Evolution of ROOT package management

Oksana Shadura, Brian Paul Bockelman, Vassil Vassilev
University of Nebraska–Lincoln, USA
oksana.shadura@cern.ch, bbockelm@cse.unl.edu, vvasilev@cern.ch

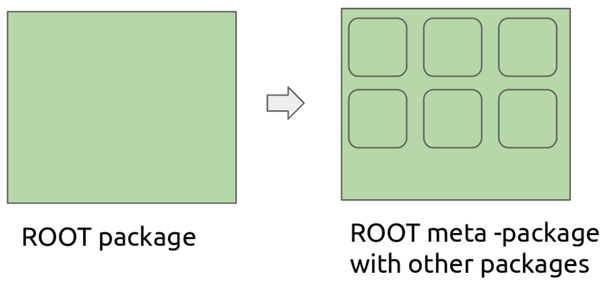
1. Motivation

1. **Build component/package of ROOT on top of installed ROOT**
2. Enable ROOT packaging become more flexible and less monolithic
3. Develop ROOT-aware dependency manager

2. ROOT-on-demand builds

ROOT C++ modules are released as a technology preview. This enables each ROOT library to be built on its own and possibly at a custom point in time.

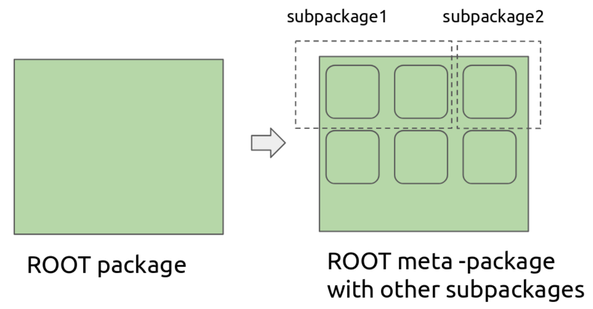
1. ROOT CMake Project ships only one package → ROOT
2. We want to extend ROOT possibility of support multiple "sub-packages"



3. ROOT CMake improvements

ROOT has around 110 CMake options, which are:

1. build features - 10 %. For example, cxx11, cxx14, pch and cling.
2. build options - 90 %. For example, gsl_shared and xml.
   Options can be unclear with no naming convention. We propose to introduce the concept of a ROOT subpackage.



4. Layering ROOT

Design goals:

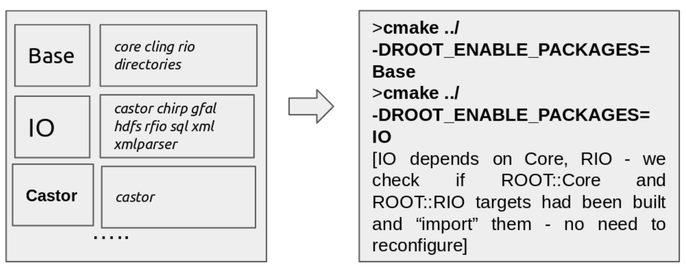
1. Arrange existing ROOT components into layers. For instance, core→mathcore→mathmore.
2. Allow each layer (subpackage) can be enabled/disabed.

This is implemented an overload of CMake add_subdirectory with iteration loop through ROOTPackageMap.cmake (similar to a package database). Similar implementation also exists in LLVM project:

- add_llvm_subdirectory(x)
- add_clang_subdirectory(x)
- add_cling_subdirectory(x)

We propose a new way of organisation of ROOT build options:
→ smallest granularity =

= 1 ROOT library is one 1 subpackage



We can add in the same way external projects using ROOT_EXTERNAL_PACKAGES, by specifiing the path for the source code of the project.

5. ROOT dependency management

We propose a different way of managing dependencies and options:

```
add_root_library(DCache
                 HEADERS TDCacheFile.h
                 SOURCES src/TDCacheFile.cxx
                 LIBRARIES ${DCAP_LIBRARIES}
                 DEPENDENCIES Core Net RIO)
```

→

```
add_root_library(DCache
                 HEADERS TDCacheFile.h
                 SOURCES src/TDCacheFile.cxx
                 EXTERNAL_DEPENDENCIES dcache
                 DEPENDENCIES Core Net RIO)
```

```
if(dcache)
  add_subdirectory(dcache)
endif()
```

→

```
add_subdirectory(dcache)
```

6. Future work: modern dependency CMake target management

We want to generate the "modern style" CMake targets for any external dependency, that could be used iteratively afterwords:

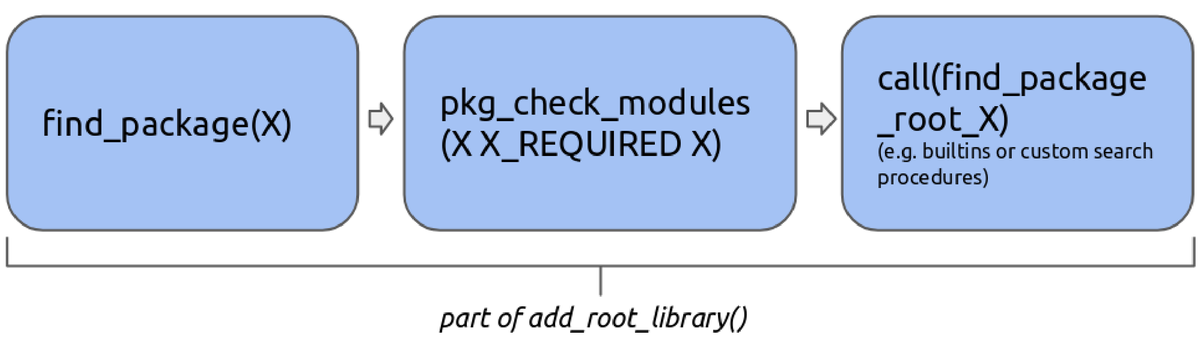
```
if(X_FOUND AND NOT TARGET X::X)
  add_library(X::X UNKNOWN IMPORTED)
  set_target_properties(X::X PROPERTIES
    IMPORTED_LOCATION "${X_LIBRARY}"
    INTERFACE_COMPILE_OPTIONS "${X_COMPILE_OPTIONS}"
    INTERFACE_INCLUDE_DIRECTORIES "${X_INCLUDE_DIR}"
  )
endif()
```

All targets are exported/imported in/from ROOTDependenciesTargets.cmake.

7. ROOT dependencies: find_package

1. add_root_library(DEPENDENCIES..) changes the way how ROOT dependencies found:

2. add_root_library(EXTERNAL_DEPENDENCIES..) changes the way how ROOT external dependencies found:



8. Use cases

- We have already build ROOT from sources, **we want to extend functionality of ROOT**.
- We touched one of ROOT component CMakeLists.txt and **we want to rebuild only this component, without reconfiguring all ROOT**.

9. Interaction with PM

- **root-get**: it will enable the ROOT-aware package management with the root-get prototype.
- **Fedora**: it will be more easier to generate more granular ROOT packages.
- **Conda**: It will support a root-minimal package to further improve install times.