HPC Application Testing Framework - buildtest

Shahzeb Siddiqui

6/15/2017

Agenda

- Software Build Tools
- Requirements for Testing Framework
- Testing Strategy
- History
- What is buildtest
- Challenges
- Current Work
- References

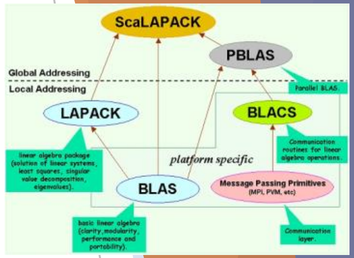
HPC Application Stack

OpenFOAM
The Open Source CFD Toolbox

ANACONDA

GCC

FFTW


ScaLAPACK
PBLAS
Global Addressing
Local Addressing
Parallel BLAS.
LAPACK
BLACS
linear algebra package (solution of linear systems, least squares, singular value decomposition, eigenvalues).
platform specific
Communication routines for linear algebra operations.
BLAS
Message Passing Primitives (MPI, PVM, etc)
basic linear algebra (clarity, modularity, performance and portability).
Communication layer.

Open MPI

ANSYS

R

NWCHEM
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

OpenCL

BOW TIE

intel
PARALLEL STUDIO XE

NVIDIA
CUDA

OpenMP

netCDF

GROMACS
FAST. FLEXIBLE. FREE.

python

ParaStation
MPI

MATLAB

MVAPICH

Java

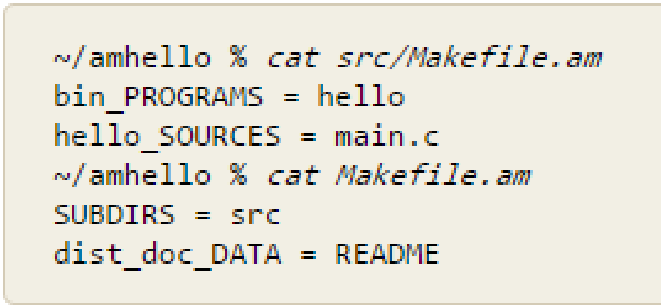
HDF
The HDF Group

intel
Math Kernel Library

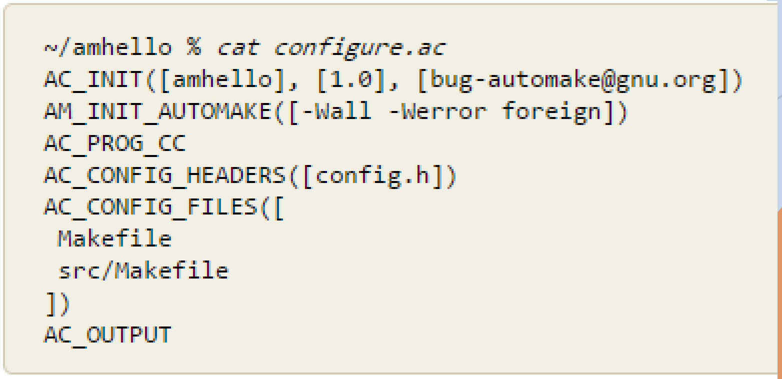
SCHRÖDINGER.

Software Build Tools

- Vendors typically provide test suite like **make test** that can perform test in the build directory and not on the binaries in the install path.
- Other testing mechanism such as CTest from vendors make use of CMakeLists.txt that must be configured. Too complex!!
- Testing utilities like Autoconf, Automake and autotest make use of M4 scripts for writing test suites that can be used by makefiles to run the test scripts

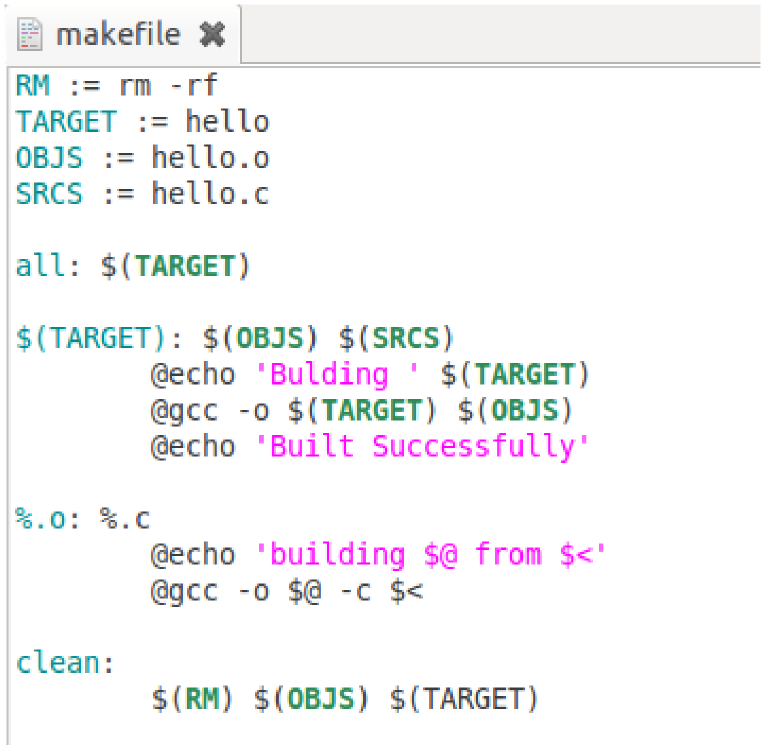


```
~/amhello % cat src/Makefile.am
bin_PROGRAMS = hello
hello_SOURCES = main.c
~/amhello % cat Makefile.am
SUBDIRS = src
dist_doc_DATA = README
```



```
~/amhello % cat configure.ac
AC_INIT([amhello], [1.0], [bug-automake@gnu.org])
AM_INIT_AUTOMAKE([-Wall -Werror foreign])
AC_PROG_CC
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([
 Makefile
 src/Makefile
])
AC_OUTPUT
```

Hello world Example in Make and CMake


makefile

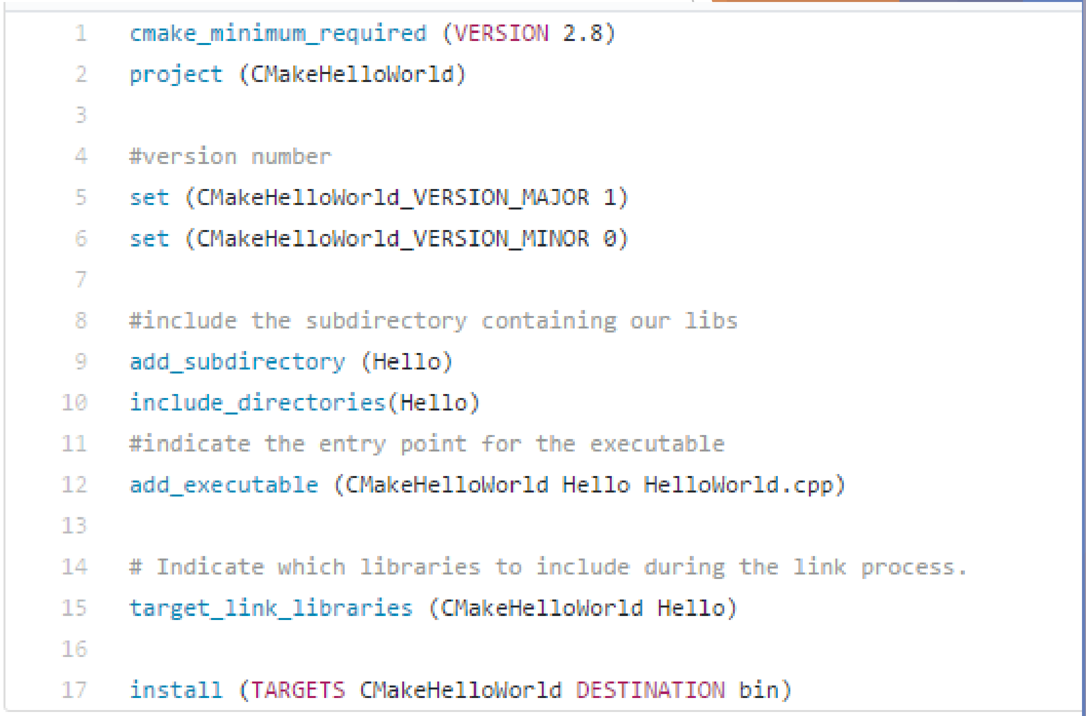
```
RM := rm -rf
TARGET := hello
OBJS := hello.o
SRCS := hello.c

all: $(TARGET)

$(TARGET): $(OBJS) $(SRCS)
	@echo 'Bulding ' $(TARGET)
	@gcc -o $(TARGET) $(OBJS)
	@echo 'Built Successfully'

%.o: %.c
	@echo 'building $@ from $<'
	@gcc -o $@ -c $<

clean:
	$(RM) $(OBJS) $(TARGET)
```



```
cmake_minimum_required (VERSION 2.8)
project (CMakeHelloWorld)

#version number
set (CMakeHelloWorld_VERSION_MAJOR 1)
set (CMakeHelloWorld_VERSION_MINOR 0)

#include the subdirectory containing our libs
add_subdirectory (Hello)
include_directories(Hello)
#indicate the entry point for the executable
add_executable (CMakeHelloWorld Hello HelloWorld.cpp)

# Indicate which libraries to include during the link process.
target_link_libraries (CMakeHelloWorld Hello)

install (TARGETS CMakeHelloWorld DESTINATION bin)
```

Software Build Tools

- GNU make, a widely used make implementation with a large set of extensions
- make, a classic Unix build tool
- Apache Ant, popular for Java platform development and uses an XML file format
- Apache Maven, a Java platform tool for dependency management and automated software build
- Gradle, an open-source build and automation system with a Groovy-based domain specific language (DSL), combining features of Apache Ant and Apache Maven with additional features like a reliable incremental build

https://en.wikipedia.org/wiki/List_of_build_automation_software

Requirements for Testing Framework

- Share test configs scripts among HPC community
- A universal HPC Test Toolkit
- Perform binary tests & compilation test
- Test configs should be easy to write
- Reproducible test builds
- Reuse test configs for any version of the application
- Conduct system package tests to detect potential bugs or corrupt system environment
- A mechanism to report PASS/FAIL for tests
- Ability to run subset of tests

Testing Strategy

- Binary Tests

  <executable> <param>

- Compilation Tests
  - Serial
    - buildcmd: <compiler> <source> -o <executable>
    - runcmd: ./<executable>
    - compiler = gcc, gfortran, g++, icc, ifort, icpc, nvcc
  - MPI
    - buildcmd: <mpi-wrapper> <source> -o <executable>
    - runcmd: mpirun –np <nproc> ./<executable>
  - Java
    - buildcmd: javac <source>.java
    - runcmd: java <source>
- Scripting Language like Python, R, Perl, Ruby, Lua
  - python example.py
  - ruby example.rb
  - perl example.pl
  - Rscript example.R
  - lua example.lua

Testing Strategy

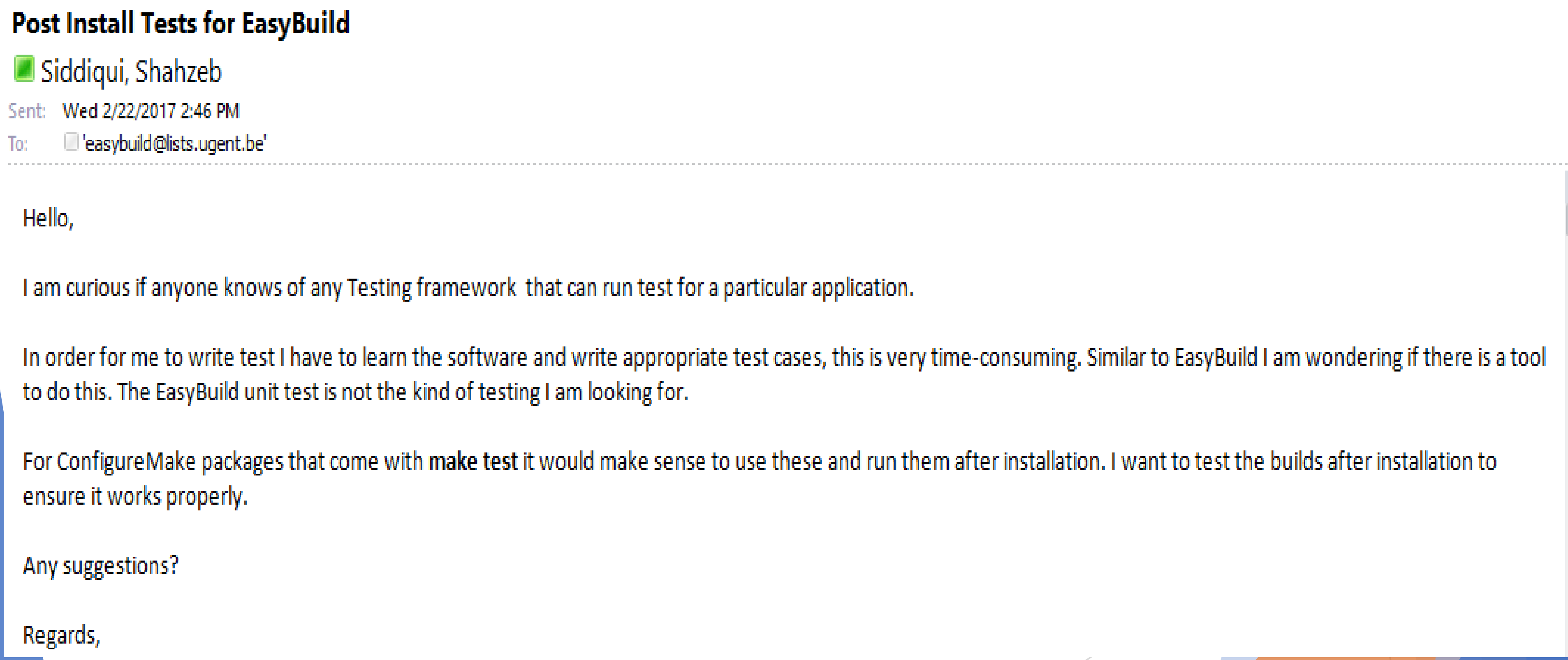
- Binary Tests are simple, just need to figure out the binary that resides in $PATH and run it with any options such as

  –version, -v, -V, --help, -h

- Compilation tests are not so straight forward.
- Most compilation tests have the following: compiler, source file, object files, executable name, build flags
- Serial programs can be done by running the executable, while mpi jobs are typically run through mpi launcher like mpirun or mpiexec
- Compiler can be detected by looking at the file extension
- Configurable options like CFLAGS, FFLAGS, CXXFLAGS, LDFLAGS can be done via YAML keys

History

- On Feb 22nd 2017, I reached out to the EasyBuild community for a testing framework for Post Installation Tests



Post Install Tests for EasyBuild

Siddiqui, Shahzeb

Sent: Wed 2/22/2017 2:46 PM

To: 'easybuild@lists.ugent.be'

Hello,

I am curious if anyone knows of any Testing framework that can run test for a particular application.

In order for me to write test I have to learn the software and write appropriate test cases, this is very time-consuming. Similar to EasyBuild I am wondering if there is a tool to do this. The EasyBuild unit test is not the kind of testing I am looking for.

For ConfigureMake packages that come with **make test** it would make sense to use these and run them after installation. I want to test the builds after installation to ensure it works properly.

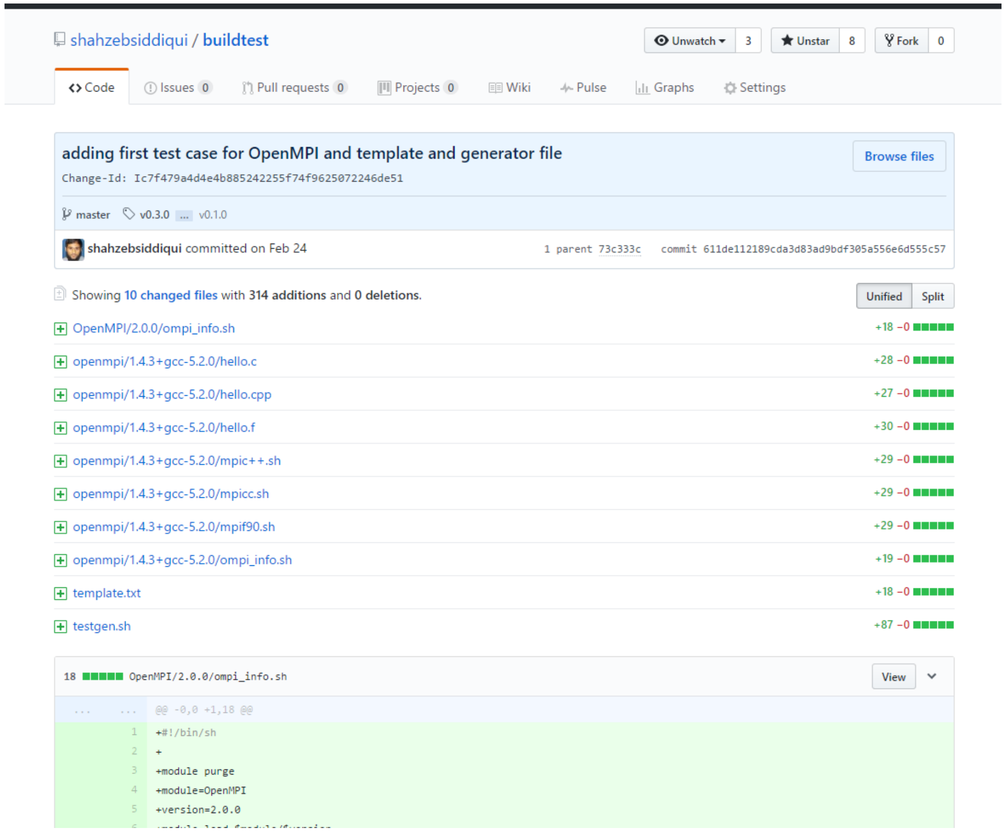
Any suggestions?

Regards,

History

- Originally called **testgen** was a shell script program that used templates and sed commands to write test scripts based on module name and compiler.
- The idea was to take argument from testgen –s <software> and apply SED commands to alter module load.
- Current implementation was not completely functional and SED could not cover special test cases
- Testgen was too dependent on SED which made this a problem
- Getopts feature was error prone and it did not provide all the elegant features of argparse Python library
- Next, I re-implemented testgen in Python now called **buildtest**

History - First commit



shahzebsiddiqui / buildtest

Unwatch 3 | Unstar 8 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

adding first test case for OpenMPI and template and generator file

```
Change-Id: Ic7f479a4d4e4b885242255f74f9625072246de51
```

master v0.3.0 ... v0.1.0

shahzebsiddiqui committed on Feb 24 — 1 parent 73c333c — commit 611de112189cda3d83ad9bdf305a556e6d555c57

Showing **10 changed files** with **314 additions** and **0 deletions**.

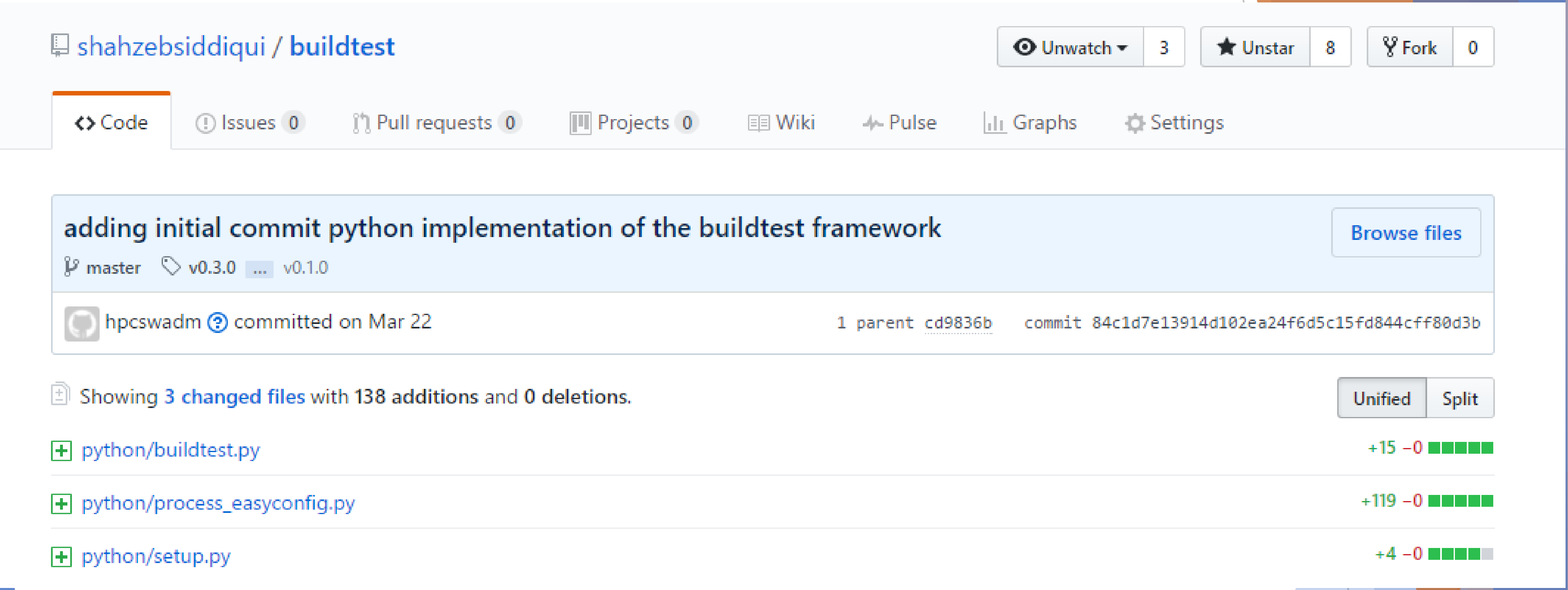
File	Changes
OpenMPI/2.0.0/ompi_info.sh	+18 −0
openmpi/1.4.3+gcc-5.2.0/hello.c	+28 −0
openmpi/1.4.3+gcc-5.2.0/hello.cpp	+27 −0
openmpi/1.4.3+gcc-5.2.0/hello.f	+30 −0
openmpi/1.4.3+gcc-5.2.0/mpic++.sh	+29 −0
openmpi/1.4.3+gcc-5.2.0/mpicc.sh	+29 −0
openmpi/1.4.3+gcc-5.2.0/mpif90.sh	+29 −0
openmpi/1.4.3+gcc-5.2.0/ompi_info.sh	+19 −0
template.txt	+18 −0
testgen.sh	+87 −0

18 OpenMPI/2.0.0/ompi_info.sh

```
@@ -0,0 +1,18 @@
+#!/bin/sh
+
+module purge
+module=OpenMPI
+version=2.0.0
```

https://github.com/shahzebsiddiqui/buildtest/commit/611de112189cda3d83ad9bdf305a556e6d555c57

History



shahzebsiddiqui / buildtest

Unwatch 3 | Unstar 8 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

adding initial commit python implementation of the buildtest framework

Browse files

master v0.3.0 … v0.1.0

hpcswadm committed on Mar 22   1 parent cd9836b   commit 84c1d7e13914d102ea24f6d5c15fd844cff80d3b

Showing 3 changed files with 138 additions and 0 deletions.   Unified | Split

python/buildtest.py +15 −0

python/process_easyconfig.py +119 −0

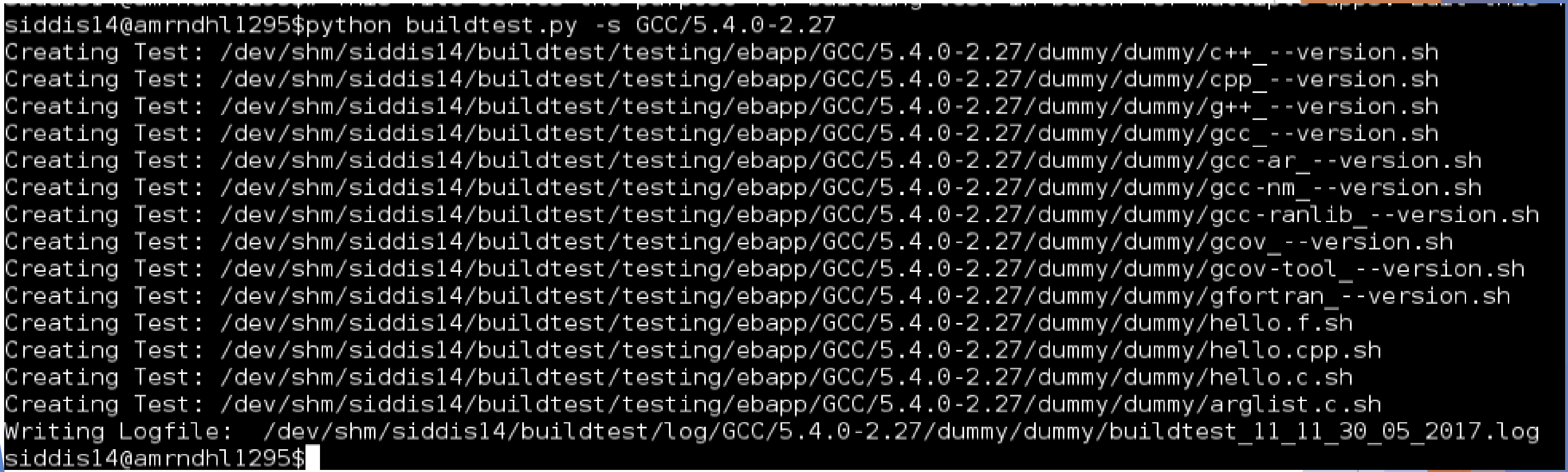
python/setup.py +4 −0

https://github.com/shahzebsiddiqui/buildtest/commit/84c1d7e13914d102ea24f6d5c15fd844cff80d3b

What is buildtest

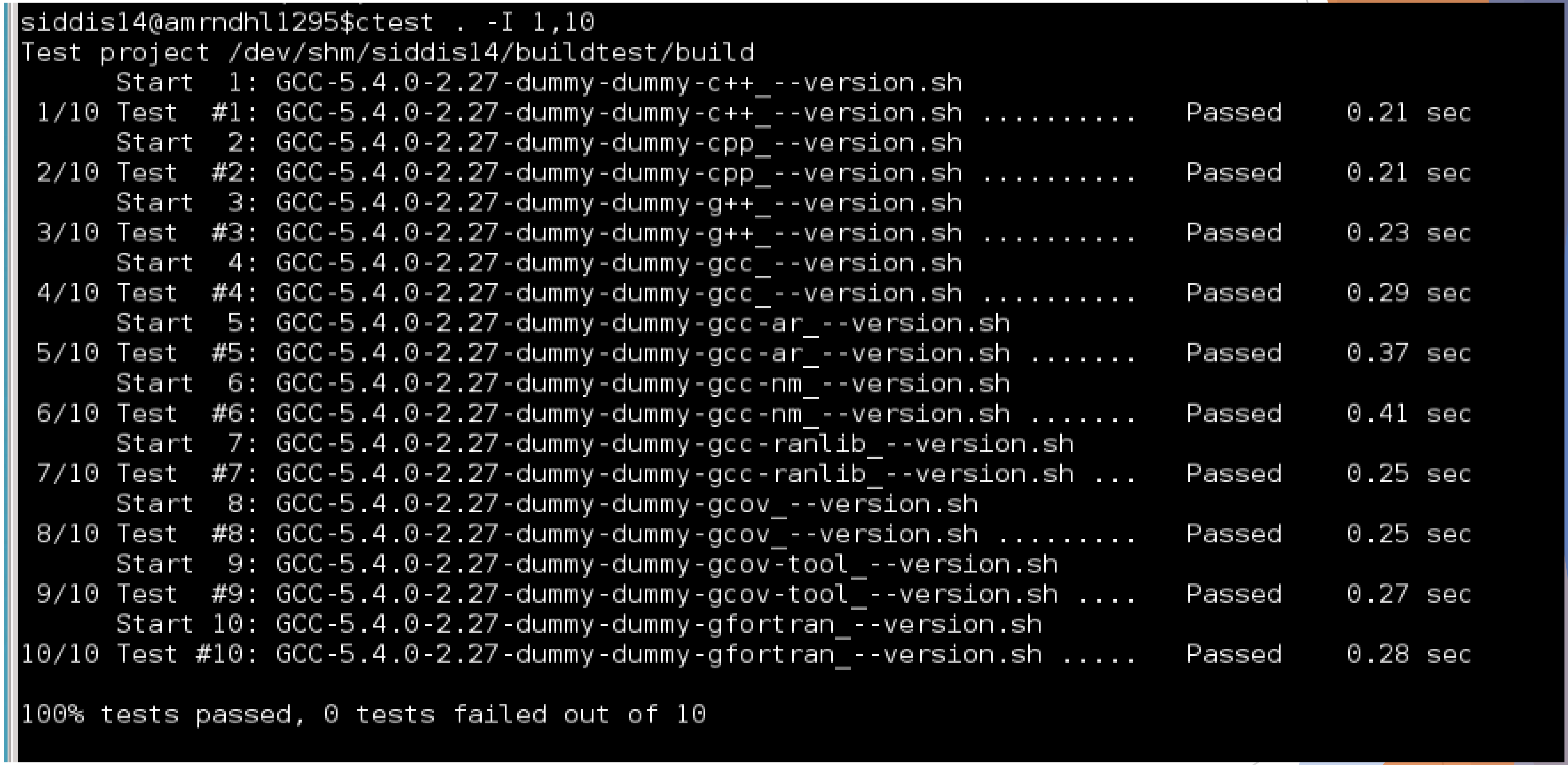
- Automatic test generating framework for writing tests scripts in YAML
- Creates tests for applications built with Easybuild as well as system package tests
- buildtest makes use of CTest for running all the test scripts and it can report whether the tests has PASSED or FAILED
- Buildtest can quickly write tests for R, Python, Perl without any YAML file, just add the script in the following repo:
  - R-buildtest-config: https://github.com/shahzebsiddiqui/R-buildtest-config
  - Python-buildtest-config: https://github.com/shahzebsiddiqui/Python-buildtest-config
  - Perl-buildtest-config: https://github.com/shahzebsiddiqui/Perl-buildtest-config

Building Test with buildtest



```
siddis14@amrndhl1295$python buildtest.py -s GCC/5.4.0-2.27
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/c++_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/cpp_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/g++_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gcc_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gcc-ar_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gcc-nm_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gcc-ranlib_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gcov_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gcov-tool_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/gfortran_--version.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/hello.f.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/hello.cpp.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/hello.c.sh
Creating Test: /dev/shm/siddis14/buildtest/testing/ebapp/GCC/5.4.0-2.27/dummy/dummy/arglist.c.sh
Writing Logfile:  /dev/shm/siddis14/buildtest/log/GCC/5.4.0-2.27/dummy/dummy/buildtest_11_11_30_05_2017.log
siddis14@amrndhl1295$
```

Running test with CTEST



```
siddis14@amrndhl1295$ctest . -I 1,10
Test project /dev/shm/siddis14/buildtest/build
      Start  1: GCC-5.4.0-2.27-dummy-dummy-c++_--version.sh
 1/10 Test  #1: GCC-5.4.0-2.27-dummy-dummy-c++_--version.sh ..........   Passed    0.21 sec
      Start  2: GCC-5.4.0-2.27-dummy-dummy-cpp_--version.sh
 2/10 Test  #2: GCC-5.4.0-2.27-dummy-dummy-cpp_--version.sh ..........   Passed    0.21 sec
      Start  3: GCC-5.4.0-2.27-dummy-dummy-g++_--version.sh
 3/10 Test  #3: GCC-5.4.0-2.27-dummy-dummy-g++_--version.sh ..........   Passed    0.23 sec
      Start  4: GCC-5.4.0-2.27-dummy-dummy-gcc_--version.sh
 4/10 Test  #4: GCC-5.4.0-2.27-dummy-dummy-gcc_--version.sh ..........   Passed    0.29 sec
      Start  5: GCC-5.4.0-2.27-dummy-dummy-gcc-ar_--version.sh
 5/10 Test  #5: GCC-5.4.0-2.27-dummy-dummy-gcc-ar_--version.sh .......   Passed    0.37 sec
      Start  6: GCC-5.4.0-2.27-dummy-dummy-gcc-nm_--version.sh
 6/10 Test  #6: GCC-5.4.0-2.27-dummy-dummy-gcc-nm_--version.sh .......   Passed    0.41 sec
      Start  7: GCC-5.4.0-2.27-dummy-dummy-gcc-ranlib_--version.sh
 7/10 Test  #7: GCC-5.4.0-2.27-dummy-dummy-gcc-ranlib_--version.sh ...   Passed    0.25 sec
      Start  8: GCC-5.4.0-2.27-dummy-dummy-gcov_--version.sh
 8/10 Test  #8: GCC-5.4.0-2.27-dummy-dummy-gcov_--version.sh .........   Passed    0.25 sec
      Start  9: GCC-5.4.0-2.27-dummy-dummy-gcov-tool_--version.sh
 9/10 Test  #9: GCC-5.4.0-2.27-dummy-dummy-gcov-tool_--version.sh ....   Passed    0.27 sec
      Start 10: GCC-5.4.0-2.27-dummy-dummy-gfortran_--version.sh
10/10 Test #10: GCC-5.4.0-2.27-dummy-dummy-gfortran_--version.sh .....   Passed    0.28 sec

100% tests passed, 0 tests failed out of 10
```

Challenges

- Design a complete build infrastructure with YAML configs to generate tests
- Creating and running tests that require GUI (X11 enabled)
- Manage large test repositories like R, Python, Perl to host tests for every package
- Add support for different test verification criteria
  - numerical difference
  - Creation of file upon execution
  - plotting graphs
  - Non zero exit status pass (?)
- Comprehensive logging and debugging feature
- Refactor code

Current Work

- Adding tests for R packages.
  - https://github.com/shahzebsiddiqui/R-buildtest-config/milestones
- Adding tests for Python
  - https://github.com/shahzebsiddiqui/Python-buildtest-config/milestones
- Adding tests for Perl
- Add support for Tcl, Lua and Ruby for buildtest
- Updating Documentation

References

- buildtest framework: https://github.com/shahzebsiddiqui/buildtest
- buildtest configs: https://github.com/shahzebsiddiqui/buildtest-configs
- R-buildtest-config: https://github.com/shahzebsiddiqui/R-buildtest-config
- Python-buildtest-config: https://github.com/shahzebsiddiqui/Python-buildtest-config
- Perl-buildtest-config: https://github.com/shahzebsiddiqui/Perl-buildtest-config
- Documentation: http://buildtestdocs.readthedocs.io/en/latest/