ROOT Users' Workshop

Sunday, 9 September 2018 - Thursday, 13 September 2018

Academy of Sciences and Arts (Akademija nauka i umjetnosti Bosne i Hercegovine)

Book of Abstracts

Contents

Opening / **1**

ROOT State of the Union and vision for 2020

Author: Axel Naumann[1]

[1] *CERN*

Corresponding Author: axel.naumann@cern.ch

Since the last ROOT Users' Workshop in 2015, ROOT has seen many new developments, new team members and contributors, production releases and infrastructure changes. This presentation will summarize where ROOT is, from measurements (lines of code, commits per second, merge request and forum activity) to summary-level news on areas of development and infrastructure (github, Discourse).

ROOT is at the heart of physics analysis - and we want to stay there! This presentation will explain what is driving the development, and where it's being driven to. It will show ROOT's role in today's data analysis context - e.g. where and how it is unique - and what it will take to keep ROOT alive and rocking for the next two decades. For instance, the team will need to focus development on few key ingredients. But what are those? See and discuss the ROOT team's answer in this presentation!

Parallelism, Heterogeneity and Distributed Data Processing / **6**

RooFit parallelization efforts

Authors: Patrick Bos[1]; Wouter Verkerke[2]; Vince Croft[3]; Inti Pelupessy[4]; Carsten Daniel Burgard[2]

[1] *Netherlands eScience Center / Nikhef National institute for subatomic physics (NL)*

[2] *Nikhef National institute for subatomic physics (NL)*

[3] *New York University (US)*

[4] *Netherlands eScience Center*

Corresponding Authors: p.bos@esciencecenter.nl, verkerke@nikhef.nl, carsten.burgard@cern.ch, vincent.croft@cern.ch, i.pelupessy@esciencecenter.nl

We present an update on our recent efforts to further parallelize RooFit. We have performed extensive benchmarks and identified at least three bottlenecks that will benefit from parallelization. To tackle these and possible future bottlenecks, we designed a parallelization layer that allows us to parallelize existing classes with minimal effort, but with high performance and retaining as much of the existing class's interface as possible. The high-level parallelization model is a task-stealing approach. The implementation is currently based on the bi-directional memory mapped pipe (BidirMMapPipe), but could in the future be replaced by other modes of communication between processes.

Input and Output of Scientific Datasets / **7**

RForest: Evolution of ROOT Tree I/O

Author: Jakob Blomer[1]

[1] *CERN*

Corresponding Author: jakob.blomer@cern.ch

The ROOT TTree data format encodes hundreds of petabytes of HEP events. Its columnar layout drives rapid analyses, as only those parts (columns) that are really used in a given analysis need to be read from storage. Its unique feature is the seamless C++ integration, which allows users to directly store their event classes without explicitly defining data schemas.

In this contribution, we present an evolution for future ROOT event I/O. Along with the ROOT 7 interface modernization, we aim for robust, where possible compile-time safe C++ interfaces to read and write event data. On the implementation side, we show a prototype that combines the best of ROOT's current TTrees with recent advances in columnar data formats. A core ingredient is a strong separation of the high-level logical data layout (C++ classes) from the low-level physical data layout (arrays of simple types). We show how a a well-defined low-level data format speeds up serialization and deserialization and facilitates fast vectorized and bulk operations. This lets ROOT I/O run optimally on the upcoming ultra-fast NVRAM storage devices, as well as file-less storage systems such as object stores.

Scientific Visualisation, GUIs and Documentation / **8**

A ROOT Gallery

Author: Jakob Blomer[1]

[1] *CERN*

Corresponding Author: jakob.blomer@cern.ch

We all learn by example. Great examples are not easy to create. I'd love to see a user driven ROOT gallery. A place where users can publish examples of beautiful plots they created in ROOT or tricky issues they solved in ROOT. The gallery should be searchable and user's should be able to comment and improve examples.

Multivariate Analysis / **9**

TMVA in the Future: Adapting to the Modern Machine-Learning Landscape

Authors: Stefan Wunsch[1]; Lorenzo Moneta[2]

[1] *KIT - Karlsruhe Institute of Technology (DE)*

[2] *CERN*

Corresponding Authors: stefan.wunsch@cern.ch, lorenzo.moneta@cern.ch

TMVA has been a pioneering effort which set a milestone for machine-learning (ML) in high-energy physics (HEP) more than ten years ago and remains in use in numerous analyses of LHC experiments.
On the other hand, the ML landscape explosively evolved during these years and - as industry stepped in - ML became suddenly one of the most active fields in science. This talk discusses how TMVA can make the difference for ML in HEP in the future by combining the developments in the ML community and recent developments in the ROOT framework.

Platforms, Infrastructure and Builds / **10**

Evolution of ROOT's CMake Build System

Authors: Guilherme Amadio[1]; Axel Naumann[1]

[1] *CERN*

Corresponding Authors: amadio@cern.ch, axel.naumann@cern.ch

ROOT 6.14/00 is the first release in which CMake is used exclusively for the build system, after deprecation and removal of the old configure/make scripts. However, the CMake build system was added in 2011, before CMake 3.0 and "modern CMake" became established. This presentation will summarize how ROOT's build system is evolving to use modern CMake constructs to become more modular and to better support users' projects that depend on ROOT.

ROOT and the Python Ecosystem / 11

PyROOT: Redesign and New Features

Authors: Enric Tejedor Saavedra[1]; Stefan Wunsch[2]

[1] *CERN*

[2] *KIT - Karlsruhe Institute of Technology (DE)*

Corresponding Authors: enric.tejedor.saavedra@cern.ch, stefan.wunsch@cern.ch

With Python already established as a key player in scientific computing, the interest for the language in HEP has been growing over the last years. In order to address that demand, the ROOT Python bindings (PyROOT) are being modernised and extended. A new experimental PyROOT has been created by leveraging Cppyy, which provides the basic bleeding edge functionality for generating automatic Python bindings for C++ libraries. On top of Cppyy, PyROOT adds the so-called ROOT pythonizations: code that provides simpler and more pythonic ways of accessing ROOT C++ classes from Python. This talk will present the most recently added pythonizations together with the current status and future of the experimental PyROOT.

End User Perspective / 12

Overtaking PROOF: The Future of Distributed Analysis in ROOT

Authors: Enric Tejedor Saavedra[1]; Javier Cervantes Villanueva[1]

[1] *CERN*

Widespread distributed processing of big datasets has been around for more than a decade now thanks to Hadoop, but only recently higher-level abstractions have been proposed for programmers to easily operate on those datasets, e.g. Spark. ROOT has joined that trend with its RDataFrame tool for declarative analysis, which currently supports local multi-threaded parallelisation. However, RDataFrame's programming model is general enough to accommodate multiple implementations or backends: users could write their code once and execute it as is locally or distributedly, just by selecting the corresponding backend. Moreover, the programmer should be provided with ergonomic interfaces, possibly web-based, which allow to dynamically plug in new resources, as well as to write, execute, monitor and debug distributed applications in the most intuitive way possible.

Parallelism, Heterogeneity and Distributed Data Processing / 13

Support for SIMD Vectorization in ROOT

Author: Guilherme Amadio[1]

Co-authors: Xavier Valls Pla [2]; Lorenzo Moneta [1]; Danilo Piparo [1]

[1] *CERN*

[2] *University Jaume I (ES)*

Corresponding Authors: danilo.piparo@cern.ch, lorenzo.moneta@cern.ch, xavier.valls.pla@cern.ch, amadio@cern.ch

The next big upgrade of the LHC will increase data volume by an order of magnitude. This is a significant challenge that will require most HEP software to adapt to be able to exploit all forms of parallelism. Portable and efficient SIMD vectorization has been a particularly difficult challenge due to incompatible programming APIs from different libraries, and rapid evolution of hardware. This presentation will discuss common vectorization techniques, VecCore, the library used by GeantV and ROOT for portable SIMD vectorization, and techniques for building software with support for multiple architectures.

Parallelism, Heterogeneity and Distributed Data Processing / 15

Support Future Data Analysis with a Parallelised ROOT

Author: Danilo Piparo[1]

[1] *CERN*

Corresponding Author: danilo.piparo@cern.ch

Future accelerators and detectors pose to HEP scientific software a series of challenges, among which the efficient analysis of the data collected by the experiments. In the past few years ROOT became to a large extent parallel but our endeavour is not complete.
This presentation is dedicated to the characterisation of the parallelisation effort which took place up to now and to the lessons learned thanks to this effort. The units of the ROOT framework are discussed where task and data parallelism have been introduced, with runtime and scaling measurements. We will give an overview of concurrent operations in ROOT, for instance in the areas of I/O (reading and writing of data), fitting / minimization, and data analysis.

Extraordinary results have been achieved with the parallelisation of ROOT and more will come in the short term. On the other hand, assuming a flat founding for HEP computing and a reasonable evolution of hardware, in order to implement the research programme of HL-LHC, our software needs to be 10x faster than today. This includes data analysis (and therewith IO) and sophisticated procedures such as fits.
What does ROOT need to provide to support data analysis and processing in such a harsh environment?
In this talk we propose our vision about the ROOT's core components which will need most attention. Notable examples are the backend for IO, the interpreter and its ability to support heterogeneous platforms and ROOT's type system.

Platforms, Infrastructure and Builds / 17

ROOT Package Manager

Authors: Brian Paul Bockelman[1]; Oksana Shadura[1]; Vasil Georgiev Vasilev[2]

[1] *University of Nebraska Lincoln (US)*

[2] *Princeton University (US)*

Corresponding Authors: brian.bockelman@cern.ch, oksana.shadura@cern.ch, vasil.georgiev.vasilev@cern.ch

This presentation discusses our work on ROOT package manager and build features for it. The part of talk will conclude the integration of package manager into ROOT runtime. Taking in account ongoing work on C++ modules in ROOT, we will align package manager's activity together with it to develop a simpler way to use software libraries and in the same time provides better compile-time scalability and eliminates problems to access the API of a library. Additionally, we try to see how we can solve a problem of versioning and binary distribution of modules among others problems.

Input and Output of Scientific Datasets / **18**

ROOT I/O compression algorithms

Authors: Brian Paul Bockelman[1]; Oksana Shadura[1]

[1] *University of Nebraska Lincoln (US)*

Corresponding Authors: brian.bockelman@cern.ch, oksana.shadura@cern.ch

We present an investigation into the usage of compression in the ROOT I/O subsystem and its impact on ROOT-related performance and file size. The goal of the talk is to explain how compression is used in ROOT and which algorithms could be most interesting in terms of ROOT performance, while describing pro and cons of compression algorithms in details. We provide a comparison analysis of ROOT compression algorithms as a results, delivered by ROOT I/O performance benchmarks.

Parallelism, Heterogeneity and Distributed Data Processing / **19**

RDataFrame: ROOT's Declarative Approach for Manipulation and Analysis of Datasets

Authors: Enrico Guiraud[1]; Danilo Piparo[2]

[1] *CERN, University of Oldenburg (DE)*

[2] *CERN*

Corresponding Authors: enrico.guiraud@cern.ch, danilo.piparo@cern.ch

ROOT's Declarative Approach for Manipulation and Analysis of Datasets

ROOT proposed a modern, declarative approach to the treatment of columnar datasets, RDataFrame. Conceived initially as a way to implement functional chains, RDataFrame became a highly performant Swiss-Army knife for dataset manipulation and analysis.
This contribution discusses RDataFrame's minimal and modern interface from the programming model point of view and how it allowed to manage internally performance optimisations and parallelism. The blurring of compiled and jitted code aiming to a simplification of the programming model is discussed: a unique feature in the C++ landscape to our knowledge.

Parallelism, Heterogeneity and Distributed Data Processing / **20**

Utilities for parallelism at task-level and data-level in ROOT

Author: Xavier Valls Pla[1]

[1] *University Jaume I (ES)*

Corresponding Author: xavier.valls.pla@cern.ch

In order to take full advantage of the current computer architectures and to improve performance with increasing amounts of data to analyze, we developed tools for the parallelization of ROOT at task-level and integrated libraries for its parallelization at data-level. This tools have been extensively deployed throughout ROOT, from vectorization and parallelization of the fit to the parallel processing of the event loop in RDataFrame. We analyze the significant performance improvement they provide and report on different strategies we followed to improve their performance.

Scientific Visualisation, GUIs and Documentation / **21**

The grammar of graphics and ROOT

Author: Xavier Valls Pla[1]

[1] *University Jaume I (ES)*

Corresponding Author: xavier.valls.pla@cern.ch

The tidyverse is a collection of R packages for data science sharing an underlying design philosophy and data structures similar to RDataFrame. ggplot is the graphics library component of the tidyverse, implementing the concepts behind The Grammar of Graphics. ggplot allows to express our analysis in a declarative, layered, functional way, closely related to the programming model behind RDataFrame.

We compare several ggplot2 visualizations with their current approximate implementation in ROOT and viceversa, showcasing how ROOT's programming model would benefit from an approach similar to the one adopted by ggplot2, a perfect fit for the new programming paradigm introduced by RDataFrame.

Platforms, Infrastructure and Builds / **22**

ROOT 6 on Windows (finally)

Author: Bertrand Bellenot[1]

[1] *CERN*

Corresponding Author: bertrand.bellenot@cern.ch

After (too) many years without any ROOT 6 release on Windows, we got our first ***preview*** release of ROOT 6 on Windows (6.14/00) and Visual Studio 2017. This presentation will expose some of the issues we had, what is the current status, what is missing, what has to be done to port ROOT on Windows 64 bit (horizon 2020?), and more generally what people should think of in order to write portable code.

Scientific Visualisation, GUIs and Documentation / **23**

ROOT Graphics

Author: Olivier Couet[1]

[1] *CERN*

Corresponding Author: olivier.couet@cern.ch

ROOT graphics had many key improvements since the last workshop, for instance ratio plots, automatic placement of legends, automatic coloring, and the highlight technique. We will present the most relevant recent developments.

Scientific Visualisation, GUIs and Documentation / **24**

ROOT 7 Graphics

Author: Olivier Couet[1]

[1] *CERN*

Corresponding Author: olivier.couet@cern.ch

In the context of ROOT 7 a new web based model for Graphics is being developed using web technologies (html, JavaScript). We will present the developments already achieved, the missing features and the foreseen functionalities. This new environment already has the basic graphics classes: RCanvas, RFrame, basic primitives and attributes and coordinates system; even ROOT6 histograms etc can be painted in the new canvas. Possible solutions for batch output will be presented.

Scientific Visualisation, GUIs and Documentation / **25**

ROOT Documentation

Author: Olivier Couet[1]

[1] *CERN*

Corresponding Author: olivier.couet@cern.ch

Doxygen is the tool ROOT used to produce the reference guide and markdown/pandoc for the all other documents. A doxygen filter has been developed the past few years coupling closely ROOT and the reference guide generation. For instance this allows to generate on the fly high definition images for inline macros' code and tutorials, to produce the tree of needed libraries for each class. Future developments, like showing tutorials' icons preview to ease tutorials finding, will be exposed.

Scientific Visualisation, GUIs and Documentation / **26**

ROOT 7 Fit Panel

Author: Iliana Betsou[None]

Corresponding Author: iliana.betsou@cern.ch

The Fit Panel is a GUI to interactively fit histograms using all the power of the ROOT fitting tools. It can fit histograms and graphs with a predefined function, or a function defined by the user. In addition, we can edit different choices for the fitting, select a specific section of the histogram to fit or select different drawing option. The ROOT 7 version of this important GUI element is based on a new technology using the open source web GUI toolkit OpenUI from SAP company. This talk will present the various challenges we had to face during the development, for instance the definition of a proper layout, or communication between the client and the server.

Scientific Visualisation, GUIs and Documentation / **27**

New tools to build web-based graphics and user interfaces in ROOT

Author: Serguei Linev[1]

[1] *GSI Darmstadt*

Corresponding Author: s.linev@gsi.de

For two decades, ROOT brought its own window system abstraction (for X11, GL, Cocoa, and Windows) together with its own GUI library. X11 is nearing the end of its lifetime; new windowing systems shine with performance and features. To make best use of them, the ROOT team has decided to re-implement its graphics and GUI subsystem using web technology.

To enable development of new web-based components RWebWindow class was introduced to ROOT. It follows clear client-server concept, which are likely similar for any application that wishes to offer remote GUI and graphics. Here the server is a C++-application, which produces data and prepares it for visualization. The client(s) will be JavaScript-based code, creating a HTML/SVG/WebGL representation of the provided data and implements all kind of interactivity – zooming, tooltips, context menus, etc. The communication will be done via websocket-based protocol, allowing data push from the server side. Local displays will be implemented with libraries like Chromium Embedded Framework (CEF).

Taking new TCanvas as prominent example, different aspects of new concept will be discussed: data organization on application side; creation of data model and versioning of different components in the canvas; communication patterns with multiple clients; efficient JavaScript-based rendering with significant amount of code sharing with JSROOT. Example with new FitPanel can be used to show how main interactive part can be implemented with open-source GUI library, but main fitting job will be performed with standard ROOT methods.

Platforms, Infrastructure and Builds / **28**

Future of ROOT runtime C++ modules

Authors: Yuka Takahashi[1]; Vasil Georgiev Vasilev[2]

[1] *University of Cincinnati (US)*

[2] *Princeton University (US)*

Corresponding Authors: yuka.takahashi@cern.ch, vasil.georgiev.vasilev@cern.ch

This talk shows the status of the C++ Modules in ROOT and CMSSW. We will demonstrate performance improvement in ROOT and in CMSSW. Runtime C++ module improves correctness and simplifies the current implementation which relies on rdict_pcms and rootmap files. We would like to describe the major challenges: improving both ROOT and Clang. We will describe the current state of the experimental implementation. The talk proposes how we can further reduce the duplicate

information in the C++ module files and how to improve their load time. The authors share performance results and implementation experience gained when migrating CMSSW and its dependencies such as HepMC, Geant, and boost.

Multivariate Analysis / **30**

Cross validation and Parallelisation for TMVA

Author: Kim Albertsson[1]

[1] *Lulea University of Technology (SE)*

Corresponding Author: kim.albertsson@cern.ch

Cross validation, an important tool for optimising and understanding generalisation performance of large models (many tunable parameters), was recently introduced in TMVA. Currently implemented to estimate model performance, developments are ongoing for model selection. This talk will give an introduction to what CV is and how it is used in TMVA, with particular focus on HEP applications.

Concurrently efforts to parallelise evaluation of cross validation has been ongoing. We will show how to use the current implementation, performance benchmarks and where development is headed.

ROOT and the Python Ecosystem / **33**

iminuit: interactive python wrapper around MINUIT2

Corresponding Author: henry.fredrick.schreiner@cern.ch

iminuit is an external Python interface to the Minuit2 C++ code,
which can be compiled standalone without the rest of ROOT. iminuit
has recently seen a boost of development which culminated in the
latest 1.3 release and will join the Scikit-HEP project in this year.
To simplify Minuit's use as a standalone CMake package, for projects
like iminuit and GooFit, a new standalone build system
was implemented for Minuit2, and has been
included in the latest release of ROOT. This system uses modern CMake
patterns, and lives in a peaceful coexistence with the ROOT build
system. The production of source packages is handled without external
scripts, and the system even supports building from inside ROOT.
Integrating this into the ROOT source and build system provided several
challenges, with some interesting solutions that will be shown.

Parallelism, Heterogeneity and Distributed Data Processing / **34**

Integrating ROOT I/O with Apache Spark

Author: Viktor Khristenko[1]

[1] *CERN*

Corresponding Author: viktor.khristenko@cern.ch

The DEEP-EST is the European Project building a new generation of the Modular Supercomputer Architecture (MSA). The MSA is a blueprint for heterogeneous HPC systems supporting high performance compute and data analytics workloads with highest efficiency and scalability.

Within the context of the project, we are working on the JVM based implementation of the ROOT File Format, spark-root/root4j, together with an Apache Spark Data Source. Current implementation allows to directly ingest HEP data, perform stream/batch processing and integrate Machine Learning pipelines with Apache Spark.

In this talk, we first discuss the intricacies and internals of the JVM-based implementation. Interesting examples of "bootstrapping ROOT" File Format will be provided as a proof of the robustness and simplicity of the structure of the format itself.

Furthermore, considering Apache Spark constitutes a query execution engine, comparisons of ROOT/c++ based workloads to Apache Spark based ones will be provided and discussed.

Platforms, Infrastructure and Builds / **35**

Static code analysis on the ROOT source code

Author: Wolf Behrenhoff[1]

[1] *Former CMS Member*

Corresponding Author: wolf@behrenhoff.de

In recent years, static code analysis tools for C++ have made huge improvements. This talk shows how to use clang-tidy and reports about the results applying it to the ROOT source code. As the resulting log is huge, finding relevant results can be challenging. Nevertheless, a few problems in ROOT's source code and even a bug in LLVM were found this way. In addition, it is shown how one can use clang-tidy's automatic code modernization tools and what problems there are when modernizing ROOT's source.

End User Perspective / **36**

Experience and views on ROOT

Author: Wolf Behrenhoff[1]

[1] *Former CMS Member*

Corresponding Author: wolf@behrenhoff.de

This talk reports on experiences and lessons learned moving from particle physics via software development to data science. It explains why we are still using ROOT from time to time and highlights some of its features making data analysis simple and fast. It also presents a few ideas to improve common tasks in ROOT and TMVA. Some annoyances and problems with ROOT are shown as well.

End User Perspective / **37**

A C++11 histogram library for Boost

Author: Hans Peter Dembinski[1]

[1] *Max-Planck-Institute for Nuclear Physics, Heidelberg*

Corresponding Author: hdembins@mpi-hd.mpg.de

In 2016, development of an open source C++ library for multi-dimensional histograms in collaboration with the Boost community started, which leverages modern C++11 features such as variadic templates and template meta-programming (TMP). Variadic templates make code possible that works for arbitrary dimensions. TMP allows one to write histograms which compile to very efficient assembler while being easy to use and customize by users. The features of the library were developed with the needs of the high-energy and astroparticle physic community in mind and include the key features of ROOT histogram classes. We present the design and state of this library. An official review for inclusion in the Boost Library collection is planned for September 2018.

Parallelism, Heterogeneity and Distributed Data Processing / 38

Adding CUDA Support to Cling: JIT Compile to GPUs

Author: Simeon Ehrig[1]

Co-authors: Axel Naumann [2]; Axel Huebl [1]

[1] *Helmholtz-Zentrum Dresden-Rossendorf and TU Dresden*

[2] *CERN*

Corresponding Authors: s.ehrig@hzdr.de, axel.naumann@cern.ch, a.huebl@hzdr.de

We present the results of a diploma thesis adding CUDA (runtime) C++ support to cling. Today's HPC systems are heterogeneous and get most of their computing power from so-called accelerator hardware, such as GPUs. Programming GPUs with modern C++ is a perfect match, allowing perfectly tailored and zero-overhead abstractions for performance-critical "kernels".

Nevertheless, tool complexity in development and debugging can be discouraging for new users. We are addressing this by not only adding low-level support for accelerators but also by going up the open source software-stack enabling interactive, CUDA C++ Jupyter notebooks, e.g. through xeus-cling.

Parallelism, Heterogeneity and Distributed Data Processing / 39

Data Analysis and Simulations in Exascale Computing: Quō vādis?

Author: Axel Huebl[1]

Co-authors: Simeon Ehrig [1]; Michael Bussmann [2]

[1] *Helmholtz-Zentrum Dresden-Rossendorf and TU Dresden*

[2] *Helmholtz-Zentrum Dresden - Rossendorf*

Corresponding Authors: s.ehrig@hzdr.de, m.bussmann@hzdr.de, a.huebl@hzdr.de

We are less than three years apart from the first, double precision Exa-Flop/s supercomputers. Already today, our scientific software stacks are facing the challenge to run efficiently on a potpourri of architectures. But the real troubles might await us at the choke points of extreme data rates, where traditional workflows of data acquisition, filtering, processing and subsequent long-term storage might not be able to be sustained anymore.

How would you like to express your scientific algorithms in a world where Flop/s are increasingly cheap, yet hard to achieve, but data movement and especially data at rest is increasingly inproportionally expensive? Would you be OK to throw data away and measure twice? Can we in situ compute results with a different prepared question instead of waiting for an always-full and quickly-purged filesystem? How do we ensure reproducibility? How large a mix of programming languages and double-implementations of algorithms can we burden before we are running out of developers (due to lack of maintainability)?

This talk will present our vision for the next years of data-driven scientific computing. Based on our experience with single-source, performance-portable C++ HPC libraries, we will present zero-overhead C++ abstractions that spare code-duplication. Together with light-weight code coupling, possible directions for analyzing resulting data rates are discussed on examples from laser-driven particle accelerator research. With such meta-programming approaches, an underestimated risk lies in cutbacks for both development workflows and user interactivity at runtime, which we want to openly change with interactive Cling-assisted execution in modern environments such as Jupyter, for which we recently enabled CUDA C++ capabilities.

Parallelism, Heterogeneity and Distributed Data Processing / 40

Asynchronous, parallel computations for complex simulation tasks in ROOT.

Author: Jochen Kerdels[1]

[1] *FernUniversität in Hagen*

Corresponding Author: jochen.kerdels@fernuni-hagen.de

The simulation of complex systems can involve a high degree of dependencies and interaction between different parts of the simulation. A standard approach to handle these dependencies is the use of a double-buffered global state where all components of the simulation are processed synchronously in lockstep. Parallelization of such an approach has the drawback that it requires a synchronization point, e.g., a barrier, for each step of the simulation. This essentially introduces a part of the simulation that is non-parallelizable resulting in a limit of the achievable speed-up as described by Amdahl's law. One alternative solution that avoids this problem is asynchronous processing where the synchronization between different parts of the simulation is encapsulated by a set of independent local buffers. In this presentation I showcase the design and use of such a simulation approach (implemented with the ROOT framework) that is used for the development of computational models of neurons in the mammalian cortex.

Experiments Perspective / 41

Belle II Software Framework - Overview, Feedback and Requests

Author: Martin Ritter[1]

[1] *LMU / Cluster Universe*

Corresponding Author: martin.ritter@lmu.de

This talk will give a short overview over the Belle II Software Framework and its use in offline and online data processing. It will highlight the use of ROOT for data storage and inter-process communication as well as our Python 3 integration and user interface. Finally we will present some feedback and requests, primarily from the offline side.

Experiments Perspective / **42**

Evolution of ROOT geometry package

Author: Andrei Gheata[1]

[1] *CERN*

Corresponding Author: andrei.gheata@cern.ch

The talk will present the plans for the evolution of the ROOT geometry package in relation with VecGeom development. The future integration of VecGeom performance features and the impact on external frameworks depending on TGeo (such as DD4HEP or VMC) will also be discussed.

Experiments Perspective / **43**

Reliable Histogram Aggregation for the LHCb software trigger

Author: Roel Aaij[1]

Co-author: Stefano Petrucci [2]

[1] *Nikhef National institute for subatomic physics (NL)*

[2] *University of Edinburgh, CERN*

Corresponding Authors: roel.aaij@cern.ch, s.petrucci@cern.ch

The second stage of the LHCb software trigger application consists of an up-front event reconstruction followed by approximately 500 independent event selections. Monitoring of the performance of the reconstruction and selections is crucial to detect and address issues that may appear as soon as possible. To this end, each process produces approximately $3 * 10^3$ monitoring histograms, resulting in $1.35 * 10^8$ histograms that need to be aggregated. To achieve this, a tiered architecture of monitoring infrastructure tasks has been put in place, which separately propagates histogram descriptions and histogram increments using ØMQ. As this separation is not possible with ROOT, it is based on standard containers and a few custom classes serialized using boost::serialization.

As a result of the asynchronous processing, different nodes may process data from different data-taking intervals. This results in up to $5 * 10^5$ ROOT histograms, belonging to 300 data-taking intervals, that need to be written to a single file per interval every few minutes. Improvements in ROOT's parallelization have allowed simplification of the code, but some bottlenecks are still present.

Multivariate Analysis / **44**

New Deep Learning Tools in ROOT/TMVA

Author: Lorenzo Moneta[1]

[1] *CERN*

Corresponding Author: lorenzo.moneta@cern.ch

In this talk, we will describe the latest additions to the Toolkit for Multivariate Analysis (TMVA), the machine learning package integrated into the ROOT framework. In particular, we will focus on the new deep learning module that contains robust fully-connected, convolutional and recurrent deep neural networks implemented on CPU and GPU architectures. We will present performance of these new libraries on benchmark datasets from high-energy physics.
We will show also performance comparisons with respect to other machine learning frameworks, such as Tensorflow and Keras.

ROOT and the Python Ecosystem / **45**

PyrooFit: a python interface to RooFit used in the Belle II collaboration

Author: Simon Wehle[1]

[1] *DESY*

Corresponding Author: simon.wehle@desy.de

In this talk we present PyrooFit, a fit framework based on python and pandas DataFrames built on top of the ROOT.RooFit package. It currently allows for simple fits of standard PDFs and easy setup of custom PDFs in one or more fit dimensions, but the ultimate goal is to develop it into a general framework independent of a specific back-end.

Finally in this talk we report on general feedback collected by the Belle II analysis community so far.

Experiments Perspective / **46**

Use of ROOT in ILCSoft

Authors: Remi Ete[1]; Frank-Dieter Gaede[2]

[1] *DESY*

[2] *Deutsches Elektronen-Synchrotron (DE)*

Corresponding Authors: remi.ete@desy.de, frank-dieter.gaede@cern.ch

The International Linear Collider is a linear e+e- collider project expected to be built in Japan. In order to study the potential physics discoveries with such a collider, the ILCSoft framework has been developed by the ILC and CLIC collaborations. As for most of the experiment frameworks, ROOT plays a central role in ILCSoft. In particular, the DD4hep package, based on the TGeo component of ROOT, provides an interface to Geant4 for simulation, geometry description for reconstruction and conditions access for data analysis. The ROOT TMVA component is also extensively used for physics analysis.

In this talk we will review the usage of ROOT in the ILCSoft framework and its use by the linear collider community.

Experiments Perspective / **47**

ROOT in LHCb Software Framework Gaudi, present and future

Author: Marco Clemencic[1]

[1] *CERN*

Corresponding Author: marco.clemencic@cern.ch

CERN LHCb experiment has been using ROOT for 20 years as back-end for storing the reconstructed event data and the stripped down data used for physics analysis.

With the ongoing work to prepare the experiment software for the upgrade of the detector happening during LHC second Long Shutdown, we are reviewing the LHCb software framework Gaudi to improve the code base in terms of performance, usability and maintainability, and ROOT being such an important piece of our software stack, we are considering how to improve its use and prepare for the ROOT v7 API.

In this contribution, we will outline our current use of ROOT and how we plan to use it in out future software.

Scientific Visualisation, GUIs and Documentation / **48**

Interactive documentation for advanced statistical modelling with RooFit and RooStats

Author: Vince Croft[1]

Co-author: Wouter Verkerke [2]

[1] *New York University (US)*

[2] *Nikhef National institute for subatomic physics (NL)*

Corresponding Authors: verkerke@nikhef.nl, vincent.croft@cern.ch

The core driving force behind the data intensive nature of research in particle physics, is the level of statistical precision achievable in measuring the parameters of the Standard Model. The RooFit and RooStats data modelling packages are the gateway that allows users to interface the complex data processing abilities provided by ROOT and the precise statistical inference tools required to probe the nature of the standard model.

Typical modern LHC analyses are described at the likelihood level by a statistical model containing a large number of signal, control and validation regions. To streamline this model building process, several higher-level languages have been developed to allow users to bypass many of the statistical concepts of the underlying RooFit probability models. However, given the complexity of such models, users need a good grasp of both high-level and low-level model building concepts and its interactions with the RooStats statistical analysis tools to understand the computational performance of the fit, to validate it's proper convergence and be able to successfully validate the final physics results.

We report on an effort to setup a new documentation project that blends descriptions of both high-level and low-level modeling tools, with links and branches to general descriptions of the underlying statistical methods, and links to 'good practices' of modelling building that have emerged from LHC analyses. An important feature of this comprehensive documentation that covers both statistical methods as well as software tools, that all code examples, both in the statistics text as well as in the tool documentation are interactive through CERN SWAN service. Centrally maintained documentation such as this ensures that all statistical tools benefit from the latest improvements in ROOT performance, new features and tools are more easily incorporated into the core functionality and issues may be tracked more efficiently.

Experiments Perspective / **49**

The SHiP perspective on ROOT

Author: Oliver Lantwin[1]

[1] *Imperial College (GB)*

Corresponding Author: oliver.lantwin@cern.ch

SHiP is an experiment that is currently being designed to look for hidden particles beyond the Standard Model at CERN's SPS. As a SHiP PhD student working (among other things) on its software, I will give an overview of the SHiP software stack, with a focus on how we use ROOT, and how we would like it improve.

I will cover some aspects that are common to small experiments using ROOT, but also some aspects unique to SHiP, which has some uncommon requirements to its software due to the nature of the experiment.

Additionally I will give some personal perspectives on using ROOT for online and offline computing and pyROOT and the scientific python stack for detector optimisation and physics studies.

Experiments Perspective / **50**

ROOT across the CMS experiment

Author: David Lange[1]

[1] *Princeton University (US)*

Corresponding Author: david.lange@cern.ch

We will discuss ROOT and its impact in the CMS offline production software, analysis and online systems. We will identify ways that CMS relies on ROOT components, which span a broad space that reaches most aspects of the experiment software stack. We'll focus our discussion on current development activities and lessons learned around areas of software operations, data storage, I/O, data quality monitoring, python-based analysis, interoperability and other topics identified by our user community.

Input and Output of Scientific Datasets / **51**

Writing ROOT files with uproot

Authors: Pratyush Das[1]; Jim Pivarski[2]

[1] *Institute of Engineering and Management, Kolkata, West Bengal*

[2] *Princeton University*

Corresponding Authors: pivarski@fnal.gov, reikdas@gmail.com

The uproot package provides access to ROOT data in several new ways: (a) natively in Python+Numpy data types, (b) installable without specialized binary dependencies, and (c) with columnar, array-at-a-time performance characteristics. This talk presents a new feature in uproot: the ability to write ROOT files, rather than just reading them. In addition to outlining the scope of which kinds of objects are writable and which aren't, we will present some key design choices, such as the use of memory-mapped files, cursors, streamer/Python class mapping, and mix-in methods.

Parallelism, Heterogeneity and Distributed Data Processing / 52

Vectorized processing of nested data

Authors: Jim Pivarski[1]; David Lange[2]; Jaydeep Nandi[3]

[1] *Princeton University*

[2] *Princeton University (US)*

[3] *National Institute of Technology, Silchar, India*

Corresponding Authors: david.lange@cern.ch, nandi.jaydeep296@gmail.com, pivarski@fnal.gov

Array-at-a-time processing is key for performance of low arithmetic intensity calculations, such as plotting, because of sequential memory access and SIMD parallelization. However, HEP data typically have nested structures and HEP algorithms typically require inner loops. We will present techniques for manipulating arrays of nested data to perform combinatoric calculations (e.g. pairs of particles per event) without explicit loops, as well as high-level idioms to express them in a data analysis.

Experiments Perspective / 53

ROOT usage by the art framework and its users

Author: Kyle Knoepfel[1]

[1] *Fermi National Accelerator Laboratory*

Corresponding Author: knoepfel@fnal.gov

The *art* framework, used by roughly a dozen HEP experiments, provides simple interactions with ROOT for its users. In addition to facilities that enable dictionary generation and ROOT-based input sources and output modules, *art* also provides easy user interaction with ROOT constructs from within the framework.

We describe how *art* and its experiments use ROOT, both within the framework and outside of it. We also discuss some of ROOT's strengths upon which *art* relies, and we propose some enhancements.

End User Perspective / 54

ROOT prompt usability features and the strict option

Authors: Yuka Takahashi[1]; Vasil Georgiev Vasilev[2]; Axel Naumann[3]

[1] *University of Cincinnati (US)*

[2] *Princeton University (US)*

[3] *CERN*

Corresponding Authors: yuka.takahashi@cern.ch, axel.naumann@cern.ch, vasil.georgiev.vasilev@cern.ch

The aim of this talk is twofold: to summarize ROOT prompt options by simple use cases, and to introduce the newly introduced –strict flag. ROOT prompt, which runs C++ interpreter in its background, supports many features while not many of those are known to users. In this talk, we would like to summarize these options classified by simple use cases.

Second part is dedicated to an option "–strict". Currently, ROOT prompt comes with automatic #include and resolving namespaces. However when debugging users' code with ROOT, this feature messes up with users code as ROOT library contains thousands of variables like "PI" and standard library functions. We introduced a "–strict" flag which enables users to get a clean C++ interpreter, without C++ superset supports. In this talk, we will show the use cases of this new option.

Experiments Perspective / **55**

The Usage of ROOT in the DD4hep Detector Description Package

Authors: Markus Frank[1]; Frank-Dieter Gaede[2]; Marko Petric[1]; Andre Sailer[1]

[1] *CERN*

[2] *Deutsches Elektronen-Synchrotron (DE)*

Corresponding Authors: andre.philippe.sailer@cern.ch, frank-dieter.gaede@cern.ch, markus.frank@cern.ch, marko.petric@cern.ch

DD4hep is a detector geometry package which can provide all auxiliary information necessary to process data from particle collisions in high energy physics such as geometry and readout information, but also interfaces to conditions and alignment data. DD4hep supports all data processing activities of an experiment: simulation, reconstruction and analysis.

The modeling of the geometry of a detector is entirely based on the ROOT geometry toolkit TGeo. DD4hep further extensively uses the ROOT core to allow persistent descriptions and the pyROOT interface to configure simulation processes with Geant4. Due to the increasing popularity we would like to provide our input concerning possible future improvements or development directions.

End User Perspective / **56**

Graduate student case studies

Author: Michael Kent Wilkinson[1]

[1] *Syracuse University (US)*

Corresponding Author: miwilkin@syr.edu

I have been using ROOT as an undergraduate and graduate student for about seven years now. The first scripts I ever wrote were ROOT macros. From naive histogram-filling in MakeClass.C to writing python wrapper-classes and analyzing unbinned fits, much of my development as a programmer has taken place in the context of ROOT.

In this talk, I present needs and examples of ROOT usage from fellow graduate students at my university, alongside examples of my own development as a ROOT user working in particle physics.

Experiments Perspective / **57**

N dimensional analysis pipeline for ALICE

Authors: Marian Ivanov[1]; Boris Rumyantsev[2]

[1] *GSI - Helmholtzzentrum fur Schwerionenforschung GmbH (DE)*

[2] *Joint Institute for Nuclear Research (RU)*

Corresponding Authors: marian.ivanov@cern.ch, boris.rumyantsev@cern.ch

We are going to present the N-dimensional analysis pipeline of the ALICE experiment, including creation of materialized views, n-dimensional histogramming package, statistical maps extraction, local regression and optimizer of physical models.
Besides we will demonstrate how the pipeline can be interfaced to the TMVA package.

We will show multiple examples of the pipeline usage in the ALICE QA, calibration and performance parametrization monitoring.

Input and Output of Scientific Datasets / 58

ROOT I/O Past, Present and Future.

Authors: Philippe Canal[1]; Danilo Piparo[2]

[1] *Fermi National Accelerator Lab. (US)*

[2] *CERN*

Corresponding Authors: danilo.piparo@cern.ch, philippe.canal@cern.ch

What is ROOT I/O? What are its strengths? What are its weakness? How does it compare to alternatives? This presentation will address these questions and present performance contrasts with other solutions. We will then review the latest additions, fixes and features. Finally, we will give our vision of where we can take ROOT I/O to make it even better and more relevant to today's environment and challenges.

End User Perspective / 61

Experience with using ROOT for final ATLAS data analysis steps

Author: Tadej Novak[1]

[1] *Jozef Stefan Institute (SI)*

Corresponding Author: tadej.novak@cern.ch

Most of the ATLAS analyses do their final step by processing TTree-based ntuples. As actual analysis selection optimisations take place at this step, it is crucial that one can process all events in a timely fashion. A (personal) experience writing a set of ROOT-based algorithms to process ntuples and create final histograms will be presented. More than 2 TB of data can be processed in about half a day with the help of simple parallelisation using different distributed computing infrastructures. The results can be used for the final publication plots.

End User Perspective / 62

Coding path of young physicist

Author: Ziga Brencic[1]

[1] *master student*

Corresponding Author: ziga.brencic@cern.ch

I would like to share my path of learning ROOT and acquiring software skills as a young physicist. I started learning ROOT in my 2nd year of physics bachelors with little of programming knowledge and some experience in C. I found mostly outdated docs and code from old guard physicists (overuse of pointers, new, delete, single file long macros, no OOP, ...). What followed was 3 years of learning how to properly code. Sadly I learned most from computer scientists or professional developers and not as much as I would have hoped from the physics community. Now that I'm finishing my masters I somehow feel confident regarding my coding skills (yet still a long way to go). I also know that I could have achieved the level of knowledge I have way faster with proper systematic guidance from the community. I think sharing my experience of what worked for me could help others struggle less on their coding path.

End User Perspective / **63**

SEASTAR analysis with ROOT6

Author: Axel Frotscher[1]

[1] *TU Darmstadt*

Corresponding Author: axel.frotscher@googlemail.com

ROOT is a universal data analyzing library, which is not only used by particle physicists, but also by nuclear physicists. This presentation will focus on the data analysis of "small" datasets (10's GB) from a typical nuclear physics experiment, parallelization for beginners and a viable alternative to the current TTreeReader class. Missing features will also be addressed from a user perspective.

Invited Talks / **66**

Thoughts and ideas of a previous actor and now project's observer about ROOT's future

Corresponding Author: rene.brun@cern.ch

I would like to present some thoughts and ideas may be useful for the future based on one side on my previous experience as an actor and now observer of the project since a few years, and on the other side on many comments/suggestions received from many users.

Invited Talks / **68**

Particle physics MasterClasses and future developments

Corresponding Author: yiota.foka@cern.ch

The International project physicsmasterclasses.org brings the excitement of cutting-edge particle-physics research into the classroom including hands-on experience with real experimental data. In 2018 about 15 000 students, 15 to 19 years of age, participated in this popular event over 6 weeks in 52 countries, hosted by 225 institutes. The existing LHC masterclasses are used in many other occasions while communities beyond LHC also start implementing their analyses (e.g. IceCube, Belle...).

I will discuss a proposal towards developing a general, experiment independent framework that could allow the implementation of further masterclasses in an economic way for developers and could ensure easy use at different environments.

As a first step, a CERN summer student project was dedicated at improving and expanding the current ALICE masterclass, developed in 2010 based on ROOT, with the goal to structure and prepare a framework for future developments. The next steps aim at using data from different experiments but also introducing new analyses and data samples.

A more ambitious part of the project will focus on taking advantage of powerful tools for browser based data analysis such as the ones that are becoming available within ROOT. Browser-based masterclasses would greatly improve easy-of-use, as no installation of software is required. Browser-based 'notebooks' can either run via the CERN SWAN service, or a SWAN service can be deployed on local resources. In this phase of the project, the possibility to run a masterclass as a browser-based notebook will be explored, with the goal of implementing a 'pilot' masterclass.

Invited Talks / 69

Making C++ Easier, Faster, and Safer

C++ remains one of the best languages for writing performance sensitive software. However, it has proven to pose significant challenges for developers, especially when a single codebase is evolved over a long period of time. We have taken an extremely large, complex C++ codebase at Google from being a huge risk and liability, into something that is sustainable, with both productive and effective developers.

This required a very wide range of efforts to accomplish. While we have discussed on several occasions specific efforts here, this talk will tie everything together to explain how the sustainable C++ experience at Google works. It will cover all of the highlights and give a lot of references for the audience to dig into afterward.

Ultimately, our goal is to make C++ easier, faster, and safer, both for our developers and the wider industry. This talk will give the ROOT community a view into our approach at Google.

Invited Talks / 70

UI5: SAP's (open source) JavaScript UI library and its evolution

UI5 is SAP's JavaScript UI library to build enterprise-ready web controls and applications, being responsive to all devices and running on almost any browser of your choice. It comes in two flavors, OpenUI5 with an open-source codebase available under the Apache 2.0 license and SAPUI5 extending the OpenUI5 codebase with SAP specific features. Driven by SAP's demanding environment, UI5 worked in its formation phase mainly on growing into an enterprise-grade UI framework fulfilling all SAP product standards, implementing the SAP Fiori design guidelines and simplifying the development and maintenance of web controls and applications. But during that time the technical innovation of UI5 has been neglected. To strengthen the technical innovation we started the UI5 evolution project. UI5 Evolution deals with topics: renovation and modularization of the Core

Framework, coming with a modern open-source Node.JS based Build and Development tooling, providing an independent, simplified and open rendering framework.

This talk will provide you an brief introduction into the UI5 technology as such, the difference between OpenUI5 and SAPUI5, SAPs interest and commitment to Open Source, the extensibility of UI5, and the ability to work with mass data. Furthermore you will gain an insight into UI5 Evolution and get an impression what we achieved so fare and what is in the pipeline.

Parallelism, Heterogeneity and Distributed Data Processing / 71

Future Accelerators: Different Languages, Different Platforms

Corresponding Author: danilo.piparo@cern.ch

For years we enjoyed on the Grid and our computer centres a uniformity in the computing hardware at our disposal. This will most probably change in the future. What does ROOT need to provide to be able to exploit heterogeneous architectures? What kind of capabilities will our interpreter need? What will be the programming model?
For years we also enjoyed a certain uniformity in the programming languages of our software. C++ and Python demonstrated to be a winning combination for HEP. Will this tandem of languages be the one we'll be using in 10 years from now? If not, is it more likely that the number crunching player is replaced by something else or will we rather replace our powerful glue? In all scenarios, how can ROOT be the framework at the centre of HEP data processing?

Opening / 72

Welcome

Invited Talks / 74

Jupyter is Going Native! C++ as a First-Class Citizen of the Ecosystem

The language-agnostic architecture of Project Jupyter enables the rich features of the Jupyter frontend for a large number of programming languages beyond Python, including Julia and R. However, despite the importance of the C++ scientific computing stack, adoption of Jupyter has remained limited in this community because of the compiled nature of the programming language.

In this presentation, we demonstrate the Xeus-Cling Jupyter kernel, which is built upon the Cling C++ interpreter from CERN and the Xeus C++ implementation of the Jupyter protocol. The Xeus-cling kernel includes features such as:

- availability of quick help for any type or variable
- the rich MIME type rendering for images, videos, or any MIME type for which renderers are available in JupyterLab or the classic Notebook
- xwidgets, a native backend to Jupyter interactive widgets, including all the widgets from the core jupyter-widgets library but also backends for bqplot (xplot), ipyleaflet (xleaflet), pythreejs (xthreejs), and ipyvolume (xvolume)

Then we dive into the ecosystem of available libraries for interactive scientific computing in C++, such as xtensor for lazy array-based computing and xframe for labeled arrays and datasets.

Experiments Perspective / **76**

CMS DQM

Corresponding Author: marcel.andre.schneider@cern.ch

77

Closing Remarks and Discussion

Corresponding Author: axel.naumann@cern.ch

End User Perspective / **78**

User feedback from LHCb

Corresponding Author: hdembins@mpi-hd.mpg.de

End User Perspective / **79**

TRooFit

Corresponding Authors: will@cern.ch, attila.krasznahorkay@cern.ch

TRooFit is an extension of the RooFit toolkit, providing a set of classes that are inspired by core ROOT objects (specifically histograms and histogram stacks) but are fully-functional RooFit pdfs or functions. TRooFit's TRooH1D and TRooHStack are fillable and drawable in the same way as a TH1D and THStack, but are simultaneously concrete RooFit PDFs that can be fit to a RooFit dataset. TRooFit's histogram classes also incorporate desirable features such as systematic variation with automatic interpolation. TRooFit provides extensions of some RooFit classes (specifically the RooFitResult and the RooWorkspace), which assist in the creation of new models, as well as inspection of existing ones. In particular, TRooFit's TRooWorkspace is able to interpret and draw workspaces built with the HistFactory tool and other tools that build models that have a similar structure.
This presentation will demonstrate using TRooFit to visualise an existing workspace, run a fit, inspect the fit result (through pull and impact plots) and obtain pre- and post-fit yields and uncertainties. Some examples of using TRooFit to build models will also be given.

End User Perspective / **80**

Questions and Discussion

End User Perspective / **81**

Boosting HEP and Computer Sciences' knowledge transfer with ROOT

Corresponding Author: giskya@gmail.com

One of the significant challenges in the development of the High Energy Physics (HEP) field is the fact that many potential -and valuable- students and young researchers live in countries where internet access and computational infrastructure are poor compared to institutions already participating in multinational experiments like those hosted at CERN. In order to accelerate the process, several projects and organisations release, produce and use Open Source tools and data to create university-level training in HEP and Computer Sciences among different institutions in South America. With the primary goal of having resources that are easy-to-deploy in different environments. ROOT and its different tools and application have been crucial for such projects. We look to present how ROOT is used as the engine of physics analysis, computational and visualisation apps for science, education and outreach projects worldwide. As a concrete example, we will share how a post-graduate project was successfully developed between Venezuela-Switzerland-Argentina thanks to ROOT and its developers.

End User Perspective / **82**

Questions and Discussion

Multivariate Analysis / **83**

Questions and Discussion

Platforms, Infrastructure and Builds / **84**

Questions and Discussion

Input and Output of Scientific Datasets / **85**

Input and Output of Scientific Datasets: Discussion Session