

CMake TPL Support

Using Find_package

Pros/Cons

Will Dicharry, Brent
Perschbacher

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



1. CMake Compatibility

- ◆ We currently don't use standard interfaces defined by Cmake.
 - ◆ `find_package(<name> [REQUIRED] [COMPONENTS] component...)`
 - ◆ `Foo_INCLUDE_DIRS, Foo_LIBRARIES, Foo_FOUND, ...`
- ◆ We typically don't use existing Find*.cmake logic.
 - ◆ *Exceptions: CUDA, Qt*

2. TPL Components

- ◆ Some TPLs have multiple components.
 - ◆ **Boost**
- ◆ Packages aren't always interested in all provided components.
 - ◆ **Boost, HDF5**

3. TPL versions

- ◆ Some packages need a different version of a TPL than another package
 - ◆ **SuperLU**

4. TPL dependencies

- ◆ TPLs depend on one another.
 - ◆ **ExodusII, NetCDF4, HDF5**
- ◆ Dependencies can be required or optional.



CMake Compatibility

- **CMake is widely used.**
- **Client applications/libraries expect find_package interface.**
- **Some Find Modules are robust.**
 - Boost
 - Qt
 - CUDA
 - Trilinos
- **Some aren't.**
 - MPI
- **Can override behavior of FindFoo.cmake at the project level by setting CMAKE_MODULE_PATH.**
 - Not locked into the system implementation.
- **Utilizing interfaces that developers expect improves interoperability.**



TPL Components/Versions

- **Some packages depend only on certain components of TPLs.**
- **Some subpackages depend only on certain components of TPLs.**
- **find_package provides a standard interface for handling this.**
 - find_package(Foo COMPONENTS bar baz)
 - Foo_LIBRARIES, Foo_bar_LIBRARY, Foo_baz_LIBRARY
- **TPL dependency isn't all or nothing.**
 - Packages can choose a minimal set of dependencies.
 - Language bindings.
- **Specific versions of TPLs are sometimes required.**
 - find_package(Foo 4.3.7 [EXACT])
- **Interface is distinct from Trilinos package architecture dependencies.**
 - Inconsistent



TPL Dependencies

- **Some TPLs depend on other TPLs.**
- **Current options:**
 - TPL1 depends on TPL2 -> put TPL2 libraries in TPL1.
 - Client packages enable both manually.
- **Solution: Find_package can call find_package internally.**
 - Similar to option 1.
 - Duplicates possible.
 - Handles REQUIRED/COMPONENTS issues nicely.
 - What about optional dependencies?
- **Probably will have issues crossing between system/custom find modules.**



TPL Support Options

- **Determine how we will work with current dependencies structure:**
 1. **Packages call find_package?**
 - Dependency decisions are made by packages.
 - How do we handle conditional compilation?
 - Would need to change package code.
 2. **Done from package architecture?**
 - Global dependency decisions based on complete package set.
 - How to handle REQUIRED/COMPONENTS?
 - Transparent to packages.



Conclusion

- **Any other requirements we're missing**
 - Cmake compatibility
 - TPL components
 - TPL versions
 - TPL dependencies
- **Who would like to be the involved in the design?**
- **Anyone opposed to moving to find_package?**

- **Automation of TPL build and install.**
 - The new Jenkins test setup provides a build farm with a variety of platforms (~20 machines currently).
 - Jenkins can dynamically distribute testing loads.
 - How do we make sure the right TPLs are available?
 - Options:
 - ◆ **TPL extra repository**
 - ◆ **Find modules that download**

- **Package dependencies using find_package.**
 - Does it make sense to use find_package for Trilinos package dependencies?
 - This might assist with some of the subpackage issues.
 - ◆ **Find_package(Thyra COMPONENTS Core EpetraAdapters)**