ROOT and Spark for Physics Analysis

E. Tejedor, D. Piparo, P. Mató (EP-SFT)

in collaboration with IT-DB, IT-ST

CERN Openlab Technical Workshop

9/12/2016

Introduction

- **ROOT**: software toolkit that provides building blocks for data processing, analysis, visualisation and storage
  - Widely adopted in HEP
  - ~250 PB in ROOT format on LHC Grid

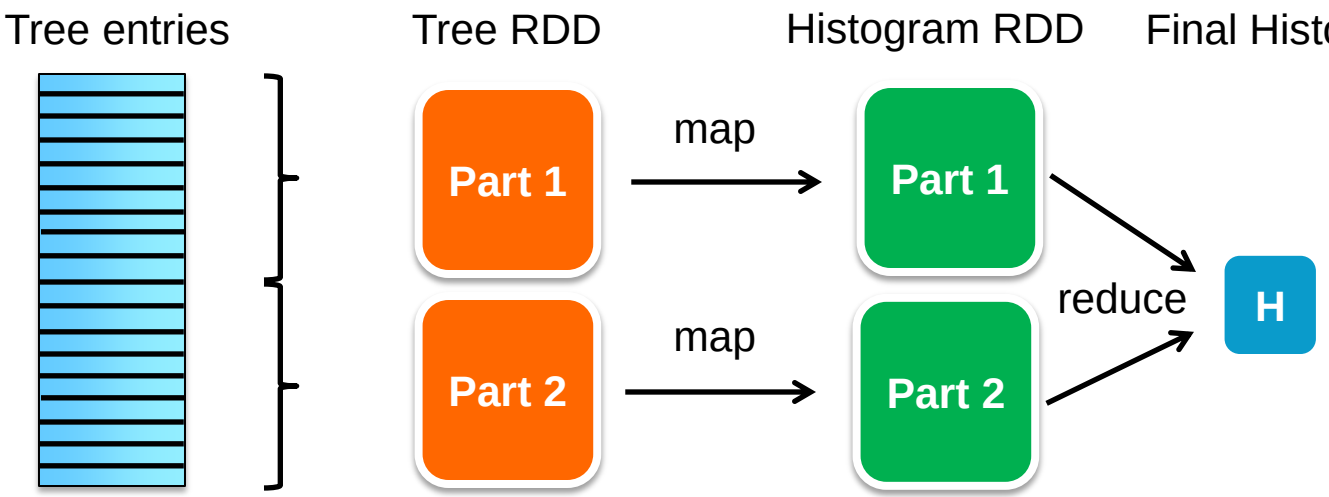
- **Apache Spark**: software framework for large-scale distributed data processing
  - Task scheduling
  - Fault tolerance
  - In-memory processing
  - Functional chains (MR)

Motivation

- Explore how Big Data technologies widely adopted in industry can help parallelising ROOT analysis
- Boost productivity of analysis setups
  - Provide a simple programming model
  - Reduce complexity of distributing computations
- Complement consolidated procedures for distributed computation in HEP
  - Batch / Grid jobs
  - PROOF (Parallel ROOT Facility)

Combining ROOT and Spark

- Spark is based on the following main concepts:
  - RDD: distributed collection of items
  - Transformations and actions applied on RDDs (map, reduce, filter, etc.)
- Model ROOT data as RDDs
  - Tree: collection of entries (1 or more files) - input of map
  - Histograms: output of map, input/output of reduce



Programming Model

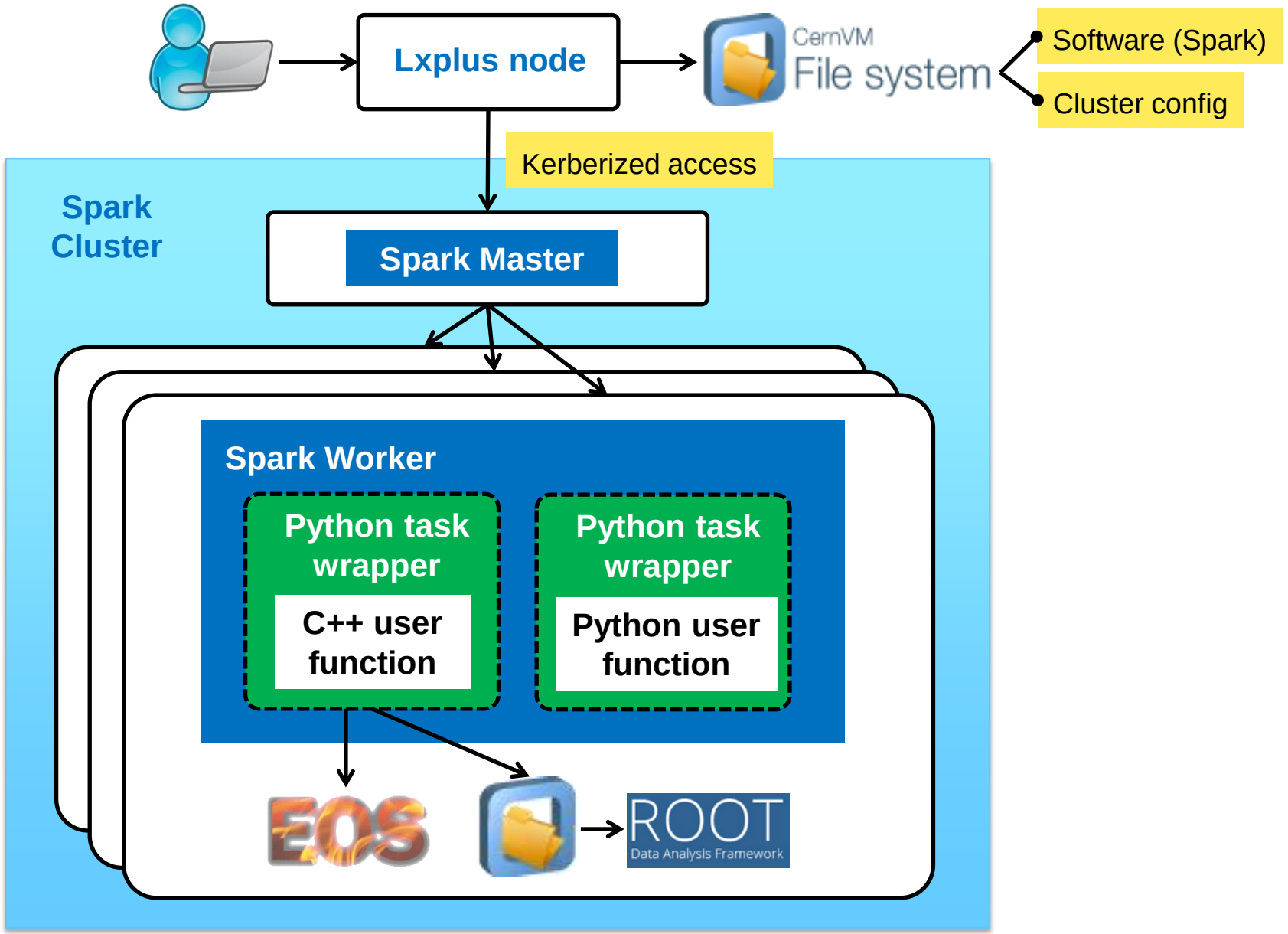
- Rely on PyROOT and PySpark
- Offer a simple, high-level API to do map-reduce on a tree
  - User provides map and reduce functions (Python or jitted C++)
  - Map function operates on a sub-range of entries (TTreeReader)
  - Partitions not tied to physical files

```
from ROOT import DistTree

dTree = DistTree(filelist = ["file0", "file1", …],
                 treename = "myTree",
                 npartitions = n)

histo = dTree.ProcessMerge(myMapFunction, myReduceFunction)
```

Implementation

- Objective: rely on existing SW technologies and IT infrastructure at CERN as much as possible
- Infrastructure
  - IT Spark clusters
  - IT container service
- Software: CVMFS
  - LCG release view
  - Cluster configuration
- Storage: EOS
  - Avoid problem of data ingestion
  - Avoid problem of splitting ROOT binary files
  - Authorized access to user's files

In a Picture


Lxplus node
CernVM
File system
Software (Spark)
Cluster config
Kerberized access
Spark Cluster
Spark Master
Spark Worker
Python task wrapper
C++ user function
Python task wrapper
Python user function
EOS
ROOT
Data Analysis Framework

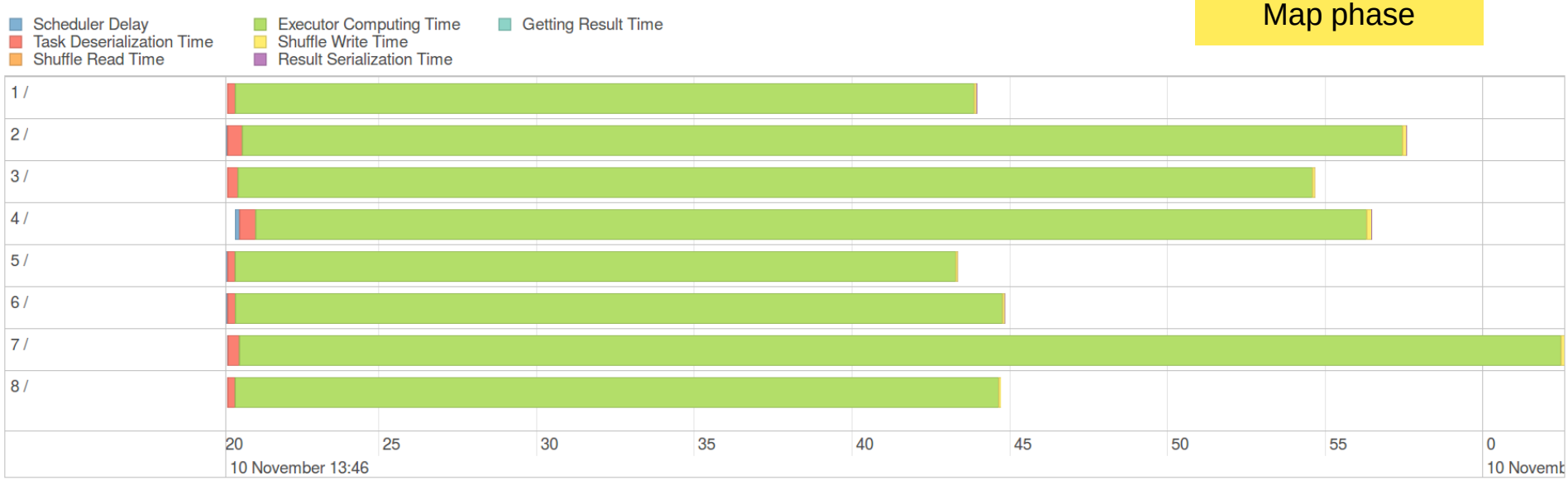
A Use Case

- CMS W mass measurement (thanks to Marc Dunser!)
- Total input dataset ~300 GB (on EOS)
  - Initial tests with just one file (~3 GB)
- Map phase: read tree range and fill a list of histograms
  - Mainly C++ code
  - Each mapper produces ~130 MB of histograms
- Reduce phase: merge lists of histograms
  - Different strategies: client-only, worker-only, tree reduce

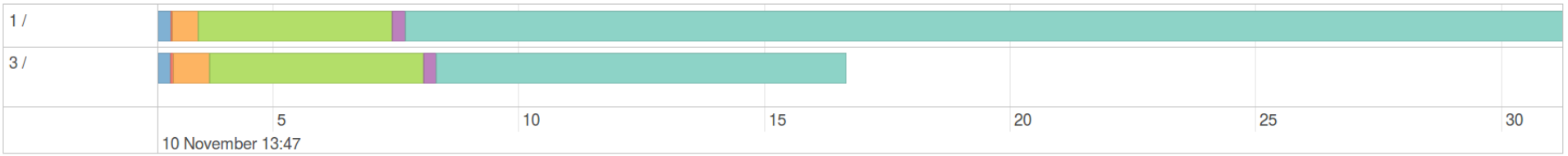
Analysis Tools

- **8 Spark workers**
- **8 partitions – 8 mapper tasks, 2 reducer tasks**

Map phase



Reduce phase

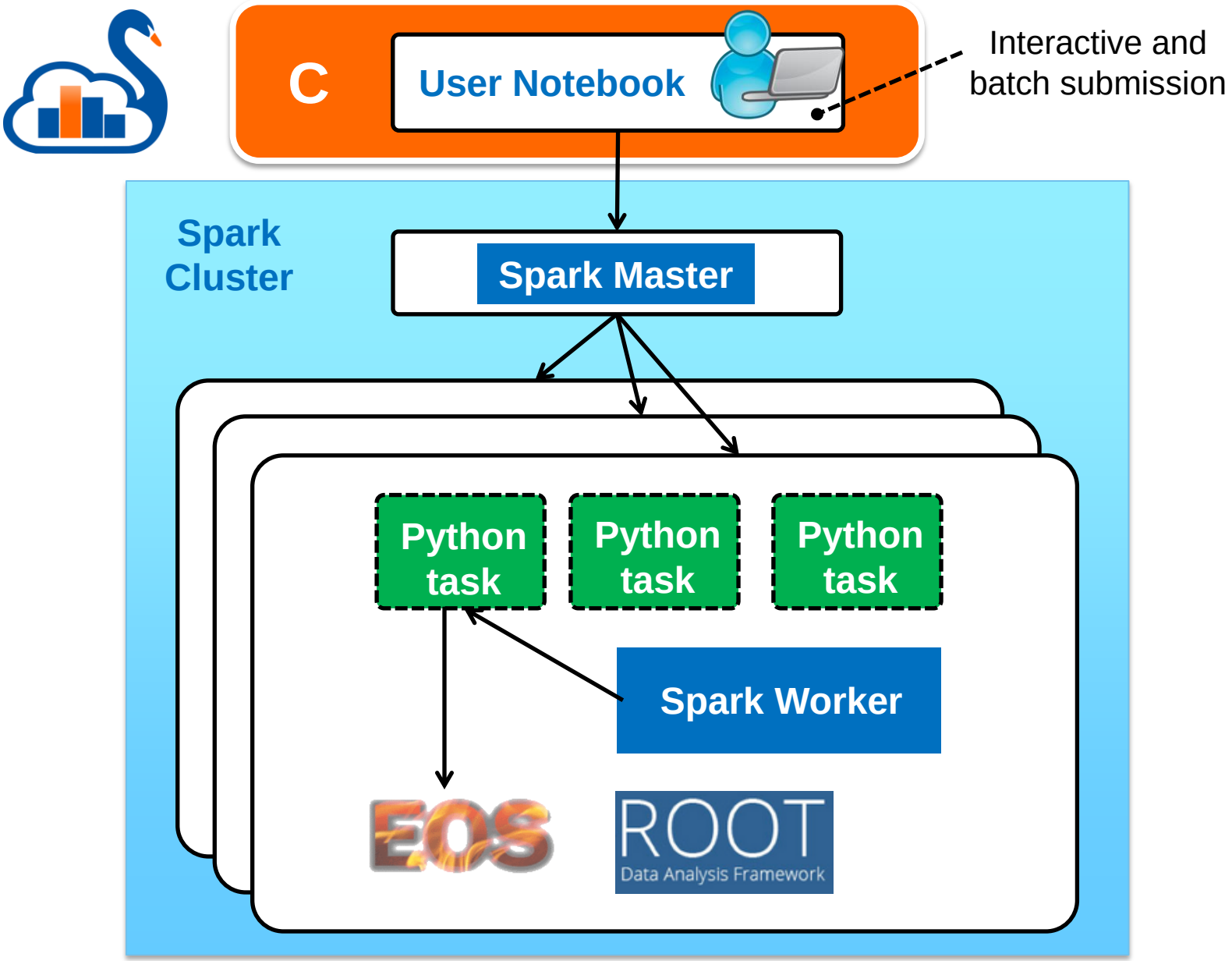


Summary

- ROOT and Spark integrated for distributed physics analysis
  - Available now, will be in ROOT soon
- MapReduce-like Python programming model
- Familiar technologies: EOS, CVMFS
- Leverage existing IT infrastructure
- Real use case: CMS analysis
- Looking for more use cases
  - Please contact us if you have one!

Backup

Offloading from SWAN



Challenges

Software

- Offer a simple, high-level API to do map-reduce on a tree
  - User provides map and reduce functions
  - Map function operates on a sub-range of entries (receives a TTreeReader)
  - Full example here
- Add a layer between Spark and user code
  - Build logical ranges of entries
  - Make partitions independent of number of files
- Allow to run both Python and C++ (jitted) functions as mappers and reducers
  - Wrapper tasks receive the function code and jit it in the workers

Infrastructure

- Use the same software environment on client (lxplus) and server (IT Spark clusters)
  - Completely based on CVMFS
  - LCG view + cluster configuration
- Allow authenticated access of data on EOS
  - Spark tasks operating on my ROOT files