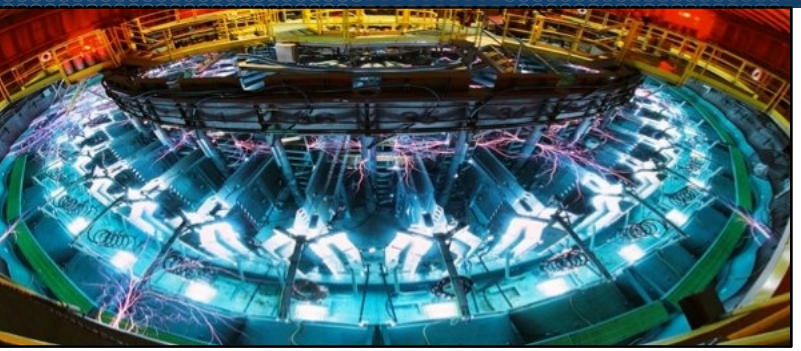
*Exceptional service in the national interest*

Sandia National Laboratories



# Kokkos CMake: Build Systems and for Modern C++

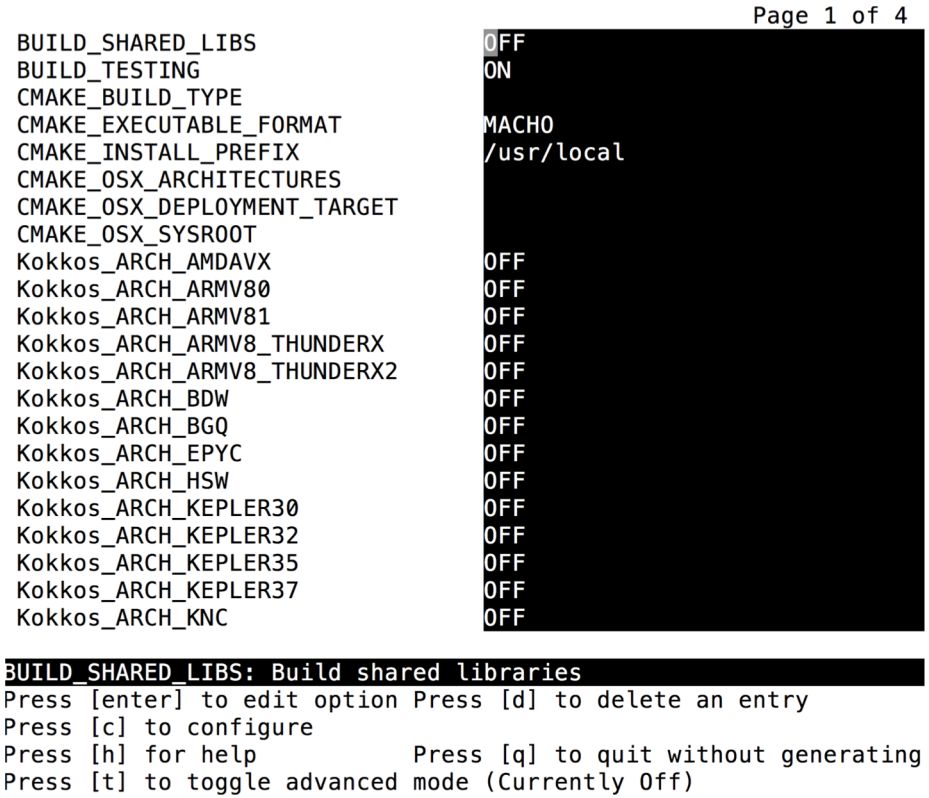
Jeremiah Wilke
Scalable Modeling and Analysis, Sandia National Labs, Livermore CA
Kokkos Bootcamp, Santa Fe, NM
1/17/2019
UUR: SAND2020-0654 PE

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

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.

# Building Kokkos

- cmake ${KOKKOS_SOURCE} –D{OPTION}:BOOL=ON –{OPTION}:STRING=NAME
  - Via command Line
- To get a list of options, use ccmake
  - ccmake –DCMAKE_CXX_COMPILER={} ${KOKKOS_SOURCE}



# Most important options for CUDA builds

Sandia National Laboratories

- -DKokkos_CXX_COMPILER=$kokkos/bin/nvcc_wrapper
  - Cannot use nvcc directly. Must use Kokkos-provided wrapper.
  - Can also use CUDA-enabled Clang
- -DKokkos_ENABLE_CUDA:BOOL=ON
  - Must activate CUDA backend
- -DKokkos_ARCH_X:BOOL=ON (X = VOLTA70 on most new-ish platforms)
  - Must activate at least one architecture (CMake will error if not)
- -DKokkos_ENABLE_CUDA_LAMBDA:BOOL=ON
  - Enable the use of lambdas in C++ code
  - Pretty much always want this on (we should probably change default to ON)
- -DKokkos_ENABLE_CUDA_RELOCATABLE_DEVICE_CODE:BOOL=ON
  - Use a CUDA function compiled in another translation unit)
  - Caveat: No RDC with Clang yet
- -DKokkos_CXX_STANDARD
  - 11 or 14 with nvcc_wrapper (17 coming soon?)
  - Need Clang for C++17 support

# Most important options for OpenMP builds

Sandia National Laboratories

- -DKokkos_CXX_COMPILER=<openmp-enabled-compiler>
  - Clang defaults to use libomp
- -DKokkos_ENABLE_OPENMP:BOOL=ON
  - Must activate OpenMP backend
- -DKokkos_ARCH_X:BOOL=ON (X = HSW,KNL,etc…)
  - Not required, but helpful

# A basic CMake project file creates and target and links to Kokkos

Sandia National Laboratories

```
cmake_minimum_required(VERSION 3.12) #need >= 3.12
project(myProject CXX) #kokkos is always a C++ project

find_package(Kokkos REQUIRED) #must find installed kokkos

add_executable(myExe source.cpp) #add my program
target_link_libraries(myExe PRIVATE Kokkos::kokkos) #link to Kokkos
```

# Kokkos not only makes writing C++ easier, it makes building C++ easier

Sandia National Laboratories

A single CMake function should populate build with all the necessary flags to build *correctly* and all the optimization/architecture flags to improve *performance*

```
FIND_PACKAGE(Kokkos REQUIRED)
ADD_LIBRARY(target ${SOURCES})
TARGET_LINK_LIBRARIES(target PUBLIC Kokkos::kokkos)
```

I need Kokkos to build – and anyone using my API needs Kokkos

```
FIND_PACKAGE(Kokkos REQUIRED)
ADD_LIBRARY(target ${SOURCES})
TARGET_LINK_LIBRARIES(target PRIVATE Kokkos::kokkos)
```

I need Kokkos to build – but using my API does not require Kokkos

```
KOKKOS_CHECK(
  DEVICES CUDA OPENMP
  OPTIONS CUDA_RELOCATABLE_DEVICE_CODE
  ARCH VOLTA70
)
```

Assert that the Kokkos configuration found meets expectations

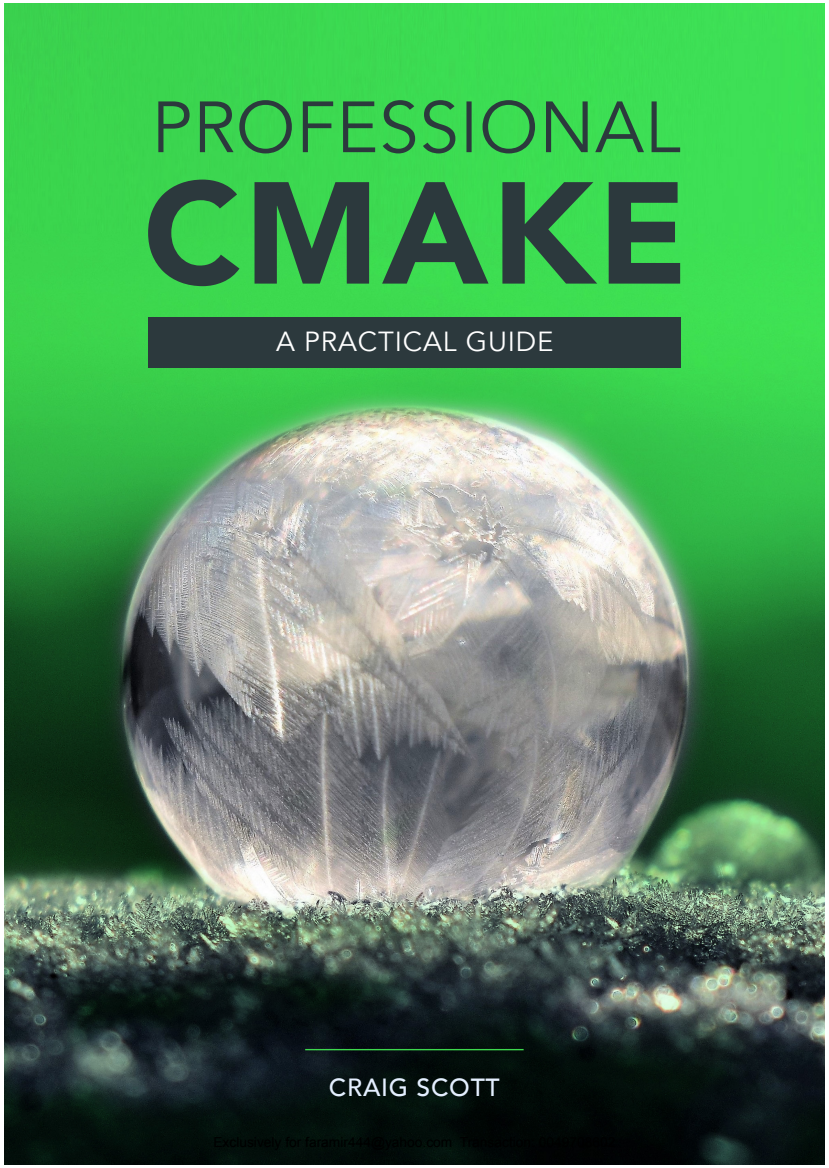
Installed Kokkos: cmake –DKokkos_ROOT=<PREFIX>

In-tree Kokkos: add_subdirectory(kokkos)

# Configuring your project based on Kokkos configuration options

Sandia National Laboratories

- Performance portability ideally means both your C++ code and your build system doesn't worry CUDA, OpenMP, HIP, etc
- cmake .. –DKokkos_ROOT=<kokkos-prefix> -DCMAKE_CXX_COMPILER=<…>
  - Just find Kokkos and link
- Variables and functions exist if customized logic needed based on Kokkos
- kokkos_check(
  OPTIONS [….]
  DEVICES [….]
  ARCH […])
- If we want to ASSERT that Kokkos was built with CUDA_LAMBDA
  - kokkos_check(OPTIONS CUDA_LAMBDA)
- If we want to QUERY whether Kokkos was built with CUDA
  - kokkos_check(DEVICES CUDA RESULT_VARIABLE KOKKOS_HAS_CUDA)
    if (KOKKOS_HAS_CUDA)
      …



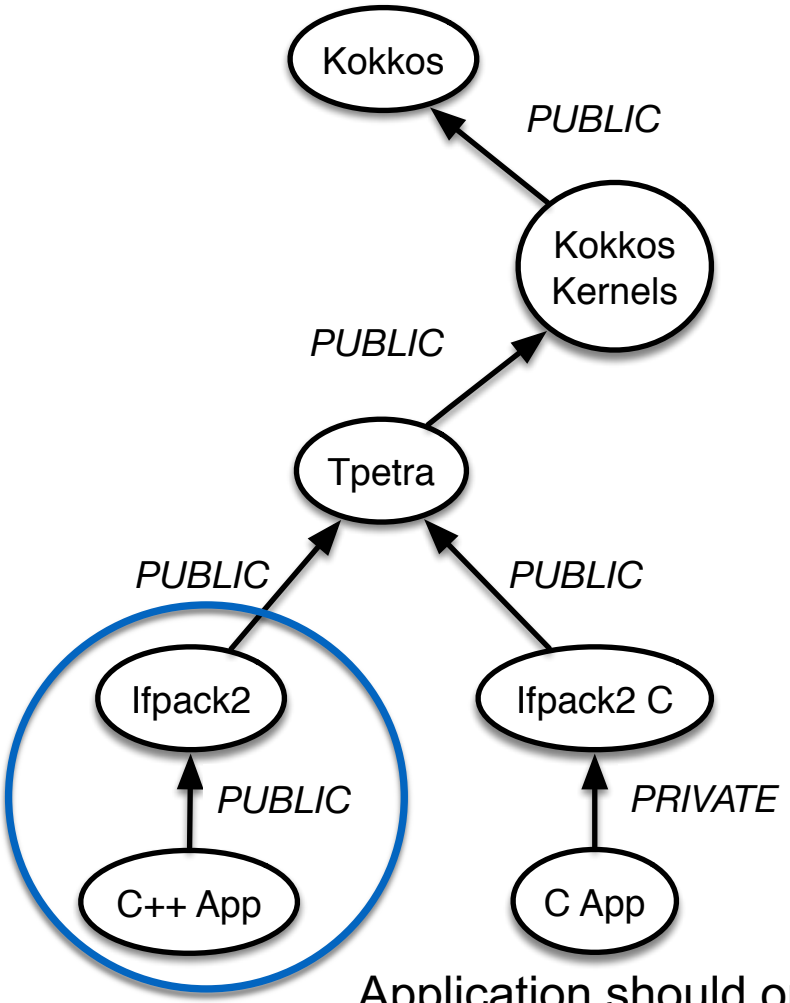

Best 40 dollars you will spend

# Modern CMake wants a clean separation of 'building' and 'using' libraries

Sandia National Laboratories

- CMake 3 (first "modern" version) released June 2014
  - Clean separation of building and using (targets and properties) has been recommended method since release
- All options should be applied specifically to TARGETS (libs, exes)
  - No more directly modifying CMAKE_CXX_FLAGS
  - No more global setting include directories and compiler flags
  - Your compiler/linker flags should be *specific* and *exact* to an individual library
- All include directories and compiler flags should be clearly defined as:
  - PUBLIC: Flag needed to build Kokkos and needed downstream to use Kokkos
    - Kokkos headers
    - Flags like –fopenmp or CUDA flags needed for the backend
    - Minimum C++ standards
  - PRIVATE: Flag only needed to build Kokkos (not needed to use)
    - Certain warning flags
    - Certain optimization flags

# Building and using makes "smaller" interfaces between libraries, solves transitive dependencies




Automake requires collecting and forwarding, e.g.

```
KokkosKernels_CXX_FLAGS =
      $(LOCAL_CXX_FLAGS) +
      $(Kokkos_CXX_FLAGS)
```

TARGET_LINK_LIBRARIES(Ifpack2) makes C++ App depend transitively on Kokkos flags (PUBLIC)

TARGET_LINK_LIBRARIES(Ifpack2_C) does not make C App depend transitively on Kokkos flags (PRIVATE)

Application should only know about its direct dependencies

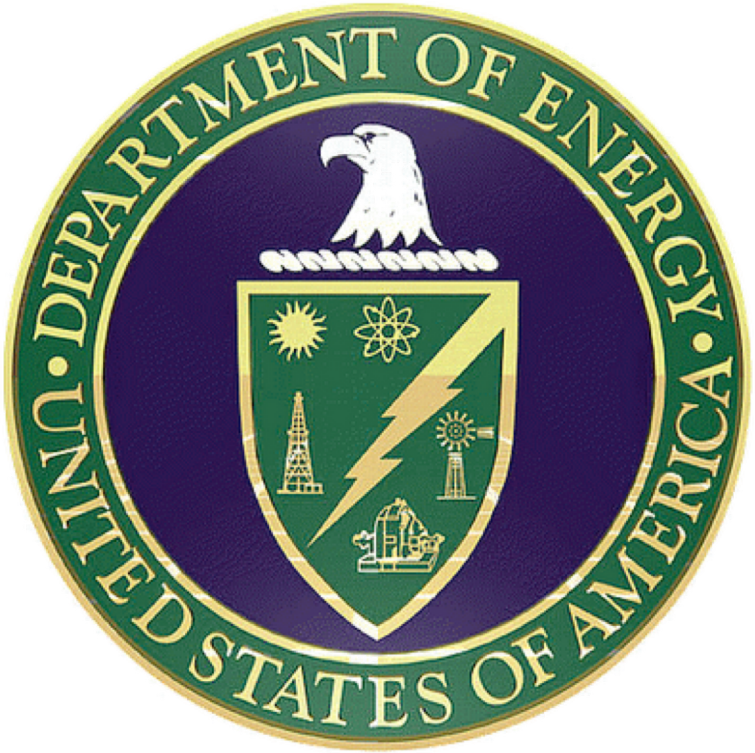
# Modern CMake (targets and properties) is much more robust than CMake 2

Sandia National Laboratories

- Modern CMake shrinks the interface size between two libraries from many, many CMake variables to a single function call
  - Gives Kokkos the freedom to make interface changes without breaking downstream codes
- Modern CMake enables the use of so-called *generator expressions* that can implement interesting build logic
  - Example: Library mixing C++/C/Fortan only adds Kokkos flags to C++ files
  - Example: Change preprocessor defines based on build type (debug/release) to add extra checks/optimizations flags
- Modern CMake helps resolves conflicting compiler features/options
  - Kokkos C++ standard vs. App C++ standard
  - Optimization/warning flags used to build Kokkos
- Avoid CMake Gotchas: Type-o's are just empty variables
  - TARGET_LINK_LIBRARIES(Kokkos::kokkos) will crash if Kokkos hasn't been correctly found or there are type-o's

# Acknowledgments

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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-NA-0003525.




National Nuclear Security Administration

Sandia National Laboratories