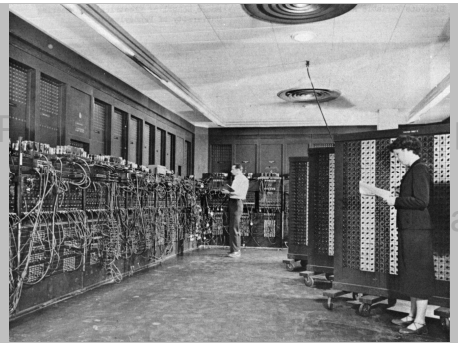
SAND2017-11532C



How to configure and build Trilinos

Mark Hoemmen, 23 Oct 2017

SAND2017-????? C (UUR)

Outline

- Why is it so hard? Rather:
  - How do Trilinos' culture & users affect configuration & building?
  - What features does Trilinos support that makes building nontrivial?
- How do I build Trilinos?
  - What programs & libraries will I need?
  - How do I configure (set options & prepare to build)?
- How do I link my application against Trilinos?
  - Please don't just paste in the libraries on your link line!
  - For Make-based build system: Makefile.export.*
  - For CMake-based build system: FIND_PACKAGE

Trilinos: Confederacy, not project

Sandia National Laboratories

- Many packages, many projects
- Packages may depend on
  - Each other: e.g., Ifpack on Epetra
  - Third-party libraries (TPLs): e.g., BLAS, Boost
  - Compilers: CUDA, MPI
- Dependencies: optional or required
- Packages may live in different repos
- Common build & test infrastructure
  - TriBITS: A project in itself, used elsewhere
  - Motivated by CASL VERA
    - Handle software licensing & access control issues

"It's not a big truck. It's a series of tubes" – Sen. Ted Stevens

Why is Trilinos a confederacy?

- "Three pearls" (Τρία μαργαριτάρια)
  - Aztec (iterative linear solvers)
  - ML (algebraic multigrid)
  - Zoltan (graph partitioning, load balancing)
- Share only repository, build, & test
- Original concept: Optional interfaces
  - AztecOO (Epetra-Aztec), Isorropia (Epetra-Zoltan), ML (Epetra)
  - Fully stand-alone; take (don't use) Epetra
- Later evolution moved away from this
  - Stratimikos: Needs Teuchos, wants Epetra
  - MueLu, Panzer: Long chain of required deps
  - Kokkos as common programming model

White pearl necklace
(see Notes for attribution)

What do I need to build Trilinos?

Sandia National Laboratories

- Minimum
  - C & C++ compiler
    - Many packages need C++11 / C99
    - We test with GCC, Intel, Clang, NVCC, & XL (IBM)
  - BLAS & LAPACK libraries
  - CMake >= 2.8.11 (prefer >= 3.3.2)
- Optional (required for some packages)
  - MPI, CUDA, Fortran
  - Many third-party libraries (see Trilinos/TPLsList.cmake for full set)
- We recommend
  - Linux, *nix, or MacOS X (Windows experience varies)
  - BLAS, LAPACK, MPI (all ABI-compatible)
  - Whatever else the packages you want require

Setting options & preparing build

Sandia National Laboratories

- Setting options & preparing build == “configuring”
  - Trilinos uses CMake for this
  - Compare to running “./configure …” with GNU Autotools
  - Users often turn this CMake invocation into a script
  - We call this the “do-configure” script, & will show examples
- Trilinos developers also use “check-in test script”
  - Python script that drives CMake, CTest, & git
  - Automatically enables packages affected by your changes
  - Lets Trilinos developers test multiple builds with different options
  - Can do asynchronous remote test & push

Hints for configuring Trilinos

Sandia National Laboratories

- Say as little as possible
  - Trilinos can often detect compilers, BLAS, & LAPACK
  - Best used with a “module” system
- “As little as possible” example: MPI
  - `TPL_ENABLE_MPI:BOOL=ON` # may be enough!
  - `MPI_BASE_DIR:FILEPATH="…"` # if not in $PATH
- What’s with `:BOOL=ON` vs. `=ON` ?
  - `:${TYPE}` lets you specify the option’s type
  - Examples: `BOOL`, `STRING`, `FILEPATH`
  - It’s optional, e.g., `Trilinos_ENABLE_OpenMP=ON`
- What’s with `ON` / `OFF`, `TRUE` / `FALSE`, etc.?
  - CMake lets you spell “true” & “false” in different ways

How do I…

- Set the install directory?
  - `CMAKE_INSTALL_PREFIX=${INSTALL_PATH}`
- Set debug or release build?
  - `CMAKE_BUILD_TYPE=DEBUG (or RELEASE)`
- Set C++ compiler flags? `CMAKE_CXX_FLAGS="…"`
  - No need to add `-g` for debug. Release adds `-O3`.
- Enable C++11 support?
  - Usually, automatically detected & enabled by default
  - If not, set `Trilinos_CXX11_FLAGS` (not `CMAKE_CXX_FLAGS`)
- Set whether to use dynamic shared libraries?
  - `BUILD_SHARED_LIBS=ON (or OFF)`

How do I…

- Tell CMake where to find MPI?
  - Remember: Say as little as possible
  - `TPL_ENABLE_MPI=ON`
  - `MPI_BASE_DIR=${PATH_TO_MPI_INSTALL}`
  - Can add nondefault `mpiexec` name, options, etc.
- Set compiler paths? (non-MPI, a.k.a. “serial” build)
  - `CMAKE_CXX_COMPILER=${PATH_TO_CXX_COMPILER}`
  - Analogous for C & Fortran compilers
- Enable OpenMP? `Trilinos_ENABLE_OpenMP=ON`
  - Usually enough; no need to specify compiler flags
- Use CUDA? See Trilinos/packages/tpetra/doc/FAQ.txt

Package-related options

- Enable a specific package (& its required deps)?
  - `Trilinos_ENABLE_${PKG}=ON`
- To see list of all available packages
  - CMake output: “Final set of [non-]enabled packages: ”
  - For TPLs: “Final set of [non-]enabled TPLs: ”
  - Or, read Trilinos/{PackagesList, TPLsList}.cmake
- Enable all (optional, fwd dep) packages?
  - `Trilinos_ENABLE_ALL_PACKAGES=ON`
  - `Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES=ON`
  - `Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES=ON`

How do I find out more?

Sandia National Laboratories

- Read Trilinos/INSTALL.rst (it's short & good!)
- Read examples in Trilinos/sampleScripts
  - Age & quality vary
  - They don't always follow "as simple as possible"
  - Beware "cargo cult configuration"
- trilinos.org/docs/files/TrilinosBuildReference.html
- Ask for help on GitHub or the e-mail list

Building Trilinos with Ninja

Sandia National Laboratories

- Ninja: alternative to Make; faster, more parallel builds
- CMake can generate Ninja build files
  - Add `-G Ninja` to cmake command-line arguments
- Some restrictions
  - Needs patch to build Fortran, or turn off Fortran in Trilinos:
    - `Trilinos_ENABLE_Fortran:BOOL=OFF`
  - Does not yet work with Trilinos' check-in test script
  - Must build from top level of build directory; can't change into subdirectories & build there
  - This mainly only affects Trilinos developers

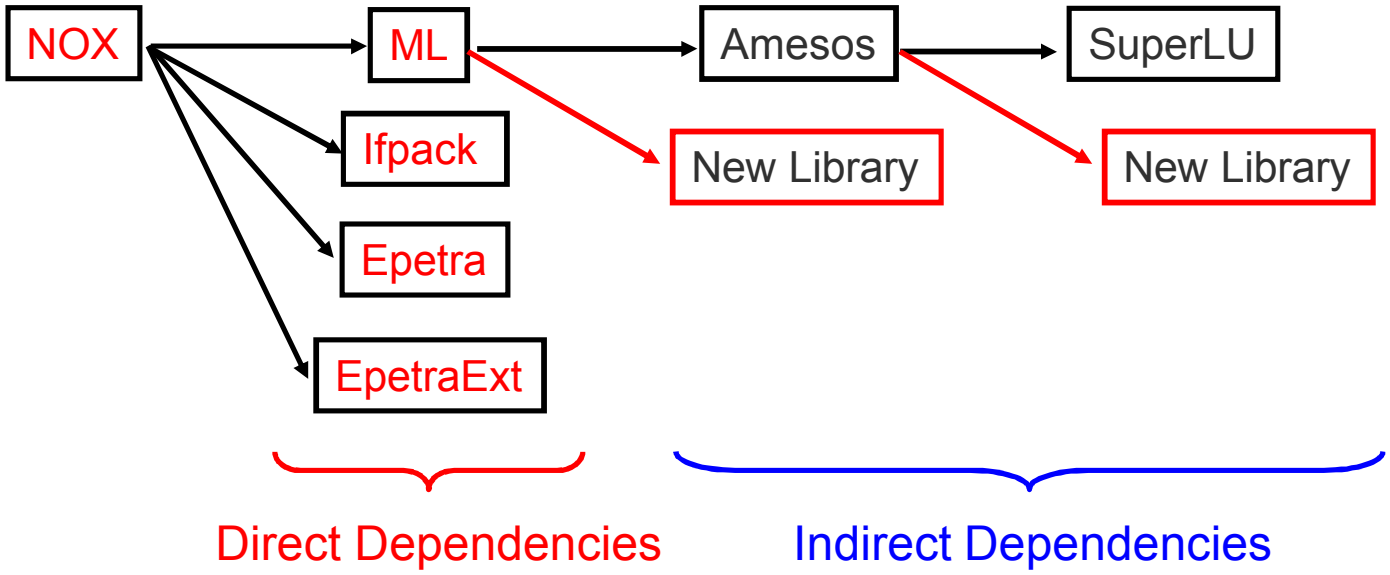
Building your app with Trilinos

Sandia National Laboratories

- 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 doesn't "-ltrilinos" work?

- Trilinos has LOTS of packages
- Top-level packages might get new package dependencies indirectly, without knowing it
- Build system is extensible; users can add new packages



Building your app with Trilinos

Sandia National Laboratories

If you are using Make:

- Makefile.export system

If you are using CMake:

- CMake FIND_PACKAGE

Example Makefile for your app

```
# You must first set the TRILINOS_INSTALL_DIR variable.

# Include Trilinos-related variables in your project.  If you only want
# 1 package, replace “Trilinos” with the package’s name, e.g., “Epetra”.
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.Trilinos

# 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)

# Use the same C++ compiler, flags, & libraries that Trilinos uses.
CXX = $(Trilinos_CXX_COMPILER)
CXXFLAGS = $(Trilinos_CXX_FLAGS)
LIBS = $(CLIENT_EXTRA_LIBS) $(SHARED_LIB_RPATH_COMMAND) \
  $(Trilinos_LIBRARIES) \
  $(Trilinos_TPL_LIBRARIES) \
  $(Trilinos_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) $<
```

Using CMake to build with Trilinos

Sandia National Laboratories

- 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 …)
- Like Makefile.export system, this pulls variables into your CMake environment

Example CMakeLists.txt for your app

Sandia National Laboratories

```
# Run “cmake -DTrilinos_PREFIX=${TRILINOS_PATH}” to configure.
SET(CMAKE_PREFIX_PATH ${Trilinos_PREFIX} ${CMAKE_PREFIX_PATH})
FIND_PACKAGE(Trilinos REQUIRED)

# Show some of the CMake variables that finding Trilinos defines.
MESSAGE("\nFound Trilinos!  Here are the details: ")
MESSAGE("   Trilinos_DIR = ${Trilinos_DIR}")
MESSAGE("   Trilinos_VERSION = ${Trilinos_VERSION}")
MESSAGE("   Trilinos_PACKAGE_LIST = ${Trilinos_PACKAGE_LIST}")
MESSAGE("   Trilinos_LIBRARIES = ${Trilinos_LIBRARIES}")
MESSAGE("   Trilinos_INCLUDE_DIRS = ${Trilinos_INCLUDE_DIRS}")
MESSAGE("   Trilinos_LIBRARY_DIRS = ${Trilinos_LIBRARY_DIRS}")
MESSAGE("   Trilinos_TPL_LIST = ${Trilinos_TPL_LIST}")
MESSAGE("   Trilinos_TPL_INCLUDE_DIRS = ${Trilinos_TPL_INCLUDE_DIRS}")
MESSAGE("   Trilinos_TPL_LIBRARIES = ${Trilinos_TPL_LIBRARIES}")
MESSAGE("   Trilinos_TPL_LIBRARY_DIRS = ${Trilinos_TPL_LIBRARY_DIRS}")
MESSAGE("   Trilinos_BUILD_SHARED_LIBS = ${Trilinos_BUILD_SHARED_LIBS}")

# Use the same compilers and flags as Trilinos does.  (No-MPI example.)
SET(CMAKE_CXX_COMPILER ${Trilinos_CXX_COMPILER})
SET(CMAKE_C_COMPILER ${Trilinos_C_COMPILER})
SET(CMAKE_Fortran_COMPILER ${Trilinos_Fortran_COMPILER})
SET(CMAKE_CXX_FLAGS "${Trilinos_CXX_COMPILER_FLAGS} ${CMAKE_CXX_FLAGS}")
SET(CMAKE_C_FLAGS "${Trilinos_C_COMPILER_FLAGS} ${CMAKE_C_FLAGS}")
SET(CMAKE_Fortran_FLAGS "${Trilinos_Fortran_COMPILER_FLAGS} ${CMAKE_Fortran_FLAGS}")

PROJECT(MyApp)
INCLUDE_DIRECTORIES(${Trilinos_INCLUDE_DIRS} ${Trilinos_TPL_INCLUDE_DIRS})
LINK_DIRECTORIES(${Trilinos_LIBRARY_DIRS} ${Trilinos_TPL_LIBRARY_DIRS})
ADD_LIBRARY(myappLib src_file.cpp src_file.hpp)
ADD_EXECUTABLE(MyApp.exe main_file.cpp)
TARGET_LINK_LIBRARIES(MyApp.exe myappLib ${Trilinos_LIBRARIES} ${Trilinos_TPL_LIBRARIES})
```

How do I get Trilinos?

Sandia National Laboratories

- Current release (12.12.x) available for download (tarball)
  - http://trilinos.org/download/
- Trilinos lives on Github: github.com/trilinos/Trilinos
  - We use a 2-branch development model, like Kokkos
  - "develop" branch is what it says (compare to "trunk")
  - "master" branch updated often; some stability requirement
- Cray packages recent releases of Trilinos
  - http://www.nersc.gov/users/software/programming-libraries/math-libraries/trilinos/
  - `$ module load trilinos`
  - Cray tunes computational kernels for best performance
  - If you don't like their build options, you may also build Trilinos yourself, but link with their optimized kernels as a TPL (CASK)
- Most packages have a BSD license; a few are LGPL