CMake: Improving Software Quality and Process

January 17, 2010
Camp KDE

Dr. Marcus D. Hanwell
R&D Engineer
Kitware, Inc.
marcus.hanwell@kitware.com
Kitware: The Company

- Founded in 1998
- Founded by 5 previous employees of GE Corporate Research
- Privately held, profitable from creation
- Revenues projected at $9 million in 2009
  - ~$14 million if subcontractors are included
- Approximately 65 employees
  - ~30% growth in 2009
  - More than 30 PhDs
Kitware: Core Technologies

- Software
- Process
- Supercomputing
- Visualization
- Medical Imaging
- Computer Vision
- Open-Source Toolkits
  - Registration
  - Segmentation
  - Image Processing
  - Measurement
  - Responsive GUI
- Large data
- Parallel computing
- Client/Server
- Web/grid architectures
- Human/Computer Interaction

Expertise in:
- Behavior/event recognition
- Detection and tracking
- Segmentation
- Change Detection

- Insight ToolKit (ITK)
- Visualization Toolkit (VTK)
- CMake
- ParaView
- Publications and consulting
Open Source Projects

- CMake
- CDash
- VTK
- ITK
- ParaView
- IGSTK
- BatchMake
NAMIC: National Alliance for Medical Computing

• An NIH national center for biomedical computing
• Kitware leads the NAMIC engineering core
• Committed to open science (source and data)
• New funding to port Slicer (open source) to Qt
• Aimed at leading edge clinical researchers
• More funding bodies are seeing value in open source
CMake: Cross Platform Build System

- Why CMake?
- CMake features
- Using CMake
- Creating CMake input files
- Where to get help
  - Mastering CMake book
  - Web page: www.cmake.org
  - http://www.cmake.org/Wiki/CMake
  - mailing list: cmake@cmake.org
Key Components

• CMake
  – Cross platform build

• CTest
  – Software testing and submission

• CDash
  – Aggregation of test results on a web server

• CPack
  – Cross platform software packaging
Why CMake? Everyone is using it

KDE 2006 – Tipping Point!

• 1200+ downloads per day from www.cmake.org
• Major Linux distributions and Cygwin provide CMake packages
• KDE, Second Life, Boost (experimentally), many others
How CMake Changes The Way We Build C++

• Boost aims to give C++ a set of useful libraries like Java, Python, and C#

• Give C++ compile portability, like the compile once and run everywhere of Java, and C#
  – Same build tool and files for all platforms
  – Easy to mix both large and small libraries
Who Is Involved?

Users

• KDE
• Second Life
• ITK
• VTK
• ParaView
• Trilinos
• Scribus
• Boost (experimentally)
• MySQL
• LLVM
• Many more

Supporters

• Kitware
• ARL
• National Library of Medicine
• Sandia National Labs
• Los Alamos National Labs
• NAMIC
Autonomous Systems Use CMake!

Argo is fully automated, with the onboard computers using code built using CMake. The helicopter is also semi-automated with similar code built using CMake!
CMake Features

• One simple language for all platforms
  – Windows, Mac OS X, Linux, UNIX variants
  – HPC/embedded platforms via cross-compilation (ParaView/Python)

• Generates native build systems
  – Makefiles (GNU, NMake, Borland, etc.)
  – KDevelop, Eclipse
  – Visual Studio 6, 7, 8, 9 IDE
  – Xcode

• Out-of-source build trees leave source clean
• Interactive configuration via GUI
• Multiple configurations (debug, release, etc.)
CMake Features - continued

• Automatic analysis
  – Implicit dependencies (C, C++, Fortran)
  – Transitive link dependencies
  – Ordering of linker search path and RPATH

• Advanced Makefile generation
  – Modular, Fast, Parallel
  – Color and progress display
  – Help targets – make help
  – Preprocessor targets – make foo.i
  – Assembly targets – make foo.s
Input to CMake

- Simple scripting language in CMakeLists.txt file(s)
- Built-in commands for common rules
  - `add_library(MyLib MyLib.cxx)`
  - `add_executable(MyExe MyMain.cxx)`
- Example project using Qt 4 and new syntax:
  ```
  cmake_minimum_required(VERSION 2.8)
  project(QtProject)
  find_package(Qt4 4.5.2 COMPONENTS QtCore QtGui QtOpenGL REQUIRED)
  include(${QT_USE_FILE})
  add_executable(mycoolapp main.cpp)
  target_link_libraries(mycoolapp ${QT_LIBRARIES})
  ```
Input to CMake: KDE Application

• Very little needs to change,

```
cmake_minimum_required(VERSION 2.8)
project(KDEProject)
find_package(KDE4 4.4 REQUIRED)
include(KDE4Defaults)
include_directories(${KDE4_INCLUDES})
kde4_add_executable(mykdeapp main.cpp)
target_link_libraries(mykdeapp
    ${KDE4_KDEUI_LIBS})
install(TARGETS mykdeapp
    ${INSTALL_TARGETS_DEFAULT_ARGS})
```
CMake Scripts

• cmake -E command
  – Cross platform command line utility
  – E.g. copy or remove files, conditionally copy...

• cmake -P script.cmake
  – Cross platform scripting utility
  – Does not generate a CmakeCache
  – Ignores all commands specific to build environment generation
CMake Process

Configure Step

- Read CMakeCache.txt
- Read CMakeLists.txt files
- Write CMakeCache.txt

Generate Step

- Write Makefiles or projects
Automatic Testing Benefits

Video of ParaView Nightly Testing
CDash Subproject Support

Main Project

Sub Projects
CDash: Query Filters
CTest Command Wrappers

Build Time: 2009-05-04 01:53:37 MDT
Found 1 Warnings
Errors are here.

Warning while building C++ object file "CMakeFiles/Kokkos_BaseSparseSolve.dir/cxx_main.cpp.o" in target Kokkos_BaseSparseSolve.

<table>
<thead>
<tr>
<th>Source File</th>
<th>packages/kokkos/test/BaseSparseSolve/cxx_main.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Kokkos</td>
</tr>
<tr>
<td>Command</td>
<td>&quot;/Users/bmprs/bin/gcc-4.3.3/bin/g++ &quot;-mnacozx-version-min=10.5&quot; &quot;-ansi&quot; &quot;-pedantic&quot; &quot;-Wall&quot; &quot;-Wno-long-long&quot; &quot;-Wwrite-strings&quot; &quot;-g&quot; &quot;-00&quot; &quot;-D_GLIBCXX_DEBUG&quot; &quot;-I/Users/bmprs/nightly/Trilinos.base/SERIAL_DEBUG/BUILD/packages/kokkos/src&quot; &quot;-I/Users/bmprs/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/src&quot; &quot;-I/Users/bmprs/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/test/BaseSparseSolve/.../BaseSparseMultiply&quot; &quot;-c&quot; &quot;CMakeFiles/Kokkos_BaseSparseSolve.dir/cxx_main.cpp.o&quot; &quot;-o&quot; &quot;/Users/bmprs/nightly/Trilinos.base/SERIAL_DEBUG/Trilinos/packages/kokkos/test/BaseSparseSolve/cxx_main.cpp&quot;</td>
</tr>
<tr>
<td>Directory</td>
<td>/Users/bmprs/nightly/Trilinos.base/SERIAL_DEBUG/BUILD/packages/kokkos/test/BaseSparseSolve</td>
</tr>
<tr>
<td>Exit Condition</td>
<td>0</td>
</tr>
<tr>
<td>Standard Output</td>
<td></td>
</tr>
</tbody>
</table>

---

Kitware

CDash 1.5.0 © 2000 Kitware Inc. [report problems]
Valgrind/Purify

Dynamic analysis started on 2009-05-03 03:38:06

Site Name: dash17.Kitware
Build Name: Linux-g++4.0

Valgrind/Purify

Dynamic analysis started on 2009-05-04 03:37:17

Site Name: dash17.Kitware
Build Name: Linux-g++4.0
TestMainBlockPassed

1001== Memcheck, a memory error detector.
1002== Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
1003== Using LibVEX, a library for dynamic binary translation.
1004== Copyright (C) 2004-2007, and GNU GPL'd, by Greenouze LLC.
1005== Using valgrind-3.3.3, a dynamic binary instrumentation framework.
1006== Copyright (C) 2000-2007, and GNU GPL'd, by Julian Seward et al.
1007== For more details, run with: -v
1008==
1009== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 119 from 1)
1010== malloc/free: in use at exit: 32,016 bytes in 27 blocks.
1011== malloc/free: in use at exit: 23,704 bytes, 7,959 bytes allocated.
1012== For counts of detected errors, run with: -v
1013== searching for pointers to 327 not-freeed blocks.
1014== checked 2,298,764 bytes.
1015==
1016== 64 bytes in 1 blocks are still reachable in loss record 15 of 36
1017== at 0x61D87: realloc (vg_replace_malloc.c:1306)
1018== by 0x62F36E: (within /usr/lib/x86_64-linux-gnu/x86_64-2.6.0)
1019== by 0x62F45E: (within /usr/lib/x86_64-linux-gnu/x86_64-2.6.0)
1020== by 0x62F4A0: XGetDynamicDatabase (in /usr/lib/x86_64-linux-gnu/x86_64-2.6.0)
1021== by 0x65F121: XGetRandomGenerator (in /usr/lib/x86_64-linux-gnu/x86_64-2.6.0)
1022== by 0x659DC7: XCopyDisplay (in /usr/lib/x86_64-linux-gnu/x86_64-2.6.0)
1023== by 0x4972D3: vtXOpenGLRenderWindowInteractor::Initialize (vtXOpenGLRenderWindow.interactor.cxx:917)
1024== by 0x42F0D0: vtXOpenGLRenderWindow::Render () (vtXOpenGLRenderWindow.cxx:265)
1025== by 0x41E505: vtXOpenGLRenderWindow::Render () (vtXOpenGLRenderWindow.cxx:1846)
1026== by 0x8011A6: TestMultiBlock:: (TestMultiBlock.cxx:142)
1027== by 0x805B3E: main (GraphicsCxxTests.cxx:506)
Email Notification

A submission to CDash for the project CMake has failing tests. You have been identified as one of the authors who have checked in changes that are part of this submission or you are listed in the default contact list.

Details on the submission can be found at http://www.cdash.org/CDash/buildSummary.php?buildid=322849

Project: CMake
Site: destiny.kitware
Build Name: HP-UX-aCC
Build Time: 2009-04-29T14:28:00 EDT
Type: Continuous
Tests failing: 85

*Tests failing* (first 5)
FindPackageTest (http://www.cdash.org/CDash/testDetails.php?test=21959898&build=322849)
FindModulesExecuteAll (http://www.cdash.org/CDash/testDetails.php?test=21959899&build=322849)

-CDash on www.cdash.org
Kitware Hosted CDash

CDash is a web-based software testing server. CDash aggregates, analyzes and displays the results of software testing processes submitted from clients located around the world. Developers depend on CDash to convey the state of a software system, and to continually improve its quality. To learn more about CDash visit the main CDash website.

Starting a project is **easy** and **free**. In just a few clicks you can start monitoring the quality of your software development.

Start My Project >

### Available Dashboards

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
<th>Submissions</th>
<th>First build</th>
<th>Last activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>automoc4</td>
<td>automoc4 is a tool which makes moc-processing with Q64 easier.</td>
<td>264</td>
<td>2009-04-18T12:33:22 EDT</td>
<td>2010-01-13 17:03:42</td>
</tr>
<tr>
<td>Avogadro</td>
<td>Avogadro is an advanced molecular editor designed for cross-platform use in computational chemistry, molecular modeling, bioinformatics, materials science, and related areas. It offers flexible rendering and a powerful plugin architecture.</td>
<td>646</td>
<td>2009-01-28T14:01:47 EST</td>
<td>2010-01-13 12:06:33</td>
</tr>
<tr>
<td>Boost</td>
<td>Boost provides free peer-reviewed portable C++ source libraries.</td>
<td>505</td>
<td>2009-05-06T19:31:42 EDT</td>
<td>2010-01-13 00:19:39</td>
</tr>
<tr>
<td>cape</td>
<td>a c++ version</td>
<td>6</td>
<td>2010-01-07 14:46:03 EST</td>
<td>2010-01-07 15:27:21</td>
</tr>
<tr>
<td>CERTI</td>
<td>CERTI</td>
<td>495</td>
<td>2008-09-26T18:27:04 EDT</td>
<td>2010-01-12 22:32:00</td>
</tr>
</tbody>
</table>
CDash Testing

- Purify / valgrind
- Coverage (gcov, bullseye)
- Configuration coverage
  - Make sure different OSes, libraries and options are covered
- Image difference testing
Testing with CMake/CTest

• Testing command in CMake
  – add_test (testName exeName arg1 arg2 …)
  – Executable is expected to return 0 for passed
  – Other passing criteria can be specified

• CTest – an executable that is distributed with CMake, can run tests in a project
  – Used for continuous integration testing
  – Client for CDash
CTest/CDash: Current State

- Alexander Neundorf has already done a lot of work getting KDE dashboards up
- Currently testing full KDE modules
- If all developers start receiving email
  - Is a KStars dev interested in kalzium errors?
  - How long until they filter/unsubscribe?
  - Marble devs only receive Marble reports

```
<table>
<thead>
<tr>
<th>Build Name</th>
<th>Update</th>
<th>Configure</th>
<th>Build</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Files</td>
<td>Error</td>
<td>Warn</td>
<td>Min</td>
</tr>
<tr>
<td>FreeBSD-cmake-2.8.0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>Linux-c++</td>
<td>12</td>
<td>0.4</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Linux-gcc-4.3.2-cmake-2.6.2</td>
<td>12</td>
<td>0.2</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>SunOS-sunstudio-12.1-cmake-2.6.2</td>
<td>12</td>
<td>0.2</td>
<td>0</td>
<td>0.6</td>
</tr>
</tbody>
</table>
```
KDE and Subprojects

• KDE modules are quite large
  – Typically more than 15 components
  – Developers don't normally work on all components in a module
  – Without subprojects testing is too noisy

• Divide KDE modules?

• Split modules into subprojects?
  – New feature in CMake 2.8
Subproject Requirements

• Add LABELS property to build system
  – Source
  – Tests

• Create custom targets for subprojects

• CMake 2.8 CTest driver scripts
  – Only need CMake 2.8 on submission hosts
  – Continue using >=CMake 2.6.2 to build

```bash
set_target_properties(avogadro PROPERTIES
  VERSION ${Avogadro_VERSION_FULL} SOVERSION 1
  LABELS "libavogadro")
```
What Subprojects Get Us

• Build KDE modules in parts
• Test KDE modules in parts
• Developers receive emails about issues
  – Only for the subproject they committed to
  – CDash summaries for each part
    • Kalzium
    • Marble
  – Not everyone who committed to kdeedu!
CTest Driver Script

• Update the source directory
  – CTest added Git support in 2.8
• Configure the module
• Loop over each subproject in the module
  – Build the custom subproject target
  – Run the tests for the subproject
• Submit the result of each part
• Uses new CTest script functionality
Example CTest Driver Script

cmake_minimun_required(VERSION 2.8)
set(CTEST_SOURCE_DIRECTORY $ENV{HOME}/kdeedu)
set(CTEST_BINARY_DIRECTORY $ENV{HOME}/kdeedubuild)
set(CTEST_CMAKE_GENERATOR “Unix Makefiles”)
set(CTEST_BUILD_COMMAND “make -j9”)

# Empty the binary directory - clean build
ctest_empty_binary_directory(“${CTEST_BINARY_DIRECTORY}”)
Example CTest Driver Script

# Write initial cache
file(WRITE
  "${CTEST_BINARY_DIRECTORY}/CMakeCache.txt" "
  CMAKE_BUILD_TYPE:STRING=DebugFull
  CMAKE_CXX_COMPILER:FILEPATH=/usr/ccache/bin/c++
  ")
# Now start the update and configure steps
set(CTEST_UPDATE_COMMAND "svn")
ctest_start(Nightly)
ctest_update(SOURCE "${CTEST_SOURCE_DIRECTORY}"
ctest_submit(PARTS Update Notes)
ctest_configure(BUILD "${CTEST_BINARY_DIRECTORY}" APPEND)
ctest_submit(PARTS Configure)
Example CTest Driver Script

include("${CTEST_SOURCE_DIRECTORY}/CTestConfig.cmake")

foreach(sub ${CTEST_PROJECT_SUBPROJECTS})
  set_property(GLOBAL PROPERTY SubProject ${sub})
  set_property(GLOBAL PROPERTY Label ${sub})
  set(CTEST_BUILD_TARGET ${sub})
  ctest_build(BUILD "${CTEST_BINARY_DIRECTORY}" APPEND)
  ctest_submit(PARTS Build)
  ctest_test(BUILD "${CTEST_BINARY_DIRECTORY}" INCLUDE_LABEL ${sub})
  ctest_submit(PARTS Test)
endforeach()
Using CTest Driver Scripts

• Invoke the CTest binary in script mode
  – ctest -S /path/to/driver_script.cmake
• Can easily be run from a cron job, etc.
• Can submit parts of the submission
• Allows for much tighter control, scripting
• Access to most of the CMake language
CTest and Git

• CTest must understand version control
  – What files have changed?
  – Who changed those files?
  – What to checkout for a nightly dashboard?
• CMake 2.8 added support for Git, Mercurial and Bazaar (distributed VCS)
• Also abstracted support
  – Easier to add others in the future
The Future: CTest and Git

• Is it possible to test changes before they are committed?
  – With distributed version control it is!
• Developer pushes experimental branch
• Requests CDash server tests this branch on all supported architectures
• If everything is green, proceed to merge
• Development tree is more stable!
DVCS: New Workflows

“Blessed” master

Integration

Tested

Only merged after testing

+++ git
Cross Compiling

• New chapter in the new CMake book
• Far more important to the KDE community with embedded devices, e.g. N900+
• Significant work has already been done
• I need to read up on this!
Kitware is Hiring

• Looking for talented developers
• Opportunities for internships this summer
• Work on interesting, challenging problems
  – Exceptional skill in C++
  – Other languages such as Python, Tcl, Java
  – Work on Linux, Unix, Mac OS X and/or Windows

http://www.kitware.com/company/jobopps.html
Thank You

• Any questions?
• I will be available afterwards
• If you would like to learn more
  – Attend the tutorial session, Monday @ 2pm
• CMake is more than just a build system!

http://www.cmake.org/

mhanwell@kde.org
CMake Tutorial

- Summary of CMake
- CMake input files
- CMake GUIs
- CTest – testing your code
- CPack – packaging
- CDash – collating test results