>
<th>0h 1min 00s 000</th>
<th>0h 1min 00s 600</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 EXECUTORS</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
</tr>
<tr>
<td>1 CORES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 JOBS</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
<td>Collected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Job Name</th>
<th>Status</th>
<th>Stages</th>
<th>Tasks</th>
<th>Submission Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>reduce</td>
<td>Completed</td>
<td>3/2</td>
<td>4/4</td>
<td>5 minutes ago</td>
<td>3s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage Id</th>
<th>Stage Name</th>
<th>Status</th>
<th>Tasks</th>
<th>Submission Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>reduce</td>
<td>Completed</td>
<td>5/10</td>
<td>5 minutes ago</td>
<td>2s</td>
</tr>
<tr>
<td>4</td>
<td>coalesce</td>
<td>Completed</td>
<td>1/1</td>
<td>5 minutes ago</td>
<td>0s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage Id</th>
<th>Stage Name</th>
<th>Status</th>
<th>Tasks</th>
<th>Submission Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>foreach</td>
<td>Completed</td>
<td>1/1</td>
<td>5 minutes ago</td>
<td>1m:20s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage Id</th>
<th>Stage Name</th>
<th>Status</th>
<th>Tasks</th>
<th>Submission Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>coalesce</td>
<td>completed</td>
<td>1/10</td>
<td>Unknown</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>foreach</td>
<td>Completed</td>
<td>8/10</td>
<td>5 minutes ago</td>
<td>1m:20s</td>
</tr>
</tbody>
</table>
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: Everything in a Box!

- Self-contained, Docker-based package with:

  - single-box installation via **docker-compose**
  - no configuration required
  - download and run services in 15 minutes

  [https://github.com/cernbox/uboxed](https://github.com/cernbox/uboxed)

One-Click Demo Deployment

Production-oriented Deployment

  - container orchestration with **Kubernetes**
  - scale-out storage and computing
  - tolerant to node failure for high-availability

[https://github.com/cernbox/kuboxed](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

Datasource

- ROOT
- CSV
- Apache Arrow
- ATLAS’ xAOD
- SQLite
- RNTuple

Define

- p_x
- p_y
- p_z
- eta
- myvar

Range Filter

- histograms, profiles
- new ROOT files
- cut-flow reports
- data reductions (mean, sum,..)
- any user-defined operation
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

```python
D = RDataFrame(dataset)
F = D.Define(...)  # Define operations
    Define(...)  # Define operations
    Filter(...)  # Define operations
H1 = F.Histo1D(...)  # Histogram
H2 = F.Histo2D(...)  # Histogram
G = F.Graph(...)  # Graph
```
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)
Use Cases
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

1.5 TB/day at end of Run 2!
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
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)
Test setup

- **Data**
  - 4.7 TB TOTEM Dataset

- **Cluster**
  - 15 workers
  - 16 cores/worker

- **Requirements**
  - Data access (EOS)
  - Software (CVMFS)

**Time Reduction**
13 hours to 10 minutes
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
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!