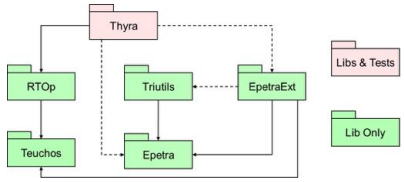
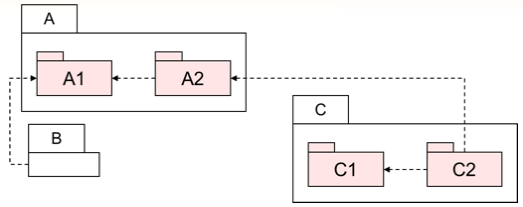
TriBITS Modernization





Roscoe A. Bartlett
Department 1424
Software Engineering and Research

December 2, 2021

Trilinos Users Group Meeting, Developers Day

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2021-15212 PE

TriBITS and CMake Background

TriBITS History

- **2007**: A partial initial CMake build system for Trilinos started by Tim Shead & Danny Dunlavy.
- **2008**: Ross takes over CMake build system and creates package-based architecture and wrappers for raw CMake.
- **2011**: TriBITS system factored out of Trilinos into independent git repo to support larger, more complex CASL VERA project.
- **2014**: Primary TriBITS development is complete and TriBITS is put out on GitHub.

CMake Developments: (Source: **Professional CMake: 10th edition**)

- **2014**: CMake 3.1: First usable target-centric modern CMake.
- **2016**: CMake 3.7: More realistic for using modern target-centric CMake.
- **2018** (Mar): CMake 3.11: Modern target-centric dependency management for aggregate projects well supported.
- **2018** (Nov): CMake 3.13: Link options and compiler options de-duplication.

Trilinos CMake Minimum versions:

- **2008**: CMake 2.6
- **2011**: CMake 2.7
- **2014**: CMake 2.8.11
- **2018**: CMake 3.10
- **2021**: CMake 3.17.0

TriBITS implemented a scalable architecture for CMake projects 6 years before than was possible with raw CMake in CMake 3.1 and 8 years before it was really well supported in CMake 3.7. But, TriBITS is now standing in the way of adopting some modern CMake features.

Accelerated CMake Adoption and Developments

- There has been significant growth in CMake adoption, maturation and feature development in recent years. (CMake is now most popular build system for C++ code in the world)
- Many features/workarounds added to TriBITS in early years have been resolved in native CMake.
- Many now-redundant TriBITS features are inconsistent and/or inferior to native CMake solutions and idioms. Examples:
  - Target-centric builds (compiler options, link options, include dirs., etc.)
  - Fortran/C name mangling (FortranCInterface.cmake)
  - Standard install locations (GNUInstallDirs.cmake)
  - RPATH Handling
  - Handling of deprecated code (GenerateExportHeader.cmake)
  - …

- However, areas where (nearly) everyone seems to agree native CMake is lacking where a (reduced) TriBITS provides value:
  - Package architecture for CMake projects (e.g. VTK Modules)
  - Helper functions for defining and managing tests (e.g. MPI, allocating tests to GPUs, limiting tests based on MPI ranks and threads, etc.)

Componentized CMake-based Projects Approaches

CMake, CTest, and CDash are great, but raw usage does not scale very well to large projects and multiple repositories and teams!

- **Multiple CMake projects:**
  - Manual builds and linking through <Package>Config.cmake files (e.g. Albany & Trilinos)
  - CMake ExternalProject: Provided as standard CMake module (only raw CMake)
  - CApp: Lightweight CMake package manager by Dan Ibanez (only raw CMake and git)
  - Google Catkin : Used for the Google Robotics Operating System (ROS) project (requires Python)
  - Spack: Source builds/package manager used in ECP project and E4S (requires Python)
  - Likely many others as well …
- **Single CMake project:**
  - Kitware VTK Modules:
  - TriBITS:
    + Support multiple repos
    + Core functionality depends only on CMake 3.17+

Goal => Develop CMake packages that allow building in single CMake projects or in separate CMake projects in arbitrary sets depending on need.

Refactoring TriBITS CMake Build System to Modern CMake

Goals for updated Trilinos (TriBITS) build system[ζ]:

- Allow packages to use raw CMake to define targets for libraries, executables, etc. according to the proposed standard (e.g. provide <Package>::<lib> and <Package>::all_libs)
- Use tribits_add_test(), tribits_add_advanced_test() and even tribits_add_executable_and_test() to define tests.
- Use TriBITS external package/TPL system to find external packages (i.e. combine requirements from all enabled packages and call find_package() just once for each external package/TPL).
- TriBITS refactoring should allow existing packages to keep working without out modification.
- The decision to use tribits_add_library() and tribits_add_executable() and other optional TriBITS convenience functions and can be made on a package-by-package basis.

[ζ] Meeting between Jeremy W., Keita T. Ross B., Dec 2020, see TriBITS #342

Constraints/Requirements:

- **Not break any existing CMakeLists.txt files** in existing TriBITS projects including Trilinos, Drekar, Charon2, DataTransferKit, MPACT, CTF, VERA, ...
- **Not break existing user Trilinos and other configure scripts**.
- Allow trimming down TriBITS and using native CMake in TriBITS projects to occur **incrementally**.
- Allow refactoring of existing Trilinos packages to use raw CMake targets and build independently from Trilinos to occur **incrementally**.

Generalized Handling of Internal and External Packages

Refactoring of TriBITS to modern CMake targets to deal with internal and external packages consistently

- <Package>::<lib>: Single (library) target
  - Self-contained modern CMake target which contains include directories, compiler options, link options, etc.
  - Standard library target for internal (TriBITS) packages built within the CMake project, **or**
  - IMPORTED target for external packages defined in <Package>Config.cmake file, **or**
  - IMPORTED or INTERFACE target generated from a TriBITS TPL specification.
- <Package>::all_libs: INTEFACE (IMPORTED) library target for all libraries for internal or external <Package>
  - From internal packages, or from external <Package>Config.cmake file, or from generated from a TriBITS TPL specification

Example of Supporting Use Cases:

- Allow an existing TriBITS project to be built and installed in smaller CMake projects. Examples:
  - **Build and install Kokkos, Kokkos-Kernel, and SEACAS as independent CMake projects** and pull them in as KokkosConfig.cmake, KokkosKernelsConfig.cmake, and SEACAS<Subpackage>Config.cmake files and build the rest of Trilinos.
  - **Build and install Tpetra and Belos as independent CMake projects** pulling in pre-installed Kokkos and KokkosKernels.
- **Allow any TriBITS package to be pulled out and built as an independent CMake project** building against pre-installed upstream packages as external packages.

TriBITS Build System Ecosystem Modernization Plan

Plan for FY22 and beyond:

- **Refactor TriBITS to use modern target-centric CMake** for handling include directories, compiler options, link options, etc. and change to using standard targets <Package>::<lib> and <Package>::all_libs for internal package and external packages/TPLs.
- **Initially avoid breaks in backwards compatibility** for developers and users as much as possible.
- **Then, incrementally refactor TriBITS, Trilinos, and other projects** that use TriBITS to switch to newer native CMake features and idioms and trim down TriBITS to it smallest possible components. (This may break backwards compatibility in some cases, but will be done in small increments that will be easier to absorb).

Roles in this plan:

- Ross Bartlett:
  - Leads the main refactoring of TriBITS.
  - Leads initial deployments into Trilinos, Kokkos, KokkosKernels, SEACAS, Drekar, Charon2, and other impacted codes, etc. (should be minimal effort).
- NextGen Analytics:
  - Supports refactor of TriBITS and deployments into Trilinos and other projects.
  - Does majority of follow-on refactoring work in TriBITS, Trilinos and other projects to switch to newer native CMake features after initial deployment of major TriBITS refactoring.

Current progress

Internal TriBITS package targets have been partially converted over to modern CMake (TriBITS PR #424)

- TriBITS Build Reference Guide Documentation:
  - 8.6 Using the installed software in downstream CMake projects
  - 8.7 Using packages from the build tree in downstream CMake projects
- Example projects:
  - TribitsSimpleExampleApp
  - TribitsExampleApp

From `TribitsSimpleExampleApp/CMakeLists.txt`:

```
find_package(TribitsExProj REQUIRED
  COMPONENTS SimpleCxx MixedLang WithSubpackages)
…
add_executable(app app.cpp)
target_link_libraries(app PRIVATE TribitsExProj::all_selected_libs)
```

Currently refactoring handling of external packages/TPLs to use <tplName>Config.cmake files and <tplName>::all_libs targets and linkages to upstream dependencies.

Gluing Together Various External Packages

Challenge: Provide standard self-contained modern CMake targets <tplName>::all_libs for all external packages/TPLs specified in different ways:

1. List of include directories, libraries, link options, etc., through TriBITS TPL_<tplName>_INCLUDE_DIRS and TPL_<tplName>_LIBRARIES variables?

   => **Solution**: Automatically handled by refactored TriBITS!

2. Pre-installed upstream TriBITS package?

   => **Solution:** Automatically handled by refactored TriBITS!

3. Using find_package(<tplName>) to find external standard (or non-standard) Find<tplName>.cmake module or <tplName>Config.cmake file provided by an external package/TPL?

   => **Solution:** Create custom FindTPL<tplName>.cmake files that call find_package(<tplName>) and construct self-contained <tplName>::all_libs target. (See How to use find_package() for a TriBITS TPL in the TriBITS Users Guide and Reference.)

NOTE: The need to create custom FindTPL<tplName>.cmake files where (partial) modern CMake is used with Find<tplName>.cmake modules or <tplName>Config.cmake files to provide IMPORTED targets may be where a majority of work of developers will be expended in transitioning to modern CMake ☹

External Packages/TPLs and Modern CMake Targets

Challenge: Support existing TriBITS TPL specifications through:

```
-D <tplName>_INCLUDE_DIRS="<Idir1>;<Idir2>;…"
-D <tplName>_LIBRARY_NAMES="<name1>;<name2>;…"
-D <tplName>_LIBRARY_DIRS="<Ldir1>;<Ldir2>;…"
```

(which are resolved using find_() calls) or explicitly through:

```
-D TPL_<tplName>_INCLUDE_DIRS="<Idir1>;<Idir2>;…"
-D TPL_<tplName>_LIBRARIES="/full/path/to/lib<libname1>.so;-L<dir2>;-l<libname2>;<libname3>;…"
```

and create **<tplName>Config.cmake** files with modern CMake IMPORTED library targets and linked targets with upstream external packages/TPLs. These files are installed and loaded from the build directory:

```
<buildDir>/external_packages/<tplName>/<tplName>Config.cmake
```

and install directory under:

```
<installDir>/lib/external_packages/<tplName>/<tplName>Config.cmake
```

- **NOTE:** These <tplName>Config.cmake files do **NOT** get found by default by find_package(<tplName)!
- **NOTE:** Arbitrary link options cannot be translated into IMPORTED library targets as of CMake 3.22 and maintain the needed ordering of the link line. Example: -Wl,-Bstatic –l<libname> cannot be handled!

Primary FY22 Goals

- **Complete refactoring to internal usage of modern CMake targets and for treating internal and external packages uniformly**
  - Clean linking against <Package>::<libname> and <Package>::all_libs for internal and external packages (and strip out old TriBITS logic)
  - Uniform dependency handling of internal and external packages (including between external packages)
- **Building and installing upstream selected packages independently:**
  - Prebuild and install Kokkos and KokkosKernels and build remaining Trilinos package against these
  - Prebuild and install SEACAS (against pre-installed Kokkos and Zoltan) and build remaining Trilinos packages against these.
- **TriBITS Meta packages**:
  - **ShyLU**: Where Trilinos_ENABLE_ShyLU=[ON|OFF] and ShyLU_ENABLE_TESTS=[ON|OFF] behaves like it is a package and ShyLU_Node and ShyLU_DD are its subpackages
  - **TrilinosLinearSolvers**: Trilinos_ENABLE_TrilinosLinearSolvers=[ON|OFF] enable (or disables) all Trilinos linear solver and preconditioner packages

To keep track of progress:

- TriBITS Refactor Kanban Board (Project Board #2)
- EPIC: TriBITS Modernization Plan (TriBITS #367)
- Bi-weekly meeting TriBITS Modernization Meetings (starting in Jan 2022)?
- SEMS Review meetings

Questions and Comments?