SAND2016-10688C

How to build Trilinos

Mark Hoemmen & Alicia Klinvex
Sandia National Laboratories
24 Oct 2016

Sandia is a multiprogram 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.

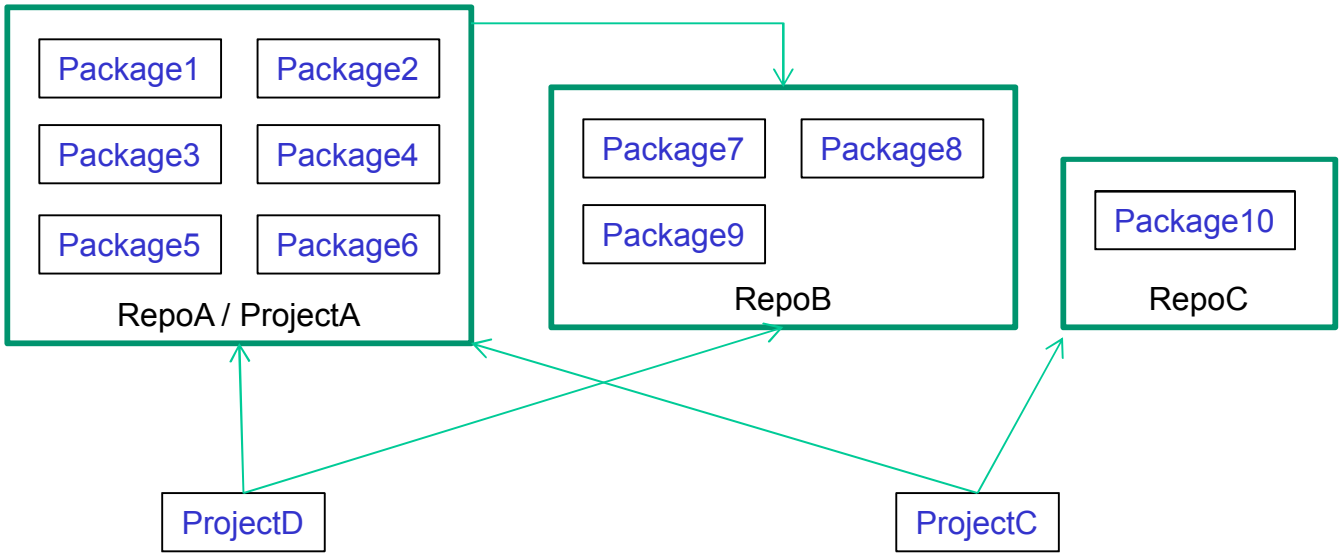
Sandia National Laboratories

Trilinos' package management

TriBITS: Trilinos/Tribal Build, Integrate, Test System

- Based on CMake, CTest, & CDash (Kitware open-source toolset)
  - Developed during Trilinos' move to CMake
  - Later extended for use in CASL projects (e.g., VERA) & SCALE
- Partitions a project into packages
  - Common CMake build & test infrastructure across packages
  - Handles dependencies between packages
- Integrated support for MPI, CUDA, & third-party libraries (TPLs)
- Multi-repository development
  - Can depend on packages in external repositories
  - Handy for mixing open-source & closed-source packages
- Test driver (tied into CDash)
  - Runs nightly & continuous integration
  - Helps target which package caused errors
- Pre-push synchronous continuous integration testing
  - Developers must use Python checkin-test.py script to push
  - The script enables dependent packages, & builds & runs tests
  - Also automates asynchronous continuous integration tests
- Lots of other features!

TriBITS: Meta Project, Repository, Packages



Current TriBITS features:

- Flexibly aggregate packages from different repositories into big projects
- Can use TriBITS in stand-alone projects (independent of Trilinos)
- In use by CASL VERA software, and several other CASL-related software packages
- Tool to manage multiple git repositories

Future changes/additions to TriBITS

- Combine concepts of TPLs & packages to allow flexible configuration & building
- TribitsExampleProject
- Trilinos-independent TriBITS documentation
- Provide open access to TribitsExampleProject and therefore TriBITS

Sandia National Laboratories

Building your application with Trilinos

Building your application with Trilinos

If you are using Makefiles:

- Makefile.export system

If you are using CMake:

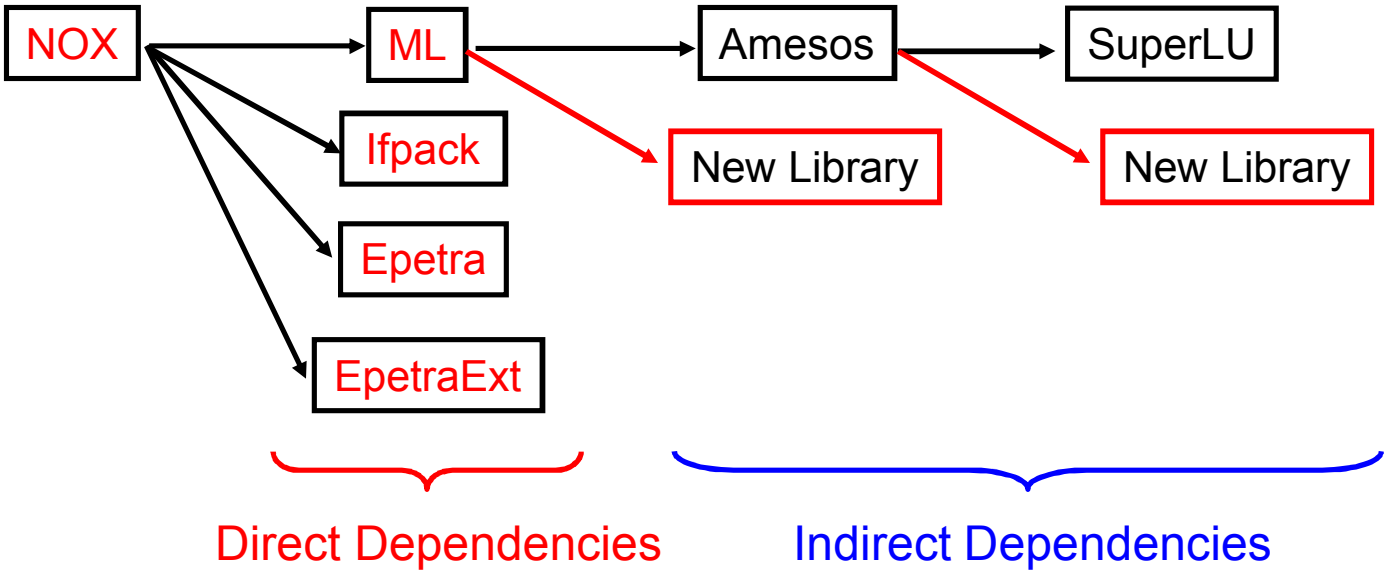
- CMake FIND_PACKAGE

Don’t write Makefiles that use Trilinos by hand!

- Which libraries? Link order matters!
  - -lnoxepetra -lnox –lepetra –lteuchos –lblas –llapack
  - Optional package dependencies affect required libraries
- Using the same compilers that Trilinos used
  - g++ or icc or icpc or …?
  - mpiCC or mpCC or mpicxx or … ?
- Using the same libraries that Trilinos used
  - Using Intel’s MKL requires a web tool to get the link line right
  - Trilinos remembers this so you don’t have to
- Consistent build options and package defines:
  - g++ -g –O3 –D HAVE_MPI –D _STL_CHECKED
- You don’t have to figure any of this out! Trilinos does it for you!
  - Please don’t try to guess and write a Makefile by hand!
  - This leads to trouble later on, which I’ve helped debug.

Why is this hard? Why doesn't "-ltrilinos" work?

- Trilinos has LOTS of packages
- Top-level packages might get new package dependencies indirectly, without knowing it:



Better to let Trilinos tell you what libraries you need!
It already does the work for you.

Using the Makefile.export system

```
#
# A Makefile that your application can use if you want to build with Epetra.
#
# You must first set the TRILINOS_INSTALL_DIR variable.

# Include the Trilinos export Makefile for the Epetra package.
# You may do this per package or for all of Trilinos.
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.Epetra

# Add the Trilinos installation directory to the library and header search paths.
LIB_PATH = $(TRILINOS_INSTALL_DIR)/lib
INCLUDE_PATH = $(TRILINOS_INSTALL_DIR)/include $(CLIENT_EXTRA_INCLUDES)

# Set the C++ compiler and flags to those specified in the export Makefile.
# This ensures your application is built with the same compiler and flags
# with which Trilinos was built.
CXX = $(EPETRA_CXX_COMPILER)
CXXFLAGS = $(EPETRA_CXX_FLAGS)

# Add the Trilinos libraries, search path, and rpath to the
# linker command line arguments
LIBS = $(CLIENT_EXTRA_LIBS) $(SHARED_LIB_RPATH_COMMAND) \
  $(EPETRA_LIBRARIES) \
  $(EPETRA_TPL_LIBRARIES) \
  $(EPETRA_EXTRA_LD_FLAGS)

#
# Rules for building executables and objects.
#
%.exe : %.o $(EXTRA_OBJS)
        $(CXX) -o $@ $(LDFLAGS) $(CXXFLAGS) $< $(EXTRA_OBJS) -L$(LIB_PATH) $(LIBS)

%.o : %.cpp
        $(CXX) -c -o $@ $(CXXFLAGS) -I$(INCLUDE_PATH) $(EPETRA_TPL_INCLUDES) $<
```

Sandia National Laboratories

Using CMake to build with Trilinos

- CMake: Cross-platform build system
  - Similar function as the GNU Autotools
- Building Trilinos requires CMake
- You don't have to use CMake to use Trilinos
- But if you do: FIND_PACKAGE(Trilinos …)
- I find this much easier than writing Makefiles

Documentation, downloading, & building Trilinos

How do I learn more?

- Documentation:
  - Per-package documentation: http://trilinos.org/packages/
  - Other material on Trilinos website: http://trilinos.org/
  - Trilinos Wiki with many runnable examples: https://code.google.com/p/trilinos/wiki/
- E-mail lists: http://trilinos.org/mail_lists.html
- Annual user meetings and other tutorials:
  - Trilinos User Group (TUG) meeting and tutorial
    - Here we are! Last week of October at SNL / NM
  - European TUG meetings (once yearly)
  - Talks available for download (slides &/or video):
    - http://trilinos.org/community/events/

How do I get Trilinos?

- Current release (12.8) available for download
  - http://trilinos.org/download/
  - Source tarball with sample build scripts
- Trilinos coming soon to Github
  - Development version, updated ~ nightly
- Cray packages recent releases of Trilinos
  - http://www.nersc.gov/users/software/programming-libraries/math-libraries/trilinos/
  - `$ module load trilinos`
  - Recommended for best performance on Cray machines
- Most packages BSD license; a few are LGPL

How do I build Trilinos?

- Need C and C++ compiler and the following tools:
  - CMake (version >= 2.8.11), BLAS, & LAPACK
- Optional software:
  - MPI (for distributed-memory parallel computation)
  - Many other third-party libraries
- You may need to write a short configure script
  - Sample configure scripts in sampleScripts/
  - Find one closest to your software setup, & tweak it
- Build sequence looks like GNU Autotools
  1. Invoke your configure script, that invokes CMake
  2. `make -j<N> && make -j<N> install`
- Documentation:
  - TrilinosBuildQuickRef.* in Trilinos source directory
  - https://trilinos.org/docs/files/TrilinosBuildReference.html
  - I will cover this, given audience interest

Hands-on tutorial

- 2 ways to use Trilinos: WebTrilinos, or install it
- WebTrilinos (build & run in your browser!)
  - Need username & password (will give these out later)
  - http://trilinos.csbsju.edu/WebTrilinosMPI-shared-12.2/c++/index.html
- Tutorial examples
  - Examples from many Trilinos packages: https://github.com/jwillenbring/hands_on_tutorial/tree/master/wiki
  - Tpetra tutorial: https://trilinos.org/docs/dev/packages/tpetra/doc/html/index.html
  - Kokkos tutorial: https://github.com/kokkos/kokkos-tutorials
  - Epetra tutorial: https://trilinos.org/docs/dev/packages/epetra/doc/html/index.html
  - For other Trilinos packages: see trilinos.org -> Packages

Other options to use Trilinos

- Virtual machine
  - Install Docker, download VM file, and run it
  - We won't cover this today
- Build Trilinos yourself on your computer
  - We may cover this later, depending on your interest
  - Prerequisites:
    - C++ compiler, CMake version >= 2.8.11, BLAS & LAPACK, (MPI)
    - Download Trilinos: trilinos.org -> Download (tarball)
  - Find a configuration script suitable for your computer
    - Trilinos/sampleScripts/
  - Modify the script if necessary, & use it to run CMake
  - make –jN, make –jN install
  - Build your programs against Trilinos
    - Use CMake with FIND_PACKAGE(Trilinos …), or
    - Use Make with Trilinos Makefile.export system

Questions?