

FOSDEM'20

Buildtest: HPC Software Stack Testing Framework

Shahzeb Siddiqui (Shahzeb.Siddiqui@3ds.com)

Dassault Systemes

FOSDEM'20

02/02/2020

GitHub: <https://github.com/HPC-buildtest/buildtest-framework>

Documentation: <http://buildtest.rtf.d.io>

Motivation

- ▶ Framework Requirements:
 - ▶ The framework is capable of testing of installed software in HPC SW stack
 - ▶ The framework is able to integrate with module system
 - ▶ The framework provides users with a markup language for writing tests
 - ▶ The framework is able to automate test creation and execution
 - ▶ The framework provides contains a test repository that is community driven
- ▶ Buildtest is not meant to replace tools like `make`, `cmake`, or `autoconf`

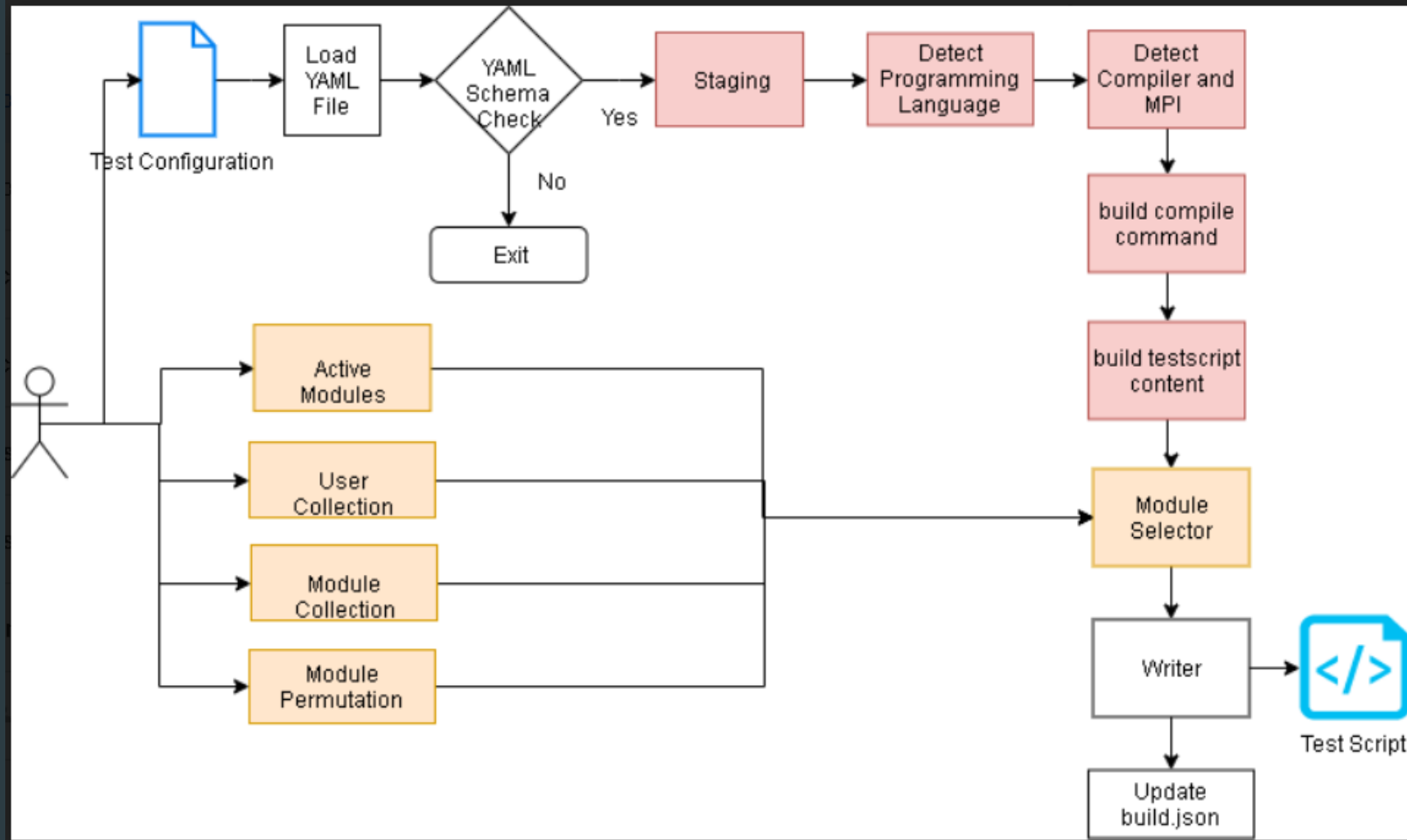
What is buildtest

- ▶ Buildtest is a framework that:
 - ▶ Automates test script creation
 - ▶ Abstracts test complexity by using test configuration written in YAML
 - ▶ Allows Portable test configurations
 - ▶ Provides many module operations
- ▶ Buildtest comes with a repository of test configuration and source files

GitHub: <https://github.com/HPC-buildtest/buildtest-framework>
Documentation: <http://buildtest.rtfid.io>

The screenshot shows the buildtest documentation website. The left sidebar contains a navigation menu with sections: BACKGROUND (Summary of buildtest, Feature Overview), REFERENCE (Installing Buildtest, Configuring buildtest, Building Test, Introspection Operation, Module Operation, Additional Features), DEVELOPMENT GUIDE (Contributing Guide, References), and a quiz section titled 'TAKE THE QUIZ!' with a description 'Beat Triplebyte's online coding quiz. Get offers from top companies. Skip resumes & recruiters.' and a note 'Sponsored - Ads served ethically'. The main content area has a header with 'Docs » buildtest' and 'Edit on GitHub'. Below the header is a 'Note' section titled 'Upcoming talks on buildtest' with two entries: 1. Conference: 5th Easybuild User Meeting (Location: Barcelono, Spain; Date: Jan 30th, 2020; Link: <https://github.com/easybuilders/easybuild/wiki/5th-EasyBuild-User-Meeting>) and 2. Conference: FOSDEM'20 (Location: Brussels, Belgium; Date: Feb 2nd, 2020; Link: https://fosdem.org/2020/schedule/track/hpc_big_data_and_data_science/). Below the note is a paragraph describing buildtest as a testing framework designed for HPC Software Stack Testing, compatible with Lmod module system, providing a set of YAML keys to write test configuration. It mentions that buildtest translates into complex test scripts, allowing users to focus on writing test configuration with minimal knowledge of the underlying system. It also states that test configuration are reusable between HPC sites with the goal of sharing tests between the HPC community. Below this paragraph is a link 'For more details on buildtest check [Summary of buildtest](#)'. At the bottom of the main content area, it says 'This documentation was last rebuild on Dec 31, 2019 and is intended for version 0.7.5.' The footer of the main content area has a 'Background' section with links to 'Summary of buildtest', 'Background', and 'Motivation'. The bottom of the sidebar has a 'Read the Docs' button and a version selector 'v: latest'.

Build Pipeline



Building a Test

- ▶ To build a test script just specify a test configuration to buildtest as follows:
`buildtest build -c <test-configuration>`
- ▶ The test configuration can be found under `$BUILDTEST_ROOT/toolkit/suite`
- ▶ Name of test configuration is formulated by replacing file separator (`/`) by a dot (`.`) so `tutorial/compilers/args.c.yml` → `tutorial.compilers.args.c.yml`
- ▶ Source code must be under `src` directory and test configuration must be named with extension `.yml`

```
$ tree toolkit/suite/
toolkit/suite/
├── benchmark
│   ├── osu
│   │   └── osu_test.yml
│   └── stream
│       ├── src
│       │   ├── mysecond.c
│       │   ├── stream.c
│       │   └── stream.c.yml
│       └── tutorial
│           ├── compilers
│           │   ├── args.c.yml
│           │   ├── hello.f.yml
│           │   ├── hello_lsf.yml
│           │   └── hello_slurm.yml
│           └── src
│               ├── args.c
│               ├── hello.c
│               ├── hello.cpp
│               └── hello.f90
├── cuda
│   ├── saxpy.c.yml
│   └── src
│       └── saxpy.c
├── mpi
│   ├── hello.c.yml
│   └── src
│       └── hello.c
├── openacc
│   ├── src
│   │   └── vecAdd.c
│   ├── vecAdd.c_pgi.yml
│   └── vecAdd.c.yml
└── openmp
    ├── clang_hello.c.yml
    ├── omp_hello.c.yml
    └── src
        └── omp_hello.c
```

Test Configuration

1 → `testtype: singlesource`
Informs buildtest this is a Single Source Compilation. Implemented as a Python Class

2 → `description: "C program that prints arguments passed to executable."`
Description of text. Limited to 80 chars

3 → `scheduler: local`
Run Test Locally

6 → `program:`
Start of Test Declaration

7 → `compiler: gnu`
Specify Compiler Name

8 → `source: args.c`
Source File to be compiled

9 → `env:`
Start of Environment Variable Declaration

12 → `pre_build: gcc --version`
Commands to run before and after compilation.

13 → `cflags: -Wall -g`
Passing flags to C compiler by setting CFLAGS variable

14 → `post_build: gcc -v`
Commands to run before and after execution.

15 → `pre_run: echo $SRCDIR $TESTDIR`
Passing Arguments to the Execution

16 → `exec_opts: hello world!`
List of Maintainers

```
1 testtype: singlesource
2 description: "C program that prints arguments passed to executable."
3 scheduler: local
4
5
6 program:
7 compiler: gnu
8 source: args.c
9 env:
10     FOO: BAR
11     X: 1
12 pre_build: gcc --version
13 cflags: -Wall -g
14 post_build: gcc -v
15 pre_run: echo $SRCDIR $TESTDIR
16 exec_opts: hello world!
17 post_run: echo post_run
18
19 maintainer:
20     - shahzeb siddiqui shahzebmsiddiqui@gmail.com
```

Intel Example

```
1 testtype: singlesource
2 description: Hello World Fortran example using GNU compiler
3 scheduler: local
4
5 program:
6   source: hello.f90
7   compiler: intel
8   fflags: -O2
9
10 maintainer:
11 - shahzeb siddiqui shahzebmsiddiqui@gmail.com
```

```
1 $ buildtest build -c tutorial.compilers.hello.f.yml -co intel --dry
2 Loading Test Configuration (YAML) file: /u/users/ssi29/gpfs/buildtest-framework/toolkit/suite/tutorial/compilers/hello.f.yml
3 Checking schema of YAML file
4 Schema Check Passed
5 Scheduler: local
6 Source Directory: /u/users/ssi29/gpfs/buildtest-framework/toolkit/suite/tutorial/compilers/src
7 Source File: hello.f90
8 Detecting Programming Language, Compiler and MPI wrapper
9 Programming Language: fortran
10 FC: ifort
11 FFLAGS: -O2
12 Test:/tmp/ssi29/buildtest/tests/Intel/Haswell/x86_64/rhel/7.6/build_0/hello.f.yml.0x28f38c1.sh
13 -----
14
15 module purge
16 module restore intel
17 TESTDIR=/tmp/ssi29/buildtest/tests/Intel/Haswell/x86_64/rhel/7.6/build_0
18 SRCDIR=/u/users/ssi29/gpfs/buildtest-framework/toolkit/suite/tutorial/compilers/src
19 SRCFILE=$SRCDIR/hello.f90
20 FC=ifort
21 FFLAGS="-O2"
22 EXECUTABLE=hello.f.yml.0xa7f9d0b4.exe
23
24 cd $TESTDIR
25 $FC $FFLAGS -o $EXECUTABLE $SRCFILE
26 $EXECUTABLE
27 rm ./EXECUTABLE
28 -----
```

Module Load Testing

```
$ buildtest module loadtest --login --numtest 5
RUN: 1 STATUS: PASSED - Testing module command: bash --login -c module purge; module load gmpi/2018a; ( File: /mxg-hpc/users/ssi29/easybuild/modules/all/gmpi/2018a.lua )
RUN: 2 STATUS: PASSED - Testing module command: bash --login -c module purge; module load numactl/2.0.11-GCCcore-6.4.0; ( File: /mxg-hpc/users/ssi29/easybuild/modules/all/numactl/2.0.11-GCCcore-6.4.0.lua )
RUN: 3 STATUS: PASSED - Testing module command: bash --login -c module purge; module load GCCcore/6.4.0; ( File: /mxg-hpc/users/ssi29/easybuild/modules/all/GCCcore/6.4.0.lua )
RUN: 4 STATUS: PASSED - Testing module command: bash --login -c module purge; module load GCCcore/7.4.0; ( File: /mxg-hpc/users/ssi29/easybuild/modules/all/GCCcore/7.4.0.lua )
RUN: 5 STATUS: PASSED - Testing module command: bash --login -c module purge; module load GCCcore/9.2.0; ( File: /mxg-hpc/users/ssi29/easybuild-HMNS/modules/all/Core/GCCcore/9.2.0.lua )

Writing Results to /tmp/ssi29/buildtest/tests/modules-load.out
Writing Results to /tmp/ssi29/buildtest/tests/modules-load.err

Module Load Summary
Module Trees: [' /mxg-hpc/users/ssi29/easybuild-HMNS/modules/all/Core', ' /mxg-hpc/users/ssi29/spack/modules/linux-rhel7-x86_64/Core', ' /mxg-hpc/users/ssi29/easybuild/modules/all/Core', ' /usr/share/lmod/lmod/modulefiles/Core']
PASSED: 5
FAILED: 0
```


Travis

- ▶ Since v0.7.4, buildtest can run its regression test in Travis. Several improvement to Travis configuration in v0.7.5
- ▶ Currently, buildtest contains approximately 30+ regression tests
- ▶ Some regression tests rely on having a software stack, so buildtest builds a mini stack using easybuild.
- ▶ Buildtest is tested for Python 3.6, 3.7, 3.8 and Lmod version 6.6.2 and 7.8.2

HPC-buildtest / buildtest-framework build unknown

Current Branches Build History Pull Requests More options

✓ **devel** moving regression test for buildtest module tree to test_mod. -o- #75 passed

Add some more regression tests.
Removed method menu() and parse_options()

⌚ Ran for 4 min 2 sec
⌚ Total time 10 min 42 sec

🔗 Commit dc0adf0
🔗 Compare 6639098..dc0adf0
🔗 Branch devel

👤 Shahzeb Siddiqui

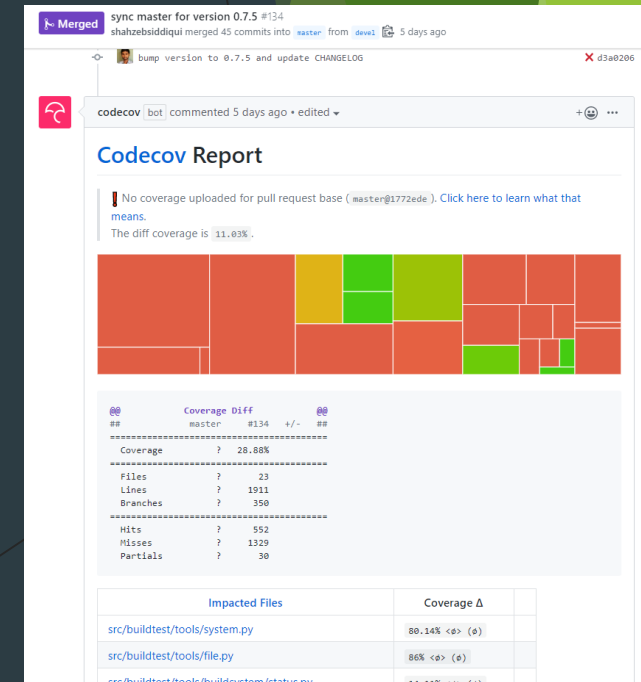
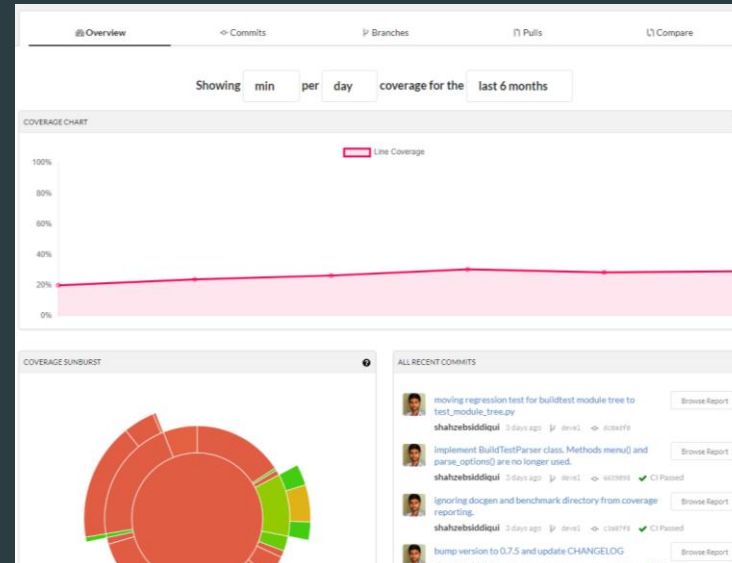
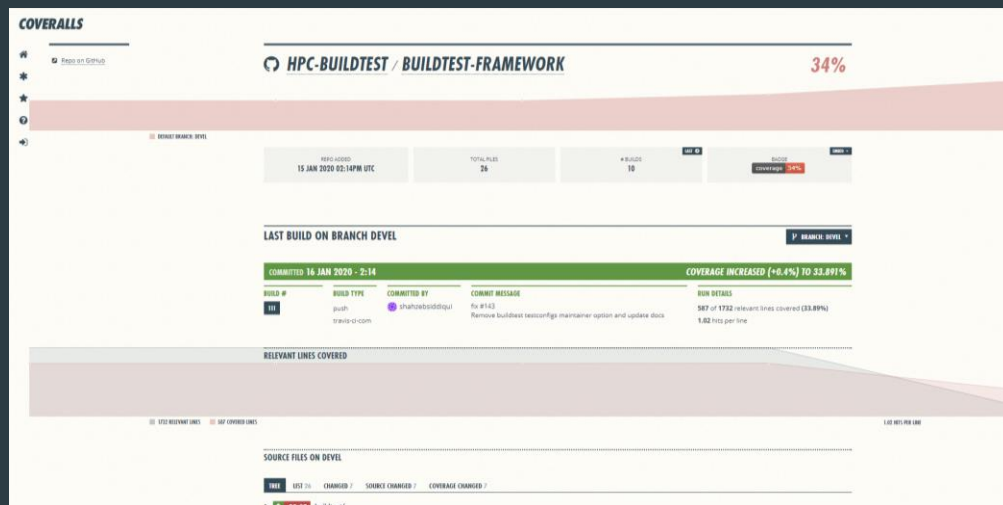
3 days ago

Build jobs View config

✓ # 75.1		</> Python: 3.6	📦 LMOD_VERSION=6.6.2	⌚ 1 min 45 sec
✓ # 75.2		</> Python: 3.7	📦 LMOD_VERSION=6.6.2	⌚ 1 min 44 sec
✓ # 75.3		</> Python: 3.8	📦 LMOD_VERSION=6.6.2	⌚ 1 min 46 sec
✓ # 75.4		</> Python: 3.6	📦 LMOD_VERSION=7.8.2	⌚ 1 min 47 sec
✓ # 75.5		</> Python: 3.7	📦 LMOD_VERSION=7.8.2	⌚ 1 min 43 sec
✓ # 75.6		</> Python: 3.8	📦 LMOD_VERSION=7.8.2	⌚ 1 min 57 sec

Coverage Report

- ▶ Since v0.7.5, buildtest can capture coverage report via codecov that is found at <https://codecov.io/gh/HPC-buildtest/buildtest-framework>
- ▶ Codecov report is automatically reported by codecov bot on pull requests
- ▶ Coveralls provides in-depth and more user-friendly coverage report like codecov



GitHub: <https://github.com/HPC-buildtest/buildtest-framework>
Documentation: <http://buildtest.rtfid.io>

GitHub Integration

- ▶ GitHub Apps Integration
 - ▶ CI: Travis
 - ▶ Code Quality: CodeCov, Coveralls, CodeFactor
 - ▶ Security: Snyk, GuardRails
- ▶ GitHub Bot Integration
 - ▶ Issue-Label Bot (<https://github.com/marketplace/issue-label-bot>)
 - ▶ Stale (<https://github.com/marketplace/stale>)
 - ▶ Trafico (<https://github.com/marketplace/trafico-pull-request-labeler>)
 - ▶ Pull-Request-Size (<https://github.com/marketplace/pull-request-size>)
- ▶ GitHub Action Integration
 - ▶ Black Code Formatter (<https://github.com/marketplace/actions/black-code-formatter>)
 - ▶ URLs-checker (<https://github.com/marketplace/actions/urls-checker>)

Future Work

- ▶ Current YAML schema has some limitation that do not address the following
 - ▶ Declaring variables in tests
 - ▶ Test permutation (compilation flags, multiple runs, environment variables, compilers)
 - ▶ Running test with a range of values (i.e running OpenMP program with range of threads `OMP_NUM_THREADS=[1-40]`)
 - ▶ Support for multiple source compilation
- ▶ Increase coverage report for regression tests

Reference

Slack Channel	https://hpcbuildtest.slack.com/
Join Slack via Heroku	https://hpcbuildtest.herokuapp.com/
Documentation	http://buildtest.readthedocs.io/
GitHub	https://github.com/HPC-buildtest/buildtest-framework
ReadTheDocs	https://readthedocs.org/projects/buildtest/
Codecov	https://codecov.io/gh/HPC-buildtest/buildtest-framework
Travis	https://travis-ci.com/HPC-buildtest/buildtest-framework
Coverall	https://coveralls.io/github/HPC-buildtest/buildtest-framework
CodeFactor	https://www.codefactor.io/repository/github/hpc-buildtest/buildtest-framework
Snyk	https://app.snyk.io/org/hpc-buildtest/
GuardRails	https://dashboard.guardrails.io/default/gh/HPC-buildtest