CMake as a Build System for DUNE

Markus Blatt

Dr. Markus Blatt
HPC-Simulation-Software & Services

OPM Symposium
May 28, 2013
Current Autotools Buildsystem

- dunecontrol provides an easy way of building the DUNE modules.
- 3 Step procedure:
  1. dunecontrol autogen: prepares modules for configure
  2. dunecontrol configure: configures modules
  3. dunecontrol make: builds modules.
- Works on Linux/Unix, Mac and partly on Windows (Cygwin)
- Uses several tools: autogen, libtool, makefiles, etc.
Why CMake (from a User Perspective)?

- Building is a two step procedure:
  - Configuration using CMake for the generator: `cmake /source/path`
  - Building using the Generators: e.g. `make`
- Supports (some) IDEs natively:
  - KDevelop
  - Code::Blocks
  - Eclipse/CDT
- Platform independent.
- Seems to be becoming a new standard.
User/Developer Requirements

- Shorter and more verbose CMakeLists.txt files when compared with Makefile.am files currently used in DUNE.
- Write equivalents in CMake to the m4-tests currently used in DUNE.
- Calculate module dependencies and automatically check whether they are satisfied.
- Write tests for finding DUNE modules.
- At least convert two DUNE modules to CMake.
- Build, test and install DUNE modules with CMake.
- Hopefully reduce build time.
Anatomy of a DUNE module with CMake

- Test for modules are found in cmake/modules/
  `<DuneModuleName>Macros.cmake`
- Package config files for installation: cmake/pkg/
  `<dune-module-name>-config.cmake.in`
- Build tree package configuration:
  `<dune-module>-{config,version}.cmake.in`
- `config.h.cmake` contains module specific preprocessor directives for platform
- various `CMakeLists.txt`
- `duneproject` can create this infrastructure for new modules.
Sample Top-level CMakeLists.txt for DUNE

```cmake
cmake_minimum_required(VERSION 2.8)
project(my-dune-module CXX)

# include the dune macros
include(DuneMacros)

# start a dune project
dune_project()

add_subdirectory(src)
add_subdirectory(m4)
add_subdirectory(dune)
add_subdirectory(doc)

# finalize the dune project
finalize_dune_project(GENERATE_CONFIG_H_CMAKE)
```
Finding and Using Package

- If not available DUNE provides Find\(<\text{Package}\>\).cmake
- Easy compile/link flags setting with add_dune_\(<\text{package}\>_\text{flags}
- Support for package specific tests that get executed by dependent modules.
- Less code duplication.
- DuneMacros
  - parses the dune.module files
  - generates dependency tree
  - searches for modules
  - executes tests.
The file *config.h*

- Contains macros and preprocessor defines for the platform.
- Watch out for the “ENABLE”- trick!
- Inherited by dependent modules
- Created by CMake from
  - `/config.h.cmake`
  - `...`
  - `/config.h.cmake`

- `config.h.cmake` has a module specific section
- with a private part.
- The non-private part of the module specific section gets inherited by dependant modules.
- Mimics the one created by autoconf.
The “ENABLE”-Trick

- Known and feared from DUNE’s autotools build system.
- Activates some packages on demand during build:
  - if COMPILE_DEFINITIONS contains ENABLE_PACKAGE
  - otherwise not.
- Needed because ...
  - distributions might not have activated some feature (e.g. SuperLU)
  - it enables testing with and without a specific feature (MPI)
- No problem if headers only!
Sample config.h.cmake

/* begin dune-istl */
/* begin private */
/* Name of package */
#define PACKAGE "@DUNE_MOD_NAME"

/* Define to the address for bug reports. */
#define PACKAGE_BUGREPORT "@DUNE_MAINTAINER@"

/* Define to the full name of this package. */
#define PACKAGE_NAME "@DUNE_MOD_NAME@"
/* ...*/
/* end private */

/* define if the Boost::Fusion headers are available */
cmakedefine HAVE_BOOST_FUSION

/* Define to ENABLE_BOOST if the Boost is there */
define HAVE_BOOST ENABLE_BOOST
/* end dune-istl */
Creating Documentation with CMake

- DUNE adds target make doc!
- Use `add_doxygen_target()` to create and install doxygen documentation
- Create PDFs with `dune_add_latex_document`:

  ```
  dune_add_latex_document(communication.tex FATHER_TARGET doc
  BIBFILES communication.bib DEFAULT_SAFEPDF INPUTS
  poosc08_test.cc
  IMAGE_DIRS figures)
  ```

- and install it with `make doc`:

  ```
  create_doc_install(${CMAKE_CURRENT_BINARY_DIR}/
  communication.pdf ${CMAKE_INSTALL_DOCDIR}/comm
  communication_safepdf)
  ```
Testing Framework

- Tests for are in subdirectories named tests.
- Several of these are supported per module.
- Built (nearly) on demand.
- For $(PROJECT\_SOURCE\_DIR)/.../tests $(PROJECT\_BINARY\_DIR)/.../tests/BuildTests.cmake is created to contain build commands and targets
- No tests built during “make all”
Example CMakeFile.txt for test

- Use CMakeLists.txt of parent directory.
- Add test target and add build dependencies
  
  ```cmake
  add_directory_test_target(_test_target)
  add_dependencies(${_test_target} ${TESTPROGS})
  ```

- Add tests:
  
  ```cmake
  add_executable(bitsetvectorortest bitsetvectorortest.cc)
  add_test(bitsetvectorortest bitsetvectorortest)
  ```
Creating New DUNE Modules

We have modified the duneproject script to create infrastructure for CMake:

- `<dune-module>/CMakeLists.txt`
- `<dune-module>/src/CMakeLists.txt`
- `<dune-module>/m4/CMakeLists.txt`
- `<dune-module>/dune/CMakeLists.txt`
- `<dune-module>/dune/<dune-module>/CMakeLists.txt`
- `<dune-module>/doc/CMakeLists.txt`
- `<dune-module>/doc/doxygen/CMakeLists.txt`
- `<dune-module>/doc/Doxyfile`
- `<dune-module>/config.h.cmake`
- `<dune-module>/cmake/pkg/<dune-module>-config.cmake.in`
- `<dune-module>/dune/<dune-module>-config.cmake.in`
- `<dune-module>/dune/<dune-module>-version.cmake.in`
Using CMake for DUNE

using dunecontrol

- Activate it using the –use-cmake switch (default is autotools)
- Provided option files will be parsed and translated for CMake

Pure CMake

- created build directories, configure, and build:

  ```bash
  for i in $MODULES; do
    mkdir $i-build; pushd $i-build
    cmake ../dune-$i
    make
    popd
  done
  ```

- Uses CMake’s package registry to automatically find location of module.
Comparison CMake vs. Autotools I

Makefile.am of autotools:

```
pamgtest_SOURCES = parallelamgtest.cc
pamgtest_CPPFLAGS = $(AM_CPPFLAGS) \ 
$(DUNEMPICPPFLAGS) $(PARMETIS_CPPFLAGS)
pamgtest_LDFLAGS = $(AM_LDFLAGS) \ 
$(DUNEMLDLDFLAGS) $(PARMETIS_LDFLAGS)
pamgtest_LDADD = $(PARMETIS_LIBS) \ 
$(DUNEMLDLIBS)
```

CMakeLists.txt of CMake:

```
add_executable(pamgtest "parallelamgtest.cc")
target_link_libraries(pamgtest "dunecommom")
add_dune_parmetis_flags(pamgtest)
```
Comparison CMake vs. Autotools II

Speed measured for building dune-common, dune-geometry, dune-grid and dune-istl with no compiler options and gcc-4.7.2. For autotools we used the configure-cache option to speed things up:

<table>
<thead>
<tr>
<th></th>
<th>CMake</th>
<th>autotools</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure + make</td>
<td>1m39s</td>
<td>3m58s</td>
</tr>
</tbody>
</table>

- We gain a factor of 2.4
- Rumors: There are faster generators than make
CMake SuperProjects

• Builtin support for external projects as dependencies:
  • Download (git, svn, homepage)
  • Configure
  • Build

• external projects can be made optional
• No need for the user to check, download, and build 3rd party software.
• Makes reproducible science possible.
• See e.g. Titan http://titan.sandia.gov/
Other projects with CMake

- Projects using CMake
  - UFC/Dolphin
  - Trilinos
  - Boost (evaluation phase during modularization)
- Project to watch: Ryppl [http://ryppl.org/](http://ryppl.org/)
  - Distributed cross-platform software management system
  - Version control, test management, package management, release management, and reporting
  - Early project phase
Status and Outlook

- All core modules do support CMake
- New modules do have CMake support
- Configuring, building, testing work.
- Missing header check and dune-web support.
- Time to convince the rest of the DUNE developers.
DUNE User Meeting

- September 24-25, 2013
- University Aachen, Germany
- Talks by DUNE users
- DUNE developer meeting from September 25
- Time to show off the OPM stuff!!
What can we do for you?

Efficient simulation software made to measure.

Expertise
- Parallelization
- High-Performance-Computing
- Numerical Methods
- Efficient Algorithms
- Software for technical simulations
- C++

Your Advantages
- Efficient Solutions
- Transparency
- Extensibility
- Quality Management by Researchers
- Use of latest Research Results
- No Runtime Costs

Dr. Markus Blatt -
HPC-Simulation-Software & Services

Clients
- Siemens
- Statoll
- TWT
- You?

Services
- Consulting
- Software Engineering
- Training
- Development of Algorithms

Hans-Bunte-Str. 8-10, 69123 Heidelberg, Germany
http://www.dr-blatt.de