ROOT Users Workshop

Monday, May 9, 2022 - Thursday, May 12, 2022

Virtual

Book of Abstracts

Contents

Second Session / 1

Fun ROOT Ana - functional ROOT analysis library

Author: Tomasz Bold[1]

[1] *AGH UST Krakow*

Corresponding Author: tomasz.bold@cern.ch

Big Data provoked a quest for better basic technologies, like programming languages,
that the analyzer has at its disposal when processing distributed datasets.
Somewhat surprisingly the well-known paradigm of functional programming,
when used in conjunction with object orientation turned out to match well those specific demands.
Several new languages like Scala or Kotlin appeared on that market and filled the niche.
They set an example that impacts the direction and design of libraries rooted
in traditionally non-functional programming languages like for instance Java streams or the c++20 Ranges library.
It is expected that next generations of students would learn that way
of thinking when it gets to data processing and expect it from every data processing system.

The characteristics of the code written in a functional style are its terseness and expressibility, especially for processing collections.
The data is typically processed in a lazy manner and thus resulting programs are quite CPU efficient.
In ROOT the RDataFrame leverages from this idea already for several years.

The FunRootAna is an attempt to provide a functional and modern collections library to ROOT users.
The design of it is inspired by Scala however the FunRootAna offers primarily the lazy approach.
A lean interface for collections processing demands a similarly typing-efficient
approach to handling the usual lifetime of the histograms, that is declaration, booking, and filling requiring instead only one line of code.

The promise made to the adopters of FunRootAna is that a line of code would be sufficient to generate a single histogram.
Would that not be fun?

To find out more: https://tboldagh.github.io/FunRootAna/

Summary:

First Session / 2

Welcome

Corresponding Author: sexton@fnal.gov

First Session / 3

ROOT State of the Union

Corresponding Author: axel@fnal.gov

First Session / 4

Cling updates

Corresponding Author: vvasilev@cern.ch

First Session / 5

ROOT as a Python module

Corresponding Author: etejedor@cern.ch

Summary:

Third Session / 6

Roofit in 2022

Corresponding Author: jonas.rembser@cern.ch

Third Session / 7

Sofie

Corresponding Author: lorenzo.moneta@cern.ch

First Session / 8

RNtuple (and object stores)

Corresponding Author: jblomer@cern.ch

First Session / 9

distRDF

Corresponding Author: vincenzo.eduardo.padulano@cern.ch

First Session / 10

RDF Vary

Corresponding Author: enrico.guiraud@cern.ch

First Session / 11

HighLO finance time series

Corresponding Author: philippe.mmd@gmail.com

First Session / 12

INFN distRDF

Corresponding Authors: diego.ciangottini@cern.ch, dciangot@fnal.gov

Second Session / 13

Conda packaging

Corresponding Author: christopher.burr@cern.ch

Second Session / 14

Homebrew, Pip install

Corresponding Author: hschrein@cern.ch

Second Session / 15

Snap

Corresponding Author: eloquism@gmail.com

Summary:

Second Session / 16

TBA

Second Session / 17

REST-for-Physics

Corresponding Author: luis.antonio.obis@gmail.com

Second Session / 18

New User Report - Aman Goel

Corresponding Author: aman.goel185@gmail.com

Second Session / 19

New User Report - Guillermo Fidalgo

Corresponding Author: guillermo.fidalgo@upr.edu

Second Session / 20

New User Report

Second Session / 21

Report from Josh BenDavid

Corresponding Author: jbendavi@mit.edu

Second Session / 22

Report from Ferdy Mercury

Corresponding Author: fernando.hueso@uv.es

Second Session / 23

Report from Nick Manganelli

Corresponding Author: nmangane@fnal.gov

First Session / 24

Bamboo

Corresponding Author: sebastien.wertz@cern.ch

Third Session / 25

RBrowser

Corresponding Author: s.linev@gsi.de

Third Session / 26

REve

Corresponding Author: mtadel@ucsd.edu

Third Session / 27

ATLAS Analysis Models

Corresponding Author: hrussell@cern.ch

Third Session / 28

High performance analysis with RDataFrame: Scaling and Interoperability

Author: Josh Bendavid[1]

[1] *MIT*

Corresponding Author: jbendavi@mit.edu

The unprecedented volume of data and Monte Carlo simulations at the HL-LHC will pose increasing challenges for data analysis both in terms of computing resource requirements as well as "time to insight". Precision measurements with present LHC data already face many of these challenges today. I will discuss performance scaling and optimization of RDataFrame for complex physics analyses, including interoperability with Eigen, Boost Histograms, and the python ecosystem to enable this.

Summary:

Third Session / 29

DUNE Analysis Models

Corresponding Author: trj@fnal.gov

Third Session / 30

CMS Analysis Models

Author: Piergiulio Lenzi[None]

Corresponding Author: piergiulio.lenzi@cern.ch

Summary:

Third Session / 31

ALICE Analysis Models

Corresponding Author: anton.alkin@cern.ch

Third Session / 32

LHCb Analysis Models

Summary:

Third Session / 33

Mu2e Analysis Models

Corresponding Author: srsoleti@fnal.gov

Third Session / 34

COMPASS Analysis Models

Author: Jan Matoušek[None]

Corresponding Author: jan.matousek@cern.ch

Summary:

First Session / 35

Julia, C++ and ROOT

Corresponding Author: philippe.gras@cern.ch

Third Session / 36

RooFit Developments in Parallelisation and User Interface

Authors: Patrick Bos[1]; Wouter Verkerke[2]; Zef Wolffs[2]

[1] *eScience Center*

[2] *NIKHEF*

Corresponding Authors: zefwolffs@gmail.com, p.bos@esciencecenter.nl, verkerke@nikhef.nl

With many researchers relying on ROOT's fitting framework RooFit for the fits of their statistical models to data there is a need for this framework to be both fast and user friendly. Recently, significant efforts were made to improve in both of these directions. For increased fitting speed a novel gradient-based parallelisation framework was built to supersede the already existing likelihood-based parallelisation framework. The main advantage of the newly implemented parallelisation strategy is its improved load balancing due to the use of a random work stealing algorithm for scheduling.

In order to hide the added complexity of this and potentially future developments from the end user a new interface to the fitting framework was also built. A factory design pattern was implemented for the building of likelihoods to allow the user to be agnostic towards the used fitting strategy on the back-end, while providing them with all the features formerly available in RooFit. This is the start of a larger effort to separate the statistical and computational aspects of RooFit.

This talk will introduce the aforementioned developments, cover recent work on benchmarking the parallelisation framework on use cases from the field of particle physics, and provide an overview of future benchmarking and improvement efforts.

Summary:

Second Session / **37**

Creative Misuse of ROOT

Author: Jack Armitage[1]

[1] *Intelligent Instruments Lab*

Corresponding Author: jack@lhi.is

From Steinway and Helmholtz, to Max Mathews and Bell Labs, to Google Magenta, scientists and musicians have long been natural collaborators, with innovations and inventions passing in both directions. Over the years, a select few creative coders have been quietly reappropriating ROOT technologies, particularly for musical applications. In one such example, Cling has been used as the basis for a C++ based live coding synthesiser [1]. In another example, Cling has been installed on a BeagleBoard to bring live coding to the Bela interactive audio platform [2]. More recently, embedded digital musical instrument designers are experimenting with machine learning for gesture recognition and audio synthesis with the SOFIE library [3]. What would happen if the ROOT community were to embrace and encourage creative misuse of Cling, SOFIE and other powerful CERN technologies? Could musical innovation and invention inspire and benefit scientific practice at CERN? This short talk proposes to review the examples and explore the questions above, with the aim of promoting discourse between ROOT technologists, CERN researchers, and creative technologists.

[1] tiny spectral synthesizer with livecoding support https://github.com/nwoeanhinnogaehr/tinyspec-cling
[2] Using the Cling C++ Interpreter on the Bela Platform
https://gist.github.com/jarmitage/6e411ae8746c04d6ecbee1cbc1ebdcd4
[3] Test running ONNX models on Bela via ROOT@CERN's SOFIE inference code generator
https://gist.github.com/jarmitage/0ac53dfecee8ed03e9f235d3e14ec9a2

Summary:

Third Session / 38

RooWorkspace serialization: RooFit goes JSON

Author: Carsten Burgard[None]

Corresponding Author: cburgard@cern.ch

The statistical model is an extremely information-dense and comprehensive representation of a physics measurement, which is why efforts to preserve and publish it have been pursued for years. One of the main challenges is to find a format in which to store a statistical model that is useful for people external to the experimental community and not bound to a specific software (such as RooFit or RooFit-derived customized pdf classes), while at the same time being general enough to cover all use-cases. With the advent of pyhf and its JSON specification for statistical models, the community has seen a surge in publishing full models, which has been well-received by theorists. While the reduced use case of HistFactory lends itself to a compact and declarative representation, the extension of such a format for a general, open-world scenario is challenging, but of critical importance. The RooJSONFactoryWSTool is a new addition to RooFit that allows the export and import of generic workspaces to JSON or YAML. The path chosen to address the challenge is to provide a fully extensible interface to register new types of functions and pdfs, allowing to import and export not only classes shipped with ROOT, but any implementation residing in a workspace. The accompanying documentation sets out a draft for a "High Energy Physics Statistics Serialization Standard" (HS3), which will hopefully be adopted and extended by the community and increase traction for the publishing of statistical models. The implementation is still in experimental stage, but available with ROOT 6.26/00.

Summary:

Second Session / 39

Awkward Arrays to RDataFrame and back

Authors: Ianna Osborne[None]; Jim Pivarski[1]

[1] *Fermilab*

Corresponding Authors: ianna.osborne@cern.ch, pivarski@fnal.gov

Awkward Arrays and RDataFrame provide two very different ways of performing calculations at scale. By adding the ability to zero-copy convert between them, users get the best of both. It gives users a better flexibility in mixing different packages and languages in their analysis.

In Awkward Array version 2, the ak.to_rdataframe function presents a view of an Awkward Array as an RDataFrame source. This view is generated on demand and the data is not copied. The column readers are generated based on the run-time type of the views. The readers are passed to a generated source derived from ROOT::RDF::RDataSource.

The ak.from_rdataframe function converts the selected columns as native Awkward Arrays.

We discuss the details of the implementation exploiting JIT techniques. We present examples of analysis of data stored in Awkward Arrays via a high-level interface of an RDataFrame.

We show a few examples of the column definition, applying user-defined filters written in C++, and plotting or extracting the columnar data as Awkward Arrays.

We discuss current limitations and future plans.

Summary:

Third Session / 40

RooFit in 2022

Authors: Jonas Rembser[None]; Lorenzo Moneta[1]

[1] *CERN*

Corresponding Authors: lorenzo.moneta@cern.ch, jonas.rembser@cern.ch

News on RooFit

Summary:

First Session / 41

RDataFrame: status and plans

Author: Enrico Guiraud[1]

[1] *EP-SFT, CERN*

Corresponding Author: enrico.guiraud@cern.ch

ROOT's RDataFrame enables the development of high-performance, highly parallel HEP analyses in C++ and Python – without requiring expert knowledge of multi-thread parallelization or ROOT I/O.

This contribution presents several features recently introduced in RDataFrame that improve the ergonomics of common HEP use cases and provides a glimpse of what is to come in the future. Topics will include interoperability of C++ and Python code, scaling up execution from a laptop to large computing clusters with minimal code changes, machine learning inference and user-friendly handling of systematic variations.

Summary:

The latest news on RDataFrame, ROOT's modern and high-level analysis interface for C++ and Python.

First Session / 42

RNTuple - The Next Generation TTree

Authors: Jakob Blomer[1]; Javier Lopez-Gomez[1]

[1] *CERN*

Corresponding Authors: jblomer@cern.ch, javier.lopez.gomez@cern.ch

The volume of generated data in upcoming HEP experiments, e.g. at the HL-LHC, is expected to increase by at least one order of magnitude. In order to retain the ability to analyse the influx of data, full exploitation of modern storage hardware and systems, such as low-latency high-bandwidth NVMe devices and distributed object stores, becomes critical.

To this end, the ROOT RNTuple I/O subsystem has been designed to address performance bottlenecks and shortcomings of ROOT's TTree. RNTuple provides a backwards-incompatible redesign of the TTree binary format and access API that evolves the ROOT event data I/O for the challenges of the upcoming decades. It focuses on a compact data format, on performance engineering for modern storage hardware, and on robust interfaces that are easy to use correctly.

In this contribution, we provide a brief introduction to RNTuple, giving some insights on its support for the Intel DAOS object store including the latest collected performance numbers.

Summary:

In this contribution, we provide a brief introduction to RNTuple, giving some insights on its support for the Intel DAOS object store including the latest collected performance numbers.

Third Session / 43

RBrowser

Author: Serguei Linev[1]

[1] *GSI Darmstadt*

Corresponding Author: s.linev@gsi.de

RBrowser is new web-based widget in ROOT combines objects and files browsing with different kinds of objects displays.

Separate library is used to provide browsing capability for file system, ROOT collections, content of TFiles and TTrees. With a little effort, custom classes and collections can be implemented.

RBrowser user interface implemented with OpenUI5. Different kind of web-based widgets are supported: TCanvas and RCanvas with JSROOT graphics, geometry viewer, text editors. Basic TTree::Draw is also possible.

Since ROOT 6.26 RBrowser is used as replacement for "classical" TBrowser.

Summary:

First Session / 44

Distributed RDataFrame: supported backends, latest improvements and future plans

Authors: Vincenzo Eduardo Padulano[1]; Enric Tejedor Saavedra[1]; Enrico Guiraud[2]

[1] *CERN*
[2] *EP-SFT, CERN*

Corresponding Authors: enrico.guiraud@cern.ch, vincenzo.eduardo.padulano@cern.ch, etejedor@cern.ch

The declarative programming model offered by RDataFrame provides high-level abstractions for users to operate on their datasets in a much more ergonomic fashion compared to previous imperative interfaces. This tool has already seen a lot of usage in real-world analyses and production environments, showing optimal results. RDataFrame has always been oriented towards parallelisation, with native support for multi-threading execution on a single machine which doesn't need changes in user code. The parallelisation capabilities have more recently been extended with a Python layer that is capable of steering and executing the RDataFrame computation graph over a set of distributed nodes, also in this case requiring minimal code changes. This new extension features a modular design, such that it can support multiple backends in order to exploit the vast ecosystem of distributed computing frameworks with Python bindings. This talk presents the design behing distributed RDataFrame, discussing the currently available execution backends, the latest improvements and how the tool will continuously evolve in the near future.

Summary:

Second Session / 45

Help yourself with the proper tools to tame ROOT

Author: Fernando Hueso González[1]

[1] *IFIC (CSIC - UV)*

Corresponding Author: fernando.hueso@uv.es

ROOT is an incredibly rich and biodiverse set of tools that lets you run the most sophisticated analyses on very large physics datasets with a great speed and only a few lines of code. However, there are a myriad of classes and options you may deploy, and is hard to know them all or choose the best suited, and the documentation is sometimes cryptic or scattered across manual, reference guide and forum.
Likewise, there are hundreds of tricks to remember, as well as caveats and hidden bugs. Among students, ROOT is notoriously famous for making easy the most complex tasks, and for turning pretty difficult the most simple ones, like zooming a graph. This not only affects the efficiency of daily or even experienced ROOT users, but also scares potential newcomers in favour of other software (numpy, MATLAB, matplotlib, …). As a consequence, the size and growth of the ROOT community is dampened, and the potentially enriching contributions of these users-to-be are lost.
To counterbalance this brain drain, as well as to tame the powerful beast that ROOT represents for daily users, I have focused my effort on improving its user-friendliness by fixing outdated documentation, improving the help messages and usability, as well as developing and sharing workflows for effective programming and debugging.
In my talk, I will review what external tools I deem essential for interacting with ROOT in a way that makes your experience (newbie or expert developers) more productive and comfortable. Finally, I will outline the high-performance data acquisition system I have programmed using ROOT for prompt gamma-ray experiments in the field of medical physics and proton therapy.

Summary:

Second Session / 46

The Adventures of a ROOT Novice

Author: Aman Goel[1]

[1] *University of Delhi*

Corresponding Author: aman.goel185@gmail.com

The ROOT data-analysis framework is a powerful and feature-rich library that is extremely popular for scientific analyses in high-energy physics. The enormousness of ROOT with all its bells and whistles can make it seem intimidating to dive into as a beginner.

Having worked with uproot (a library for reading and writing ROOT files in pure Python and NumPy), in this talk, I will discuss my experience with ROOT after my recent first contact with it at the Software Carpentry Workshop. I will discuss my experience with ROOT as I learnt about its various features such as PyROOT, C++ interpretation, histogramming, interactive plots, and RDataFrame among many others.

Summary:

Second Session / **47**

Learning experience from Software Carpentries

Author: Guillermo Fidalgo-Rodríguez[1]

[1] *University of Puerto Rico at Mayagüez*

Corresponding Author: guillermo.fidalgo@upr.edu

I am a graduate student from the University of Puerto Rico at Mayagüez.

ROOT has been the primary tool when analyzing and plotting HEP data. Recently there have been many improvements to ROOT interfaces like the adaptation to the python language. As a young member of the HEP community I have mainly used python-ROOT interfaces, like Uproot, Awkward Arrays, Pandas, and Coffea for physics analysis and other projects. However, as I advance my experience I would like to go a bit deeper into the usage of ROOT and related tools. In this short report I discuss my experience from ROOT training on the latest iteration of the Software Carpentry workshop.

Summary:

Second Session / **48**

12345: Lessons Learned building an Analysis Framework around RDataFrame and CMS NanoAOD

Author: Nicholas Manganelli[1]

[1] *University of California Riverside (US)*

Corresponding Author: nmangane@cern.ch

With the advent of the Compact Muon Solenoid Experiment's smallest centrally-maintained data format, NanoAOD, a description of proton-proton collisions for general physics analysis is reduced to just 2-4kB per event. ROOT's RDataFrame, an efficient engine for processing HEP data using declarative syntax, easy multithreading, and flexible interfaces from C++ and python, is well-suited as a core building block for a new framework. Lessons learned, for both physicists looking to use RDF and developers looking for feedback, will be presented based on the experience of a lone graduate student building a framework "from scratch" to almost-public results.

Summary:

Train The Trainer / 49

Welcome and introduction

Corresponding Author: etejedor@cern.ch

Train The Trainer / 50

Update on ROOT documentation

First Session / 51

Julia, C++, and ROOT

Author: Philippe Gras[1]

[1] *Université Paris-Saclay - CEA/Irfu - France*

Corresponding Author: philippe.gras@cern.ch

Julia is a programming language, with a growing interest in the HEP community, that allies easy of programming, similar to Python, with computing performance, similar to C and C++. Reuse of legacy libraries is essential for the adaption of a language in High energy physics. I will present in this talk how Julia can be interfaced with legacy C++ libraries and in particular with the ROOT framework. In particular, we will see how the generation of the glue code providing the Julia programming interface to a C++ library can be automatized.

Summary:

Train The Trainer / 52

ROOT training material

Corresponding Author: vincenzo.eduardo.padulano@cern.ch

Train The Trainer / 53

Training tools

Corresponding Author: etejedor@cern.ch

Train The Trainer / 54

Introduction to ROOT - what to teach?

Corresponding Author: lorenzo.moneta@cern.ch

Train The Trainer / 55

Teaching RDataFrame

Corresponding Author: enrico.guiraud@cern.ch

Summary:

Train The Trainer / 56

Teaching RooFit

Corresponding Author: jonas.rembser@cern.ch

Summary:

Train The Trainer / 57

External trainers - Bill Seligman

Author: William Seligman[1]

Co-author: William Seligman

[1] *Columbia University Nevis Labs*

Corresponding Authors: seligman@fnal.gov, seligman@nevis.columbia.edu

Summary:

Train The Trainer / 58

External trainers - Oliver Lantwin

Author: Oliver Lantwin[1]

[1] *LAPP*

Corresponding Author: oliver.lantwin@lapp.in2p3.fr

Summary:

Train The Trainer / 59

External trainers - Antonio Iuliano

Author: Antonio Iuliano[1]

[1] *Università di Napoli and INFN*

Corresponding Author: antonio.iuliano@cern.ch

Summary:

Train The Trainer / 60

External trainers - David Brunner

Author: David Brunner[1]

[1] *DESY*

Corresponding Author: david.brunner@desy.de

Summary:

Train The Trainer / 61

Final discussion & wrap-up

Open discussion:
- Topics from the audience
- How to engage more people in teaching ROOT?
- How to share material?
- How to stay in contact?

Summary:

Third Session / 62

RooFit Developments in Parallelisation and User Interface

Corresponding Author: zefwolffs@gmail.com

Third Session / 63

RooWorkspace serialization: RooFit goes JSON

Corresponding Author: cburgard@cern.ch

Second Session / 64

ROOT Packaged as a Snap

Author: James Carroll[None]

Corresponding Author: eloquism@gmail.com

The Linux platform suffers from API/ABI incompatibility across distributions and libraries. The ROOT team and community provide several package formats to compensate, however the new Linux/ROOT user will still have to grapple with understanding of how to use these formats and their related quirks.

This presentation discusses the Snap package of ROOT, aimed primarily at the new ROOT/Linux user who simply wants to be able to access the core ROOT functionality with as little setup as possible. The Snap format presents advantages and disadvantages alike, but offers a very attractive solution using an immutable, isolated, and automatically updating container package.

Examples of ideal use cases include the classroom environment, where users would be provided with a reliable and consistent ROOT interpreter, PyROOT bindings, JupyROOT support, and a pre-selected bundle of data-science packages. A reproducible and easily deployable environment helps eliminate overhead in debugging environment errors; allowing the user to simply proceed with their work with minimal fuss.

Summary:

Second Session / 65

REST-for-Physics

Author: Luis Antonio Obis Aparicio[1]

[1] *University of Zaragoza*

Corresponding Author: luis.antonio.obis.aparicio@cern.ch

REST-for-Physics is an event oriented analysis framework based on ROOT.
It has a modular structure with multiple libraries and packages allowing from processing of raw detector data to generation and analysis of simulated data (using Geant4).
REST provides a unified event format and extensive metadata tracking capabilities, which makes it suitable for complex working environments.

Summary:

REST-for-Physics is an event oriented analysis framework based on ROOT

Third Session / 66

REve

Authors: Alja Mrak Tadel[1]; Matevz Tadel[1]; Serguei Linev[2]

[1] *UCSD*

[2] *GSI Darmstadt*

Corresponding Authors: amraktadel@ucsd.edu, mtadel@ucsd.edu, s.linev@gsi.de

REve is a migration and extension of the 15-year old TEve for ROOT-7 and web-based graphics. Cling allows for experiment-independent implementation of physics data-collection interfaces and table-views. R&D work has been ongoing for almost 4 years and REve is now approaching the stage where it can be used to implement certain visualization applications. Main design ideas and development drivers, current status, and future plans will be presented.

Summary:

First Session / 67

Fast and readable analysis with Bamboo

Authors: Pieter David[None]; Sébastien Wertz[1]

[1] *UCLouvain*

Corresponding Author: sebastien.wertz@cern.ch

ROOT's RDataFrame is a powerful tool to implement an analysis in a declarative way, reducing boilerplate and abstracting away technical details, but can be considered a low-level interface around which a complete analysis framework needs to be built.
Bamboo implements a high-level interface to RDataFrame that allows users to express the analysis logic in a concise functional style in terms of objects (collections of variables) and collections of objects.
This effectively represents an analysis description language embedded in python, with performance approaching that of dedicated native code thanks to the underlying RDataFrame and cling JIT compilation.
In addition, Bamboo provides many features needed for a complete analysis, such as the bookkeeping of input samples, handling of systematic uncertainties, submission and monitoring of jobs on a batch cluster, etc., making it a complete turnkey analysis framework.
This talk presents the motivation behind Bamboo, an overview of its most useful features, and the plans for developments in the near future.

Summary:

First Session / 68

Using and Adapting ROOT for High-frequency Financial Market Data

Authors: Philippe Debie[1]; Axel Naumann[2]; Marjolein Verhulst[1]; Joost Pennings[3]; Jonas Rembser[2]; Serdar Demirel[1]; Lorenzo Moneta[2]

[1] *Wageningen University*

[2] *CERN*

[3] *Wageningen University, Maastricht University*

Corresponding Authors: axel@fnal.gov, philippe.debie@wur.nl, lorenzo.moneta@cern.ch, jonas.rembser@cern.ch

In particle physics, ROOT is the dominant framework for data analysis, resulting in strong collaborations, easy data sharing and reproducible research. On contrary, research on the high-frequency structure of financial markets lacks such an universally used framework, and often uses custom implementations to analyse large data sets. This results in a relative small number of high-frequency publications.

Our research is part of an unique collaboration between researchers in particle physics and in finance. We use ROOT to store, process and analyse high-frequency market data. This presentation will show the similarities of data in particle physics and finance, and argues the suitability of adapting ROOT for finance data. We present a framework built on top of ROOT to work with time series data. In addition, we show a prototype of an implementation integrating this framework into RDataframe, thus reducing the entry level of using ROOT for finance and leveraging the multi-threading performance of RDataframe.

Summary:

First Session / 69

ROOT as a Python Module

Author: Enric Tejedor Saavedra[1]

[1] *CERN*

Corresponding Author: etejedor@cern.ch

ROOT can be used as a Python module thanks to PyROOT, the Python-C++ bindings provided by ROOT. With ROOT v6.22, a new PyROOT was released; it is now designed on top of the cppyy library, which provides support for modern C++.

This talk will mainly focus on the customizations or "pythonizations" for ROOT classes that have been recently implemented in the new PyROOT. This includes interoperability with NumPy for RDataFrame, the ability to just-in-time compile Python callables with Numba and use them in RDataFrame operations, and a context manager for TFile.

Furthermore, PyROOT now offers a @pythonization decorator that can be used to register pythonizations for C++ user classes, which are lazily executed only if the class is actually used from the user code.

Summary:

First Session / 70

CMS experience at INFN with distRDF over HTCondor

Authors: Daniele Spiga[1]; Diego Ciangottini[2]; Tommaso Tedeschi[None]

[1] *INFN Perugia*

[2] *INFN, Perugia (IT)*

Corresponding Author: diego.ciangottini@pg.infn.it

In the context of R&D activities for the evolution of the analysis computing model for the CMS experiment, one of the focus is the capability to leverage an (quasi-)interactive and declarative approach to enhance both the user experience and the analysis throughput (meant as the end to end result delivery time). Another key point is how to make use of both grid resources and opportunistic ones in a coherent and efficient way.

In this talk we will show how distributed RDataFrame has been tested importing a CMS analysis to RDF framework and executing it on a prototype analysis-facility infrastructure at INFN. The presented scenario allows the user to login in a central JupyterHUB instance (or directy via ssh) and to

schedule the RDataFrame payload on a remote Dask cluster instantiated via HTCondor. It will also be shown how RDataFrame allowed the transition from legacy code with a minimal effort.

Summary:

Third Session / 71

Close Out

Corresponding Author: axel@fnal.gov

Second Session / 73

Belle 2 Analysis Models

Corresponding Author: frank.meier@duke.edu