CMake as a Build System for DUNE

Markus Blatt[1] Christoph Grüninger[2]

[1]HPC-Simulation-Software & Services
[2]Department of Hydromechanics and Modelling of Hydrosystems, University of Stuttgart

2nd Dune User Meeting
September 26, 2013

Autotools build system

dunecontrol provides an easy way of building DUNE modules

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: autoconf, automake, libtool, make files, 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, ninja-build
- generate projects for Xcode, KDevelop, Code::Blocks, Eclipse/CDT, Visual Studio
- platform independent
- easier to update
- faster configure
- shorter and more verbose build system files
- less cryptic files and output
- out of source builds
- used by increasing number of projects

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
- various CMakeLists.txt
- duneproject can create this infrastructure for new modules.

Sample Top-level CMakeLists.txt for DUNE

```
1  cmake_minimum_required(VERSION 2.8.6)
   project(dune-testtest CXX)
3  [..]
   #find dune-common and set the module path
5  find_package(dune-common)
   list(APPEND CMAKE_MODULE_PATH ${dune-common_MODULE_PATH}
7    "${PROJECT_SOURCE_DIR}/cmake/modules")

9  #include the dune macros
   include(DuneMacros)
11
   # start a dune project with information from dune.module
13 dune_project()

15 add_subdirectory("dune")
   [..]
17 add_subdirectory("m4")

19 # finalize the dune project, e.g. generating config.h etc.
   finalize_dune_project(GENERATE_CONFIG_H_CMAKE)
```

Finding and Using Package

- If not available DUNE provides Find<Package>.cmake
- Easy compile/link flags setting with `add_dune_<package>_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
  - <dune-common-module-source>/config.h.cmake
  - ...
  - <module-source>/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.
- If package found HAVE_<PACKAGE> = ENABLE_<PACKAGE>
- Activates some packages if COMPILE_DEFINITIONS contains ENABLE_<PACKAGE>
- 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

```
add_directory_test_target(_test_target)
add_dependencies(${_test_target} ${TESTPROGS})
```

- Add tests:

```
add_executable(bitsetvectortest bitsetvectortest.cc)
add_test(bitsetvectortest bitsetvectortest)
```

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>/config.h.cmake`
- `<dune-module>/cmake/pkg/<dune-module>-config.cmake.in`
- `<dune-module>/<dune-module>-config.cmake.in`
- `<dune-module>/<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:

```
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)\
$(DUNEMPILDFLAGS) $(PARMETIS_LDFLAGS)
pamgtest_LDADD = $(PARMETIS_LIBS)\
$(DUNEMPILIBS)
```

CMakeLists.txt of CMake:

```
add_executable(pamgtest "parallelamgtest.cc")
target_link_libraries(pamgtest "dunecommon")
add_dune_parmetis_flags(pamgtest)
```

Comparison CMake vs. Autotools II

Speed measured (gcc-4.7.3, openSuse 12.3, CMake 2.8.11)

1. module=dune-common autogen:configure
2. CONFIGURE_ FLAGS="CXXFLAGS="-O0" –cache-file=/tmp/dune.cache" module=dune-localfunctions,dune-istl all
3. re-run 2. with cache

autotools use configure-cache

	CMake	autotools
1. dune-common	11.4s	20.7s
2. core modules	2m02s	1m45s
3. re-run	9s	1m23s

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

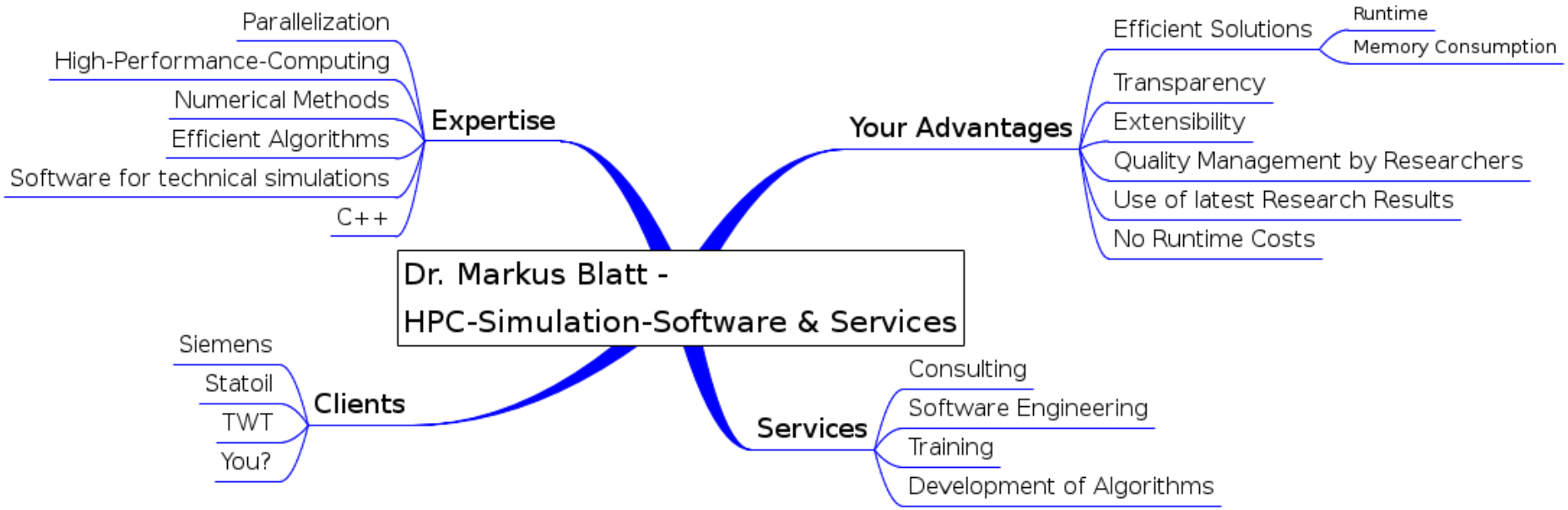
- UFC/Dolphin
- Trilinos
- deal.II
- Boost (evaluation phase during modularization)
- LLVM / Clang
- ParaView, The Visualization Toolkit
- Qt5, KDE, MySQL, Blender

Status and Outlook

- All core modules do support CMake
- New modules do have CMake support
- Configuring, building, testing work.
- Missing `make dist` and dune-web support.
- Time to convince the rest of the DUNE developers.

What can we do for you?

Efficient simulation software made to measure.



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