ROOT in the HEP Software Foundation

Pere Mato/CERN on behalf of ROOT Team
HSF Workshop, SLAC, 28th November 2014

Why ROOT is interested in HSF?

- ROOT is one of the essential software packages in basically 100% of HEP experiments software stacks
  - I/O, histogramming, statistics, graphics, interactivity, scripting, data analysis, etc.
- If we build **the** HEP Software Foundation, ROOT must to be part of it
- We need to better integrate all HEP software components, and ROOT is an essential ingredient
- ROOT has played the role of 'hosting' contributions that are useful to the HEP community
  - Providing build & testing infrastructure, integration, distribution, licensing, support infrastructure, etc.
    ==> makes the life easier to users
- HSF should just generalize what ROOT has been doing so far

ROOT
Data Analysis Framework

Opportunity for Contributors

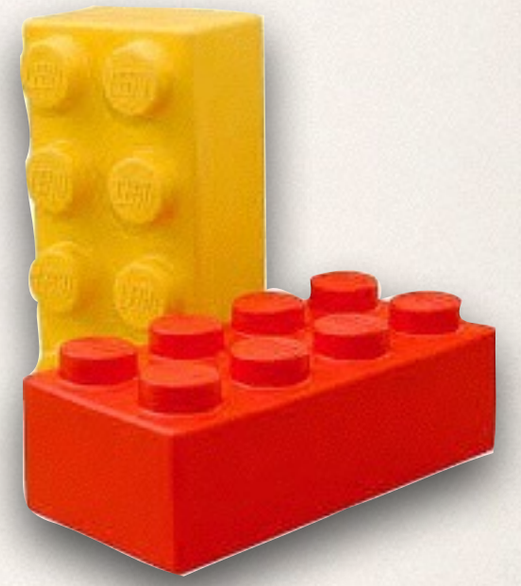
- We would like to facilitate contributions to ROOT without engaging our responsibility in the maintenance and user support
  - layered software modules or plugins that can bring new functionality to the end-users
  - e.g. systems like Jenkins/Drupal/R provides a platform for developers to contribute in an easy manner
- ROOT is 20 years old, and some parts requires re-engineering
- Contributions can be at the beginning on peripheral functionality and later on the core functionality once we know the direction we are going
  - Exploit modern hardware (many-core, GPU, etc.) to boost performance
  - Modernize implementations (C++11 constructs, use existing libraries, etc.)
  - Need to solve the backward compatibility

ROOT Main Directions

- **Cling Interpreter and its full exploitation**
  - C++11/14, JIT compilation opens many possibilities (e.g. TFormula, automatic differentiation, improved interactivity, etc.)
- **Parallelization**
  - Seek for any opportunity in ROOT to do things in parallel to better exploit the new hardware (e.g. Ntuple processing, I/O, minimization, etc.)
- **Packaging and modularization**
  - Incorporate easily third party packages (e.g. VecGeom in TGeom)
  - Build/install modules and plugins on demand. Facilitate contributors to provide new functionality
- **ROOT as-a-service**
  - Thin client plugged directly into a ROOT supercomputing cloud, computing answers quickly, efficiently, and without scalding your lap
- **Re-thinking user interface**
  - Explore new ways to provide thin-client web-based user interfaces

Open to External Contributions

- Core team should define a **model** for defining **modules** extending the functionality of ROOT
  - Provides the 'platform' with a number of core modules with the basic functionality
  - The ingredients are C++ introspection, modular build system, plugin system, flexible testing system



- Essential for making ROOT more open and encourage external people to contribute
  - Authors may keep ownership of their modules and provide user support (e.g. bug fixes, documentation)
  - Core team ensures consistency and provides support for the full life-cycle
- Support for on-demand installation of packages (with one or more modules)
  - Like is currently done in R

Current examples are FISTIO, Ruby, …

Main ROOT distribution cases

- ROOT standalone with a number of ‘options’ enabled
  - Typically end-users doing analysis with ROOT
  - Building ROOT from sources should be simple (single command)
    - Automatically including all the ‘core’ dependent packages on the same build
    - Built-in package versions selected by ROOT team
    - For the ‘options’ may need to provide external builds
- ROOT integrated as part of a larger software stack
  - Experiment’s applications
  - Typically all dependent **packages are externally provided**
    - Only way to ensure consistency
    - Versions selected by experiment librarians (within constrains)

HSF Services and Activities

Project hosting infrastructure	Not needed
Building and testing infrastructure	Not needed, except for esoteric platforms
Teams for certification and integration	ROOT tested together with the rest of HEP software can be very beneficial (e.g. LCG nightlies to test ROOT-6)
Software repositories and package managers	ROOT uses a number of external packages and standardizing on a set of package managers can be beneficial. ROOT may need to be adapted.
Access to computing resources on many platforms and architectures	Access to non-standard hardware, new hardware pre-views
Access to software development tools	Yes
Training in software technologies and tools	Coordination can be beneficial. We could provide training, materials, etc for ROOT, and receive training for software technologies and tools.
Support for IP and licensing issues	Not needed, but we should be ready to adapt to common practices
Peer reviews	Often the ROOT team is invited to reviews. ROOT can also benefit from external reviews
Access to scientific software journals	Yes, we need to worry about the career of our developers Agree on a common scientific software journal
Task forces or “SWAT” teams to solve specific issues	The ROOT team has participated often in such task forces
Consultancy for new experiments or projects	Often the ROOT team is invited to experiment discussions