buildtest Documentation

Release 2019.21.01

Shahzeb Siddiqui

May 15, 2019

BACKGROUND

Note: This project is under development, the official release will be in 2020.

Welcome to buildtest is a framework for automating tests creation to conduct software stack testing for HPC facilities. buildtest aims to abstract test complexity such that a user can focus on writing test with minimal knowledge of the system. Test configuration are reusable between HPC sites with the goal of sharing tests between the HPC community.

For more details on buildtest check *Summary of buildtest*

This documentation was last rebuild on May 15, 2019 and is intended for version 0.6.3.

CHAPTER

ONE

SUMMARY OF BUILDTEST

Contents

1.1 What are we trying to solve?

When you ask your HPC facilities the following question

What type of testing are you doing against your software stack?

The most common response you get are the following

- **NONE**
- **I dont care, my users do the testing**.

Those that do any form of testing will write test scripts in-house with hard-coded file paths, module names, and site-specific configuration that cannot be **reusable** or shared between other HPC sites. Some may take one extra step and automate all test via CI tools (Jenkins, Travis, ctest) and may even use dashboard like CDASH for visualizing test results.

The mission of this project (*buildtest*) is to provide a mechanism to automate test creation of test scripts and the ability to *share* test scripts in a high level configuration format (YAML) that is easily interpreted

1.2 Motivation

A typical HPC facility supports hundreds of applications that is supported by the HPC team. Building these software is a challenge and then figuring out how this software stack behaves due to system changes (OS release, kernel path, glibc, etc. . .) is even more difficult.

Application Testing is difficult! Commercial and open-source application typically provide test scripts such as **make test** or **ctest** that can test the software after building (**make**) step. Unfortunately, these methods perform tests prior to

installation (`make install`) so the ability to test software in production is not possible. One could try to change the the vendor test script to the install path but this requires significant change into a complex makefile. Some software don't have any test scripts and you are on your own.

Writing test scripts manually can be tedious, also there is no sharing of tests and most likely they are not compatible to work with other HPC sites because of different software stack and hard-coded paths specific to the site. Easybuild and Spack are great tools for automating the installation of the entire software stack for an HPC system, buildtest is meant to complement these tools to test the software that is currently installed in your system.

1.3 How was buildtest started?

Our institution was going to move our HPC cluster to another Data Center (DC), and my task was to conduct software testing before and after the DC move. This project started out by writing individual test scripts to test specific features of a software. Most of the software testing was geared for `compilers`, `mpi`, `R`, `Python`, etc... Each test script could be run adhoc or via master script that would run everything. Originally buildtest was implemented in bash and then ported over to Python due to language limitation of bash.

1.4 Description

buildtest is a framework to automate testing for software stack in HPC sites. buildtest aims to abstract test complexity so the user can focus on writing test with minimal knowledge of the system. buildtest provides a rich set of YAML keys with *key*, *value* pairs to define test options that buildtest will parse and create a shell-script.

buildtest supports job submission to batch scheduler currently `LSF` and `SLURM`. buildtest assumes your software is installed and your facility has module environment Lmod so you can load the software environment.

1.5 buildtest features

0

- Provide a rich YAML API to write test configuration that is **reusable** and **site agnostic**
- Verify modulefiles can be loaded and conduct `module load` testing.
- Sanity check for binaries for application and system packages
- List software packages provided by `MODULEPATH`
- Support for logging
- Search for YAML and test scripts
- Summary of run output
- Support for multiple shells (csh, bash, sh)
- Generate job scripts (SLURM, LSF) for each test and automate job submission
- Support for benchmark

buildtest is available on Github at https://github.com/HPC-buildtest/buildtest-framework

CHAPTER

TWO

CONCEPTS

2.1 Modules

buildtest will detect modules installed in your system with the help of Lmod utility (`spider`). For background details on **spider** check the official documentation: https://lmod.readthedocs.io/en/latest/136_spider.html

buildtest will run the following command during the setup:

```
$LMOD_DIR/spider -o spider-json $BUILDTEST_MODULEPATH
```

The above output is not readable since it is in json so you can pipe this to the following:

```
$LMOD_DIR/spider -o spider-json $BUILDTEST_MODULEPATH | python -m json.tool
```

In buildtest we make use of json library to convert output to json. The output will look something like this:

```
"libffi": {
    "/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/libffi/3.2.1-GCCcore-
↪6.4.0.lua": {
        "path": "/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/libffi/3.
↪2.1-GCCcore-6.4.0.lua",
        "Description": "The libffi library provides a portable, high level␣
↪programming interface to\n various calling conventions. This allows a programmer to␣
↪call any function\n specified by a call interface description at run-time.\n",
        "name_lower": "libffi",
        "parent": [
            "default:eb/2018"
        ],
        "epoch": 1528138610,
        "full": "libffi/3.2.1-GCCcore-6.4.0",
        "full_lower": "libffi/3.2.1-gcccore-6.4.0",
        "name": "libffi",
        "help": "\nDescription\n===========\nThe libffi library provides a portable,␣
↪high level programming interface to\n various calling conventions. This allows a␣
↪programmer to call any function\n specified by a call interface description at run-
↪time.\n\n\nMore information\n================\n - Homepage: http://sourceware.org/
↪libffi/\n",
        "markedDefault": false,
        "whatis": [
            "Description: \n The libffi library provides a portable, high level␣
↪programming interface to\n various calling conventions. This allows a programmer to␣
↪call any function\n specified by a call interface description at run-time.\n",
            "Homepage: http://sourceware.org/libffi/"
        ]
```

(continues on next page)

(continued from previous page)


```
    },
    "/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/intel-CUDA/2017.1.132-GCC-5.4.
↪0-2.27-8.0.44/impi/2017.1.132/libffi/.3.2.1.lua": {
        "path": "/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/intel-CUDA/2017.1.
↪132-GCC-5.4.0-2.27-8.0.44/impi/2017.1.132/libffi/.3.2.1.lua",
        "Description": "The libffi library provides a portable, high level␣
↪programming interface to various calling\nconventions. This allows a programmer to␣
↪call any function specified by a call interface description at run-time.",
        "name_lower": "libffi",
        "parent": [
            "default:eb/2017:icc/.2017.1.132-GCC-5.4.0-2.27:CUDA/8.0.44:impi/2017.1.
↪132",
            "default:eb/2017:ifort/.2017.1.132-GCC-5.4.0-2.27:CUDA/8.0.44:impi/2017.1.
↪132",
            "default:icc/.2017.1.132-GCC-5.4.0-2.27:CUDA/8.0.44:impi/2017.1.132",
            "default:ifort/.2017.1.132-GCC-5.4.0-2.27:CUDA/8.0.44:impi/2017.1.132",
            "default:medsci:hpc/eb-2017-core:icc/.2017.1.132-GCC-5.4.0-2.27:CUDA/8.0.
↪44:impi/2017.1.132",
            "default:medsci:hpc/eb-2017-core:ifort/.2017.1.132-GCC-5.4.0-2.27:CUDA/8.
↪0.44:impi/2017.1.132"
        ],
        "epoch": 1505283586,
        "full": "libffi/.3.2.1",
        "full_lower": "libffi/.3.2.1",
        "name": "libffi",
        "help": "\nDescription\n===========\nThe libffi library provides a portable,␣
↪high level programming interface to various calling\nconventions. This allows a␣
↪programmer to call any function specified by a call interface description at run-
↪time.\n\n\nMore information\n================\n - Homepage: http://sourceware.org/
↪libffi/\n",
        "markedDefault": false,
        "whatis": [
            "Description: The libffi library provides a portable, high level␣
↪programming interface to various calling\nconventions. This allows a programmer to␣
↪call any function specified by a call interface description at run-time.",
            "Homepage: http://sourceware.org/libffi/"
        ]
    }
}
```


Note: Please note the output above is from Lmod 6, there are slight difference in the format in Lmod 7 that we will discuss later

In buildtest this is handled by class `BuildTestModule`. The spider output returns a dictionary that contains details of all modules based on MODULEPATH, along with full path to module files, and the metadata for a module. This captures all the details that you may get when running `module spider`.

2.1.1 Get Unique Software

To get a list of unique software you could run `module -t spider`:

```
$ module -t spider | head -n 10
20140726
```

(continues on next page)

(continued from previous page)

```
2015-workaround
20150316
2016-01-x64
3212u1
Advisor/
Advisor/2017_update1
Amber/
Amber/14-AmberTools-15-patchlevel-13-13
Anaconda2/
Anaconda2/4.2.0
Anaconda2/5.1.0
Anaconda3/
Anaconda3/4.2.0
Anaconda3/5.1.0
Aspera-Connect/
Aspera-Connect/3.6.1
```

Though `module -t spider` gives you the output it is not the best way to retrieve the result but rather use the `spider` utility. In buildtest you can get this by calling `BuildTestModules` class and invoke the method `get_unique_modules` as shown below

```
module = BuildTestModule()
module.get_unique_modules()
```

The method `get_unique_modules()` is returning the keys from the dictionary. It checks if abspath of module is in one of the module trees in `BUILDTEST_MODULEPATH` so that it retrieves unique module only defined by `BUILDTEST_MODULEPATH`. Typically, `spider` will retrieve all modules that may belong to other module trees and we dont want that.

```
def get_unique_modules(self):
  """Return a list of unique full name canonical modules """
  unique_modules_set = set()
  for module in self.module_dict.keys():
      for mpath in self.module_dict[module].keys():
          for tree in config_opts["BUILDTEST_MODULEPATH"]:
              if tree in mpath:
                  unique_modules_set.add(module)
                  break
  return sorted(list(unique_modules_set))
```

The above method is typically used by `buildtest list --software` to return a list of unique software.

2.1.2 Get Unique Module Versions

When users load modules (`module load GCC/5.4.0`) they are loading a specific software-version. Even when user does `module load GCC` without a version. Lmod will resolve to the default version even if user doesn't specify this.

```
module = BuildTestModule()
module.get_unique_fname_modules()
```

The method `get_unique_fname_modules()` returns a sorted list of module full name. Recall from the dictionary we are retrieving the keyword `full` from the dictionary

```
"/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/libffi/3.2.1-GCCcore-6.4.0.lua": {
        "path": "/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/libffi/3.2.1-GCCcore-6.4.0.lua",
        "Description": "The libffi library provides a portable, high level programming interface to\n various calling conventions. This allows a programmer to call any function\n specified by a call interface description at run-time.\n",
        "name_lower": "libffi",
        "parent": [
            "default:eb/2018"
        ],
        "epoch": 1528138610,
        "full": "libffi/3.2.1-GCCcore-6.4.0",
        "full_lower": "libffi/3.2.1-gcccore-6.4.0",
        "name": "libffi",
        "help": "\nDescription\n===========\nThe libffi library provides a portable, high level programming interface to\n various calling conventions. This allows a programmer to call any function\n specified by a call interface description at run-time.\n\n\nMore information\n================\n - Homepage: http://sourceware.org/libffi/\n",
        "markedDefault": false,
        "whatis": [
            "Description: \n The libffi library provides a portable, high level programming interface to\n various calling conventions. This allows a programmer to call any function\n specified by a call interface description at run-time.\n",
            "Homepage: http://sourceware.org/libffi/"
        ]
    },
```

The implementation of `get_unique_fname_modules()` is shown below.

```
def get_unique_fname_modules(self):
 """Return a list of unique canonical fullname of module
 where abspath to module is in one of the
 directories defined by BUILDTEST_MODULEPATH"""
 software_set = set()

 for module in self.get_unique_modules():
     for mpath in self.module_dict[module].keys():
         fname = ""
         if self.major_ver == 6:
             fname = self.module_dict[module][mpath]["full"]
         elif self.major_ver == 7:
             fname = self.module_dict[module][mpath]["fullName"]

         # only add module files that belong in directories specified
         #  by BUILDTEST_MODULEPATH.
         for tree in config_opts["BUILDTEST_MODULEPATH"]:
             if tree in mpath:
                 software_set.add(fname)
                 break

 return sorted(list(software_set))
```

Also note we make use of set to avoid duplicate entries and only add modules to set whose filepath is in `BUILDTEST_MODULEPATH`.

Note: Lmod 6 and 7 have some difference in the dictionary, just to name a few. The key `full` has been changed to `fullName` in Lmod 7. Here is an example dictionary format from Lmod 7

```
    "gompi": {
       "/gpfs/apps/easybuild/2019/SkyLake/redhat/7.5/modules/all/gompi/2018b.lua": {
           "pV": "000002018.*b.*zfinal",
           "Description": "GNU Compiler Collection (GCC) based compiler toolchain,\n including OpenMPI for MPI support.",
           "whatis": [
               "Description: GNU Compiler Collection (GCC) based compiler toolchain,\n including OpenMPI for MPI support.",
               "Homepage: (none)"
           ],
           "wV": "000002018.*b.*zfinal",
           "help": "\nDescription\n===========\nGNU Compiler Collection (GCC) based compiler toolchain,\n including OpenMPI for MPI support.\n\n\nMore information\n================\n - Homepag
e: (none)\n",
           "parentAA": [
               [
                   "eb/2019"
               ]
           ],
           "hidden": false,
           "Version": "2018b",
           "fullName": "gompi/2018b"
       }
   },
```

Due to this slight change, buildtest will check the Lmod version before checking for the full module name retrieved by key `full` in Lmod 6 or `fullName` in Lmod 7.

2.1.3 Module File Path

To retrieve the absolute path to a module file you can retrieve the inner keys. The dictionary is categorized by software and each key represents full path to module file.

The lines of interest are the following

```
"Autoconf": {
    "/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/Autoconf/2.69-GCCcore-6.4.0.lua": {
        <METADATA>
    },
    "/nfs/grid/software/RHEL7/easybuild/modules/all/Compiler/GCC/5.4.0-2.27/Autoconf/.2.69.lua": {
        <METADATA>
    }
}
 "Automake": {
    "/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/Automake/1.15.1-GCCcore-6.4.0.lua": {
        <METADATA>
    }
}
```

Implementation for `get_modulefile_path()` is described below.

```
def get_modulefile_path(self):
  """Return a list of absolute path for all module files"""
  module_path_list  = []
  for k in self.get_unique_modules():
      for tree in config_opts["BUILDTEST_MODULEPATH"]:
          for mpath in self.module_dict[k].keys():
              if tree in mpath:
                  module_path_list.append(mpath)

  return module_path_list
```

This method is used to return a list of modulefile paths in `BUILDTEST_MODULEPATH`.

2.1.4 Get Parent Modules

Parent modules are modules that need to be loaded first before loading the module of interest. In *Hiearchical Module Naming Scheme* you will have some modules that load another module tree (**MODULEPATH**) typically these are set in compilers, mpi, numlibs modules.

Luckily `spider` has way to retrieve parent modules for any module defined by the key `parent` in the json object.

```
"/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/5.4.0-2.27/OpenMPI/2.0.0/zlib/.1.2.8.lua": {
        "Description": "zlib is designed to be a free, general-purpose, legally unencumbered -- that is,\n not covered by any patents -- lossless data-compression library for use on virtually any\n computer hardware and operating system.",
        "epoch": 1506614076,
        "full": "zlib/.1.2.8",
        "full_lower": "zlib/.1.2.8",
        "help": "\nDescription\n===========\nzlib is designed to be a free, general-purpose, legally unencumbered -- that is,\n not covered by any patents -- lossless data-compression library for use on virtually any\n computer hardware and operating system.\n\n\nMore information\n================\n - Homepage: http://www.zlib.net/\n",
        "markedDefault": false,
        "name": "zlib",
        "name_lower": "zlib",
        "parent": [
            "default:eb/2017:GCC/5.4.0-2.27:OpenMPI/2.0.0",
            "default:medsci:hpc/eb-2017-core:GCC/5.4.0-2.27:OpenMPI/2.0.0"
        ],
        "path": "/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/5.4.0-2.27/OpenMPI/2.0.0/zlib/.1.2.8.lua",
        "whatis": [
            "Description: zlib is designed to be a free, general-purpose, legally unencumbered -- that is,\n not covered by any patents -- lossless data-compression library for use on virtually any\n computer hardware and operating system.",
            "Homepage: http://www.zlib.net/"
        ]
    },
```

Note: The output above is from Lmod 6 and `parent` key is one of those keys that has changed in Lmod 7 which will be discussed later

In this example, the module `zlib/.1.2.8` is in a Hierarchical Tree built by `GCC/5.4.0` and `OpenMPI/2.0.0`. The `parent` key is a list of different module combination that can be used to load this module.

Shown below is one way to load `zlib/.1.2.8` using the first combination of parent modules.

```
buildtest-framework[master !?] $ ml
No modules loaded
buildtest-framework[master !?] $ ml eb/2017 GCC/5.4.0-2.27 OpenMPI/2.0.0 zlib/.1.2.8
buildtest-framework[master !?] $ ml

Currently Loaded Modules:
  1) eb/2017          3) binutils/.2.27   5) numactl/2.0.11   7) OpenMPI/2.0.0
↪          9) FFTW/3.3.4                                  11) zlib/.1.2.8
  2) GCCcore/.5.4.0   4) GCC/5.4.0-2.27   6) hwloc/1.11.3     8) OpenBLAS/0.2.19-
↪LAPACK-3.6.0  10) ScaLAPACK/2.0.2-OpenBLAS-0.2.19-LAPACK-3.6.0
```

We can confirm this by running the second parent combination to load `zlib/.1.2.8`

```
(siddis14-TgVBs13r) docs[master !?] $ ml
No modules loaded
(siddis14-TgVBs13r) docs[master !?] $ ml medsci hpc/eb-2017-core GCC/5.4.0-2.27
↪OpenMPI/2.0.0 zlib/.1.2.8
(siddis14-TgVBs13r) docs[master !?] $ ml

Currently Loaded Modules:
  1) medsci             3) GCCcore/.5.4.0   5) GCC/5.4.0-2.27   7) hwloc/1.11.3    9)
↪OpenBLAS/0.2.19-LAPACK-3.6.0  11) ScaLAPACK/2.0.2-OpenBLAS-0.2.19-LAPACK-3.6.0
  2) hpc/eb-2017-core   4) binutils/.2.27   6) numactl/2.0.11   8) OpenMPI/2.0.0  10)
↪FFTW/3.3.4                    12) zlib/.1.2.8
```

Recall in Lmod 6, `parent` is a list with modules separated by colon separator (`:`) and each entry starts with word `default`.

In Lmod 7 the parent key is renamed to `parentAA` see below

```
1      "gompi": {
2         "/gpfs/apps/easybuild/2019/SkyLake/redhat/7.5/modules/all/gompi/2018b.lua": {
3             "pV": "000002018.*b.*zfinal",
4             "Description": "GNU Compiler Collection (GCC) based compiler toolchain,\n
  ↪including OpenMPI for MPI support.",
5             "whatis": [
6                 "Description: GNU Compiler Collection (GCC) based compiler toolchain,
  ↪\n including OpenMPI for MPI support.",
7                 "Homepage: (none)"
8             ],
9             "wV": "000002018.*b.*zfinal",
10            "help": "\nDescription\n===========\nGNU Compiler Collection (GCC) based
  ↪compiler toolchain,\n including OpenMPI for MPI support.\n\n\nMore
  ↪information\n================\n - Homepag
11 e: (none)\n",
12            "parentAA": [
13                [
14                    "eb/2019"
15                ]
16            ],
17            "hidden": false,
18            "Version": "2018b",
19            "fullName": "gompi/2018b"
```

(continues on next page)

(continued from previous page)

```
        }
    },
```

The `parentAA` is now a **list of list** where each list corresponds to a set of parent modules to be loaded before loading actual module.

In buildtest we can get the parent for any module with the following code

```
module_name = "GCC/5.4.0-2.27"
module = BuildTestModule
parent_module = module.get_parent_modules(module_name)
```

The method `get_parent_modules` returns a list of modules to be loaded for the specified module. In the implementation we only get the first parent combination of modules.

The implementation for `get_parent_modules` can be shown below

```
def get_parent_modules(self,modname):
   """Get Parent module for specified module file."""
   for key in self.module_dict.keys():
       for mod_file in self.module_dict[key].keys():
           mod_full_name = parent_mod_name = ""

           if self.major_ver == 6:
               mod_full_name = self.module_dict[key][mod_file]["full"]
           elif self.major_ver == 7:
               mod_full_name = self.module_dict[key][mod_file]["fullName"]

           if modname == mod_full_name:
               if self.major_ver == 6:
                   parent_mod_name = self.module_dict[key][mod_file]["parent"]
               elif self.major_ver == 7:
                   # for modules that dont have any parent the dictionary
                   # does not declare parentAA key in Lmod 7. in that
                   # case return empty list
                   if "parentAA" not in self.module_dict[key][mod_file]:
                       parent_mod_name = []
                   # otherwise retrieve first index from parentAA.
                   # ParentAA is a list of list
                   else:
                       parent_mod_name = self.module_dict[key][mod_file]["parentAA
↪"][0]

                   return parent_mod_name

               mod_parent_list = parent_mod_name
               parent_module = []
               # parent: is a list, only care about one entry which
               # contain list of modules to be loaded separated by :
               # First entry is default:<mod1>:<mod2> so skip first
               # element
               for entry in mod_parent_list[0].split(":")[1:]:
                   parent_module.append(entry)

               return parent_module

   return []
```

2.2 How does buildtest leverage modules

buildtest will inject modules when writing test script. When you build a test from a configuration file you can load modules into your test script. See *Testing with modules* for more details.

For instance, running a binary test such as the utility `ompi_info` from OpenMPI can be done by loading the openmpi module and running the binary test via `buildtest build --binary` or set `BUILDTEST_BINARY=True`.

Below is a list of modules when loading openmpi

```
(siddis14-TgVBs13r) buildtest-framework[master !?+] $ ml

Currently Loaded Modules:
  1) eb/2018          3) binutils/2.28-GCCcore-6.4.0   5) zlib/1.2.11-GCCcore-6.4.0
↪   7) hwloc/1.11.8-GCCcore-6.4.0
  2) GCCcore/6.4.0    4) GCC/6.4.0-2.28                6) numactl/2.0.11-GCCcore-6.4.0
↪   8) OpenMPI/3.0.0-GCC-6.4.0-2.28
```

Let's run the binary test, buildtest will attempt to test every module.

```
(siddis14-TgVBs13r) buildtest-framework[master !?+] $ buildtest build -b
Detecting Software:eb/2018
No $PATH set in your module  eb/2018   so no possible binaries can be found
There are no binaries for package: eb/2018
Detecting Software:GCCcore/6.4.0
Generating  19  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/GCCcore/6.4.0
Detecting Software:binutils/2.28-GCCcore-6.4.0
Generating  18  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/binutils/2.28-GCCcore-
↪6.4.0
Detecting Software:GCC/6.4.0-2.28
No $PATH set in your module  GCC/6.4.0-2.28   so no possible binaries can be found
There are no binaries for package: GCC/6.4.0-2.28
Detecting Software:zlib/1.2.11-GCCcore-6.4.0
No $PATH set in your module  zlib/1.2.11-GCCcore-6.4.0   so no possible binaries can
↪be found
There are no binaries for package: zlib/1.2.11-GCCcore-6.4.0
Detecting Software:numactl/2.0.11-GCCcore-6.4.0
Generating  6  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/numactl/2.0.11-GCCcore-
↪6.4.0
Detecting Software:hwloc/1.11.8-GCCcore-6.4.0
Generating  15  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/hwloc/1.11.8-GCCcore-6.
↪4.0
Detecting Software:OpenMPI/3.0.0-GCC-6.4.0-2.28
Generating  11  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/OpenMPI/3.0.0-GCC-6.4.
↪0-2.28
```

The test for `ompi_info` is written with the appropriate module.

```
$ cat /home/siddis14/buildtest/software/OpenMPI/3.0.0-GCC-6.4.0-2.28/ompi_info.sh
 #!/bin/sh
```

(continues on next page)

(continued from previous page)

```
module load OpenMPI/3.0.0-GCC-6.4.0-2.28
which ompi_info
```

CHAPTER

THREE

FEATURE OVERVIEW

3.1 Listing Software and Modules

buildtest can report unique software and module versions found in your module system based on `spider` utility provided by Lmod. This feature can be useful for HPC facilities to see a count of all software packages and see what versions are installed for each package along with the filepath to module file.

buildtest collects this information via `BUILDTEST_MODULEPATH` which is equivalent to `MODULEPATH` but can be tweaked by buildtest to add & remove directory at will.

Shown below is unique software list using `buildtest list --software`

```
buildtest list --software
RHEL6-apps
cctsoft
clinpharm
deprecated
easybuild
eb
ljmedchemsoft
lmod
medsci
omics
pharmsci
ru
settarg
siterestricted
statistics
use.own.eb


Total Software Packages:  16
```

Similarly we can retrieve full name of module file and absolute path to module file using `buildtest list --module` or short option `buildtest list -m`

```
buildtest list --modules

    Full Module Name                       |      ModuleFile Path
-------------------------------------------|----------------------------
[32mRHEL6-apps                                 |        /nfs/grid/software/
↪moduledomains/RHEL6-apps.lua[0m
cctsoft                                    |        /nfs/grid/software/moduledomains/
↪cctsoft
```

(continues on next page)

(continued from previous page)

```
clinpharm                                  |        /nfs/grid/software/moduledomains/
↪clinpharm
deprecated                                 |        /nfs/grid/software/moduledomains/
↪deprecated
easybuild                                  |        /nfs/grid/software/moduledomains/
↪easybuild
[32meb/2017                                    |        /nfs/grid/software/
↪moduledomains/eb/2017.lua[0m
[32meb/2018                                    |        /nfs/grid/software/
↪moduledomains/eb/2018.lua[0m
ljmedchemsoft                              |        /nfs/grid/software/moduledomains/
↪ljmedchemsoft
[32mlmod/6.5.1                                 |        /usr/share/lmod/lmod/
↪modulefiles/Core/lmod/6.5.1.lua[0m
medsci                                     |        /nfs/grid/software/moduledomains/
↪medsci
omics                                      |        /nfs/grid/software/moduledomains/
↪omics
pharmsci                                   |        /nfs/grid/software/moduledomains/
↪pharmsci
ru                                         |        /nfs/grid/software/moduledomains/ru
[32msettarg/6.5.1                              |        /usr/share/lmod/lmod/
↪modulefiles/Core/settarg/6.5.1.lua[0m
siterestricted                             |        /nfs/grid/software/moduledomains/
↪siterestricted
statistics                                 |        /nfs/grid/software/moduledomains/
↪statistics
use.own.eb/append                          |        /nfs/grid/software/moduledomains/
↪use.own.eb/append
use.own.eb/prepend                         |        /nfs/grid/software/moduledomains/
↪use.own.eb/prepend


Total Software Modules: 18
[32mTotal LUA Modules: 5[0m
Total non LUA Modules: 13
```

3.2 Module Testing

HPC sites may have hundreds if not thousand module files, it would be great to test all of them. buildtest can conduct `module load` testing on module files and report `SUCCESS` or `FAIL` upon module load by checking exit status.

```
buildtest module loadtest
module load RHEL6-apps
RUN: 1/18 STATUS: PASSED - Testing module: RHEL6-apps
_____________________________________________________________________________
module load cctsoft
RUN: 2/18 STATUS: PASSED - Testing module: cctsoft
_____________________________________________________________________________
module load clinpharm
RUN: 3/18 STATUS: PASSED - Testing module: clinpharm
_____________________________________________________________________________
module load deprecated
RUN: 4/18 STATUS: PASSED - Testing module: deprecated
```

(continues on next page)

(continued from previous page)

```
_____________________________________________________________________________
module load easybuild
RUN: 5/18 STATUS: PASSED - Testing module: easybuild
_____________________________________________________________________________
module load eb/2017
RUN: 6/18 STATUS: PASSED - Testing module: eb/2017
_____________________________________________________________________________
module load eb/2018
RUN: 7/18 STATUS: PASSED - Testing module: eb/2018
_____________________________________________________________________________
module load ljmedchemsoft
RUN: 8/18 STATUS: PASSED - Testing module: ljmedchemsoft
_____________________________________________________________________________
module load lmod/6.5.1
RUN: 9/18 STATUS: PASSED - Testing module: lmod/6.5.1
_____________________________________________________________________________
module load medsci
RUN: 10/18 STATUS: PASSED - Testing module: medsci
_____________________________________________________________________________
module load omics
RUN: 11/18 STATUS: PASSED - Testing module: omics
_____________________________________________________________________________
module load pharmsci
RUN: 12/18 STATUS: PASSED - Testing module: pharmsci
_____________________________________________________________________________
module load ru
RUN: 13/18 STATUS: PASSED - Testing module: ru
_____________________________________________________________________________
module load settarg/6.5.1
RUN: 14/18 STATUS: PASSED - Testing module: settarg/6.5.1
_____________________________________________________________________________
module load siterestricted
RUN: 15/18 STATUS: PASSED - Testing module: siterestricted
_____________________________________________________________________________
module load statistics
RUN: 16/18 STATUS: PASSED - Testing module: statistics
_____________________________________________________________________________
module load use.own.eb/append
RUN: 17/18 STATUS: PASSED - Testing module: use.own.eb/append
_____________________________________________________________________________
module load use.own.eb/prepend
RUN: 18/18 STATUS: PASSED - Testing module: use.own.eb/prepend
_____________________________________________________________________________
Writing Results to /tmp/modules-load.out
Writing Results to /tmp/modules-load.err
_____________________________________________________________________________
                     Module Load Summary
Module Trees:                              ['/nfs/grid/software/moduledomains', '/etc/
↪modulefiles', '/usr/share/modulefiles', '/usr/share/lmod/lmod/modulefiles/Core']
PASSED:                                    18
FAILED:                                    0
_____________________________________________________________________________
```

3.3 Building Test

To build a test, buildtest requires an input configuration file that can be specified by option `-c` or long option `--config`. This option is part of `buildtest build`

Shown below is an example build

```
buildtest build -c /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml -vv
_____________________________________________________________________________
compiler: gnu
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.cpp
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.cpp exists!
Programming Language Detected: c++
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.
↪sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_gnu.yml.sh
_____________________________________________________________________________
#!/bin/sh
module purge
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
g++ -O3 -o hello.cpp.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.cpp
./hello.cpp.exe
 rm ./hello.cpp.exe
_____________________________________________________________________________
```

buildtest can insert modules into test, just load the modules before you build the test and it will insert them into your test script.

```
ml eb/2018; ml GCC; buildtest build -c /home/siddis14/buildtest-framework/toolkit/
↪buildtest/suite/compilers/helloworld/hello_gnu.yml -vv
_____________________________________________________________________________
compiler: gnu
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.cpp
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.cpp exists!
Programming Language Detected: c++
```

(continues on next page)

(continued from previous page)

```
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.
↪sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_gnu.yml.sh
_____________________________________________________________________________
#!/bin/sh
module purge
module load eb/2018
module load GCCcore/6.4.0
module load binutils/2.28-GCCcore-6.4.0
module load GCC/6.4.0-2.28
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
g++ -O3 -o hello.cpp.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.cpp
./hello.cpp.exe
 rm ./hello.cpp.exe
_____________________________________________________________________________
```

3.4 System Package Test

buildtest can generate tests for system packages using the option `buildtest build --package <package>`. Currently, system package test only perform sanity check against binaries found in the system. The framework will automatically generate binary test by checking the system default paths i.e `/usr/bin`, `/usr/local/bin`, `/usr/sbin`.

For instance to build test for the system package `coreutils` you can do the following

```
buildtest build --package coreutils
Detecting System Package: coreutils
Generating  102  binary tests
Binary Tests are written in  /tmp/siddis14/buildtest/tests/system/coreutils
Writing Log file to:  /tmp/siddis14/buildtest/logs/system/coreutils/buildtest_08_29_
↪14_05_2019.log
```

3.5 Running The Test

You can run the in several ways. The easiest way to run the test is via buildtest using `buildtest run -S <suite>`

Here is the output of the following test

```
buildtest run -S openmp
Running All Tests from Test Directory: /tmp/siddis14/buildtest/tests/suite/openmp
================================================================
                          Test summary
Package:  openmp
Executed 14 tests
Passed Tests: 14 Percentage: 100.0%
Failed Tests: 0 Percentage: 0.0%
SUCCESS: Threshold of 100.0% was achieved
Writing results to /tmp/siddis14/buildtest/run/buildtest_08_29_14_05_2019.run
```

3.6 TAB Completion

buildtest use the `argcomplete` python module to autocomplete buildtest argument. Just press TAB key on the keyboard to fill in the arguments.

For instance if you just type `buildtest` followed by TAB you should see the following.

```
$ buildtest
benchmark     --clean-logs  -h            list          module        --scantest    --
↪show-keys    -V            yaml
build         find          --help        --logdir      run           --show        --
↪submitjob    --version
```

Note: You will need to press the TAB key few times before it shows all the arguments

3.7 Log files

All buildtest logs will be written in `BUILDTEST_LOGDIR`.

buildtest will store log files for `buildtest build -s <app_name>/<app_ver>` in `BUILDTEST_LOGDIR/<app_name>/<app_ver>`. If toolchain option is specified for instance `buildtest build -s <app_name>/<app_ver> -t <tc_name>/<tc_ver>` then buildtest will store the logs in `BUILDTEST_LOGDIR/<app_name>/<app_ver>/<tc_name>/<tc_ver>`.

Similarly logs for system tests like `buildtest --package <package>` will be stored in `BUILDTEST_LOGDIR/system/<package>`

You may override BUILDTEST_LOGDIR option at command line via `buildtest --logdir` and you may even store individual buildtest runs in separate directories such as the following

```
buildtest build -s OpenMPI/3.0.0-GCC-6.4.0-2.28 --logdir=/tmp
```

CHAPTER

FOUR

SETUP

Contents

4.1 Requirements

buildtest is supported on Redhat or Centos

You need the following packages to get started.

- Python > 3.6 or higher
- Lmod `yum install Lmod`

If you want to build Lmod see the following links

- http://lmod.readthedocs.io/en/latest/030_installing.html

4.2 Installing buildtest

To get started clone the buildtest repos in your filesystem:

```
$ git clone git@github.com:HPC-buildtest/buildtest-framework.git
```

Once you clone the repos you will want to install the python dependencies for buildtest which can be done by running

```
$ pip install docs/requirements.txt
```

The requirements.txt can be installed via pip in your python environment (virtualenv, conda or pipenv .

To configure buildtest source the file `sourceme.sh`:

```
$ source sourceme.sh
```

4.3 Setting up auto-complete on buildtest arguments

Before you start using buildtest you may want to setup autocomplete feature in your shell by running

```
eval "$(register-python-argcomplete buildtest)"
```

This command works for `bash` or `sh` shell, if you are using `tcsh` you can run

```
eval `register-python-argcomplete --shell tcsh buildtest`
```

For more details on argcomplete please visit https://pypi.org/project/argcomplete/

Note: It is highly recommended to setup auto-complete feature when using buildtest to make use of tab completion

4.4 buildtest version (`buildtest -V`)

You can check the current version of buildtest by running the following:

```
$ buildtest -V
buildtest version:  0.6.3
```

CHAPTER

FIVE

CONFIGURING BUILDTEST

Contents

5.1 Configuration File

To configure buildtest you will need to create a YAML file at `$HOME/.buildtest/settings.yml`. This file is responsible for configuring buildtest to work for your test system. Shown below is the configuration file that can be found in the git repo.

```
# Delete Build Directory before writing tests.  Valid Options: [True, False]
BUILDTEST_CLEAN_BUILD: False

# binary test support. Valid options [True, False]
BUILDTEST_BINARY: False

# Configure shell when running test.
# Valid options:  [bash, csh, sh].
# Default: [sh]
BUILDTEST_SHELL: sh

# list root of module tree where modules are installed. Specify multiple module tree␣
↪as a list
BUILDTEST_MODULEPATH: []
```

(continues on next page)

(continued from previous page)

```
# control output of Lmod spider.
# Valid values are
# current: retrieve modules whose abspath is a subdir of directories defined in␣
↪BUILDTEST_MODULEPATH
# all: retrieve all records
BUILDTEST_SPIDER_VIEW: current

# control how parent modules are retrieved.
# Valid option: first, all
BUILDTEST_PARENT_MODULE_SEARCH: first

# criteria for success threshold when running job.
# Valid Option: [0.0-1.0]
BUILDTEST_SUCCESS_THRESHOLD: 1.0

# directory where buildtest logs will be written.
BUILDTEST_LOGDIR: /tmp/$USER/buildtest/logs

# directory where test scripts will be generated
BUILDTEST_TESTDIR: /tmp/$USER/buildtest/tests

# directory where buildtest will write .run files
BUILDTEST_RUN_DIR: /tmp/$USER/buildtest/run
```

5.2 Variable Description

Variables	Description	Values
BUILDTEST_CLEAN_BUILD	deletes test directory before writing test scripts	`True`, `False`
BUILDTEST_LOGDIR	directory where buildtest logs will be written	`<directory>`
BUILDTEST_TESTDIR	directory where test scripts will be generated	`<directory>`
BUILDTEST_MODULEPATH	colon separated list of root of module trees	`<directory>`
BUILDTEST_SHELL	shell type to use when generating tests	`shell`, `csh`, `bash`
BUILDTEST_SPIDER_VIEW	Control view of Lmod spider output	`current`, `all`
BUILDTEST_RUN_DIR	directory where buildtest will write *.run* files	`<directory>`
BUILDTEST_SUCCESS_THRESHOLD	criteria for success threshold when running test	`[0.0-1.0]`
BUILDTEST_BINARY	conduct sanity check for binary commands	`True`, `False`

5.3 Configuring Module Trees

BUILDTEST_MODULEPATH takes colon separated list of root of a module tree in your system that serves as module files. buildtest will read all module files and use this to figure out what modules can be tested.

Let's assume `/opt/apps` and `/workspace/apps` are root of the module tree, so we can specify this in your configuration as follows:

```
BUILDTEST_MODULEPATH:
- /opt/apps
- /workspace/apps
```

If you set an invalid directory path in `BUILDTEST_MODULEPATH` you will get the following message

Error: /opt/apps directory does not exist, specified in BUILDTEST_MODULEPATH

If you don't specify a module tree for BUILDTEST_MODULEPATH then buildtest will read the value of MODULEPATH. You may add,remove or list module tree.

To see the list of module tree you can run `buildtest module -l`:

```
$ buildtest module -l
/nfs/grid/software/moduledomains
/etc/modulefiles
/usr/share/modulefiles
/usr/share/lmod/lmod/modulefiles/Core
```

At this time you will notice BUILDTEST_MODULEPATH is not set and it takes value of MODULEPATH:

```
$ cat ~/.buildtest/settings.yml  | grep -i BUILDTEST_MODULEPATH
BUILDTEST_MODULEPATH: []

$ echo $MODULEPATH
/nfs/grid/software/moduledomains:/etc/modulefiles:/usr/share/modulefiles:/usr/share/
↪modulefiles/Linux:/usr/share/modulefiles/Core:/usr/share/lmod/lmod/modulefiles/Core
```

You can add new module tree through command line using `buildtest module -a` which will update the configuration file:

```
$ buildtest module -a /usr/share/lmod/lmod/modulefiles/Core
Adding module tree: /usr/share/lmod/lmod/modulefiles/Core
Configuration File: /home/siddis14/.buildtest/settings.yml has been updated
```

Similarly you can remove module tree from your configuration via `buildtest module -r`:

```
(siddis14-TgVBs13r) buildtest-framework[master !?] $ buildtest module -r /etc/
↪modulefiles
Removing module tree: /etc/modulefiles
Configuration File: /home/siddis14/.buildtest/settings.yml has been updated
```

5.4 Configure Spider View

Lmod `spider` retrieves module details in json format, buildtest is running `spider -o spider-json $BUILDTEST_MODULEPATH` to get all the modules. The configuration `BUILDTEST_SPIDER_VIEW` can control the output. When `BUILDTEST_SPIDER_VIEW=all` then buildtest will retrieve all records including records for modules that are not part of current `MODULEPATH`.

If you want to restrict the search of module retrieval to those defined in `BUILDTEST_MODULEPATH` then set `BUILDTEST_SPIDER_VIEW=current`. buildtest will only retrieve records whose modulefile absolute path is a subdirectory of `BUILDTEST_MODULEPATH`. When `BUILDTEST_MODULEPATH` is not set, it will take the value of `MODULEPATH` and setting `BUILDTEST_SPIDER_VIEW=current` can be useful in testing modules that are visible to module environment.

5.5 Configure Shell

buildtest supports test creation for `sh`, `bash`, and `csh`. The test are created with the appropriate extension. The default shell is `sh`.

To configure the shell use the variable `BUILDTEST_SHELL` in your configuration file:

```
BUILDTEST_SHELL: sh
```

To change the shell to `bash` or `csh` you can do either:

```
BUILDTEST_SHELL: bash
BUILDTEST_SHELL: csh
```

If you specify an invalid value you may get the following message

Error: BUILDTEST_SHELL expects value [`sh`, `bash`, `csh`] current value is tcsh

5.6 Clean Build

buildtest will write test in directory specified by `BUILDTEST_TESTDIR`. Often times, you may want to preserve tests across subsequent builds.

For instance you may be interested in building test for different shell and preserve all tests during the previous builds, this can be done by setting `export BUILDTEST_CLEAN_BUILD=False` or set in configuration as follows:

```
BUILDTEST_CLEAN_BUILD: False
```

Setting `BUILDTEST_CLEAN_BUILD` to `False` instructs buildtest to preserve build directory where test are written. This will allow user to keep test if they ran the following:

```
$ buildtest build -p gcc  --shell sh
$ buildtest build -p gcc --shell csh
$ buildtest build -p gcc --shell bash
```

If you want buildtest to delete the build directory before writing any tests you can set `export BUILDTEST_CLEAN_BUILD=True` or set the following in configuration file:

```
BUILDTEST_CLEAN_BUILD: True
```

5.7 Configure Test Directory

buildtest will write test scripts in `BUILDTEST_TESTDIR`. This can be specified in configuration file or environment variable.

For instance you may want to set the testing directory to `$HOME/tmp` which can be done in configuration:

```
BUILDTEST_TESTDIR: "${HOME}/tmp/"
```

This can be done via environment variable as:

```
export BUILDTEST_TESTDIR=$HOME/tmp
```

If the directory does not exist, buildtest will create it assuming you have the appropriate permissions.

You may specify this at the command line via `buildtest build --testdir`

See example below where we customize testdir at runtime using the command line option

```
buildtest build --package gcc --testdir /home/siddis14/tmp
Detecting System Package: gcc
Generating  8  binary tests
Binary Tests are written in  /home/siddis14/tmp/system/gcc
Writing Log file to:  /tmp/siddis14/buildtest/logs/system/gcc/buildtest_08_29_14_05_
↪2019.log
```

5.8 Log Directory

buildtest will write the logs specified by `BUILDTEST_LOGDIR` this can be set at the configuration file or environment variable.

Setting this at the configuration file can be done as follows:

```
BUILDTEST_LOGDIR: /tmp/buildtest/logs
```

Once this is set you will find a log file per execution. For instance, if you build tests for `GCCcore/6.4.0`, you will find the logs in the following path

```
/tmp/buildtest/logs/GCCcore/6.4.0/
```

All log files are named as follows `buildtest_HH_MM_DD_MM_YYYY.log` to preserve the date and time stamp.

Here is an example for a buildtest log file from Jan 20th 2019:

```
$ ls -l /tmp/buildtest/logs/GCCcore/6.4.0/buildtest_21_20_20_01_2019.log
-rw-r--r-- 1 siddis14 amer 81358 Jan 20 21:20 /tmp/buildtest/logs/GCCcore/6.4.0/
↪buildtest_21_20_20_01_2019.log
```

5.9 Test Threshold

buildtest provides a mechanism to set a success threshold during test execution that can be used to determine if your software passes or fails.

This can be set by using `BUILDTEST_SUCCESS_THRESHOLD` which is a value between `[0.0-1.0]` which will be used when running test.

```
if success_threshold >= <passed tests>/< total tests>
    SUCCESS
else
    FAIL
```

Here is an example test run where all test have passed and success threshold is 1.0

```
$ buildtest run -s GCCcore/6.4.0
Check Configuration
============================================================
                         Test summary
Application:  GCCcore/6.4.0
Executed 32 tests
Passed Tests: 32    Percentage: 100.0%
Failed Tests: 0    Percentage: 0.0%
SUCCESS: Threshold of 100.0% was achieved
Writing results to /tmp/buildtest_10_26_30_01_2019.run
```

CHAPTER

SIX

INTROSPECTION OPERATION

Contents

6.1 List Options (`buildtest list --help`)

```
buildtest list --help
usage: buildtest [options] [COMMANDS] list [-h] [-s] [-m] [-ec]

optional arguments:
  -h, --help          show this help message and exit
  -s, --software      get unique software from Lmod spider command
  -m, --modules       get full module name and path to module files
  -ec, --easyconfigs  Return a list of easyconfigs from a module tree
```

buildtest comes with a set of options for listing useful info such as

- List Unique software
- List software-modulefile relationship
- List of easyconfigs

6.1.1 List Software (`buildtest list --software`)

buildtest can report the software list by running the following `buildtest list --software` or short option `buildtest list -s`

buildtest determines the software list based on the module trees specified in `BUILDTEST_MODULEPATH` and processes each module tree and returns a unique software list

```
buildtest list --software
RHEL6-apps
cctsoft
clinpharm
deprecated
easybuild
eb
ljmedchemsoft
lmod
medsci
omics
pharmsci
ru
settarg
siterestricted
statistics
use.own.eb


Total Software Packages:  16
```

6.1.2 Listing Modules (`buildtest list --modules`)

If you want to view a breakdown of all modules then use `buildtest list --modules` or short option `buildtest list -m`

The output will be sorted by software and each entry will correspond to the full path of the modulefile.

```
buildtest list --modules

    Full Module Name                        |      ModuleFile Path
--------------------------------------------|-----------------------------
[32mRHEL6-apps                                  |        /nfs/grid/software/
↪moduledomains/RHEL6-apps.lua[0m
cctsoft                                     |        /nfs/grid/software/moduledomains/
↪cctsoft
clinpharm                                   |        /nfs/grid/software/moduledomains/
↪clinpharm
deprecated                                  |        /nfs/grid/software/moduledomains/
↪deprecated
easybuild                                   |        /nfs/grid/software/moduledomains/
↪easybuild
[32meb/2017                                     |        /nfs/grid/software/
↪moduledomains/eb/2017.lua[0m
[32meb/2018                                     |        /nfs/grid/software/
↪moduledomains/eb/2018.lua[0m
ljmedchemsoft                               |        /nfs/grid/software/moduledomains/
↪ljmedchemsoft
[32mlmod/6.5.1                                  |        /usr/share/lmod/lmod/
↪modulefiles/Core/lmod/6.5.1.lua[0m
```

(continues on next page)

(continued from previous page)

```
medsci                                  |        /nfs/grid/software/moduledomains/
↪medsci
omics                                   |        /nfs/grid/software/moduledomains/
↪omics
pharmsci                                |        /nfs/grid/software/moduledomains/
↪pharmsci
ru                                      |        /nfs/grid/software/moduledomains/ru
[32msettarg/6.5.1                            |         /usr/share/lmod/lmod/
↪modulefiles/Core/settarg/6.5.1.lua[0m
siterestricted                          |        /nfs/grid/software/moduledomains/
↪siterestricted
statistics                              |        /nfs/grid/software/moduledomains/
↪statistics
use.own.eb/append                       |        /nfs/grid/software/moduledomains/
↪use.own.eb/append
use.own.eb/prepend                      |        /nfs/grid/software/moduledomains/
↪use.own.eb/prepend


Total Software Modules: 18
[32mTotal LUA Modules: 5[0m
Total non LUA Modules: 13
```

6.1.3 List easyconfigs from module trees (`buildtest list --easyconfigs`)

buildtest can return a list of easyconfigs from module trees defined in `BUILDTEST_MODULEPATH`. You can run `buildtest list --easyconfigs` or short option `buildtest list -ec`.

buildtest will report full path to easyconfigs and also report any errors if it can't find any easyconfig. If you specify a module tree that is not built by easybuild you can expect some **warning** or **error** messages which is intended.

buildtest will attempt to search for any file with `.eb` extension in `easybuild` directory that is part of install directory of each software for every easybuild app.

```
buildtest list --easyconfigs
No easyconfigs found!


Unable to find easyconfigs for the following, please investigate this issue!

Reading File: /nfs/grid/software/moduledomains/RHEL6-apps.lua Unable to find variable␣
↪root in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/cctsoft Unable to find variable root␣
↪in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/clinpharm Unable to find variable root␣
↪in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/deprecated Unable to find variable␣
↪root in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/easybuild Unable to find variable root␣
↪in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/eb/2017.lua Unable to find variable␣
↪root in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/eb/2018.lua Unable to find variable␣
↪root in module file. This module is not generated by easybuild
Reading File: /nfs/grid/software/moduledomains/ljmedchemsoft Unable to find variable␣
↪root in module file. This module is not generated by easybuild
```

(continues on next page)

(continued from previous page)

```
Reading File: /usr/share/lmod/lmod/modulefiles/Core/lmod/6.5.1.lua Unable to find␣
↪variable root in module file. This module is not generated by easybuild
```

If an easyconfig is not found you may get the following message

Error: Could not find easyconfig in /clust/app/easybuild/2018/IvyBridge/redhat/7.3/software/NWChem/6.8.revision47-intel-2018a-2017-12-14-Python-2.7.14/easybuild

6.2 Find Subcommands

6.2.1 Find Operations (`buildtest find --help`)

```
buildtest find --help
usage: buildtest [options] [COMMANDS] find [-h] [-fc FINDCONFIG] [-ft FINDTEST]

optional arguments:
  -h, --help            show this help message and exit
  -fc FINDCONFIG, --findconfig FINDCONFIG
                        Find buildtest YAML config files. To find all yaml files use -
↪fc all
  -ft FINDTEST, --findtest FINDTEST
                        Find generated test scripts. To find all testscripts use -ft␣
↪all
```

6.2.2 Find YAML configs (`buildtest find -fc`)

To find all yaml configuration use the option `buildtest find --find-configs all` or short option `buildtest find -fc all`

This will give you an output of all **.yml** files in $BUILDTEST_CONFIGS_REPO that is used to configure on your test scripts.

```
buildtest find -fc all
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/helloworld/hello_
↪slurm.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/helloworld/hello_
↪lsf.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/helloworld/hello_
↪gnu.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/helloworld/hello_
↪args.c.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/helloworld/hello_
↪intel_fortran.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/mpi/examples/mpi_ping.c.
↪slurm.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/mpi/examples/hello.c.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/mpi/examples/mpi_ping.c.
↪lsf.yml
/home/siddis14/buildtest-framework/toolkit/buildtest/suite/mpi/matrixmux/mm_mpi.f.yml
```

6.2.3 Find test scripts (`buildtest find -ft`)

To find all test scripts generated by buildtest you can run `buildtest find --find-test all` or short option `buildtest find -ft all`

This will report the full path to all tests currently generated by buildtest.

```
buildtest find -ft all
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_slurm.yml.slurm
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_lsf.yml.lsf
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.sh
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_args.c.yml.sh
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_fortran.yml.sh
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.csh
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.bash
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_fortran.yml_
↪0xbb23d23609a7a848b9ebe7ef002a9323.sh
/tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_fortran.yml_
↪0x710d2b7cfdfb9a935b735f48330554ed.sh
```

6.3 Show Options (`buildtest show --help`)

```
buildtest show --help
usage: buildtest [options] [COMMANDS] show [-h] [-c] [-k {singlesource}]

optional arguments:
  -h, --help            show this help message and exit
  -c, --config          show buildtest environment configuration
  -k {singlesource}, --keys {singlesource}
                        show yaml keys
```

6.3.1 Show Configuration (`buildtest show --config`)

buildtest can display it's configuration by running `buildtest show --config`. The configuration can be changed by the following:

1. Command Line
2. Environment Variable (`BUILDTEST_`)
3. Configuration File (`settings.yml`)

buildtest will read configuration from `settings.yml`. User may override any configuration values by environment variables that start with `BUILDTEST_`. The command line will override environment variables and configuration variables runtime.

Shown below is a sample configuration from buildtest by running `buildtest show --config`.

```
buildtest show -c
         buildtest configuration summary
         (C): Configuration File,  (E): Environment Variable
BUILDTEST_BINARY                                (C) = False
BUILDTEST_CLEAN_BUILD                           (C) = False
BUILDTEST_CONFIGS_REPO                          (C) = /home/siddis14/buildtest-
↪framework/toolkit
```

(continues on next page)

(continued from previous page)

```
BUILDTEST_LOGDIR                                    (C) = /tmp/siddis14/buildtest/logs
BUILDTEST_MODULEPATH                                (C) = /nfs/grid/software/
↪moduledomains:/etc/modulefiles:/usr/share/modulefiles:/usr/share/lmod/lmod/
↪modulefiles/Core
BUILDTEST_PARENT_MODULE_SEARCH                      (C) = first
BUILDTEST_RUN_DIR                                   (C) = /tmp/siddis14/buildtest/run
BUILDTEST_SHELL                                     (C) = sh
BUILDTEST_SPIDER_VIEW                               (C) = current
BUILDTEST_SUCCESS_THRESHOLD                         (C) = 1.0
BUILDTEST_TESTDIR                                   (C) = /tmp/siddis14/buildtest/tests
```

`buildtest show --config` will show the updated configuration if you set any `BUILDTEST_*` environment variables.

For instance, if you want to override buildtest log using `BUILDTEST_LOGDIR` environment variable then `buildtest show --config` will report the overridden value denoted with **(E)** to indicate configuration was set by environment variable.

See example below

```
1        $ BUILDTEST_LOGDIR=$HOME buildtest show -c
2             buildtest configuration summary
3             (C): Configuration File,  (E): Environment Variable
4    BUILDTEST_BINARY                                (C) = False
5    BUILDTEST_CLEAN_BUILD                           (C) = False
6    BUILDTEST_CONFIGS_REPO                          (C) = /home/siddis14/buildtest-
↪framework/toolkit
7    BUILDTEST_EASYBUILD                             (C) = False
8    BUILDTEST_LOGDIR                                (E) = /home/siddis14
9    BUILDTEST_MODULEPATH                           (C) = /clust/app/easybuild/2018/
↪commons/modules/all:/clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all:/nfs/
↪grid/software/moduledomains:/etc/modulefiles:/usr/share/modulefiles:/usr/share/lmod/
↪lmod/modulefiles/Core
10   BUILDTEST_OHPC                                  (C) = False
11   BUILDTEST_PREPEND_MODULES                       (C) = []
12   BUILDTEST_RUN_DIR                               (C) = /tmp/buildtest
13   BUILDTEST_SHELL                                 (C) = sh
14   BUILDTEST_SUCCESS_THRESHOLD                     (C) = 1.0
15   BUILDTEST_TESTDIR                               (C) = /home/siddis14/buildtest
```

Note: if you plan to customize your buildtest configuration with configuration file and environment variable, always check your shell environment first to avoid having values overridden accidentally

6.3.2 Show Keys

buildtest can show YAML keys based on testblock. Currently, there is one testblock defined **singlesource**.

To show yaml keys you can run `buildtest show -k singlesource` to view all the YAML keys that pertain to `testblock: singlesource` found in YAML files

```
buildtest show -k singlesource
                                    General Keys
Keys                  | Description
-------------------------------------------------------------------------------
↪--------------
```

(continues on next page)

(continued from previous page)

```
args                  | Input arguments to be passed to the executable
compiler              | Specify a compiler that will be used for compiling the source␣
↪program
flags                 | Specify the build flags for compiling the source program
input                 | Specify input file for the executable. This file must be in
↪"src" directory
ldflags               | Flags that will be passed to linkder (ld)
lsf                   | Specify LSF configuration
slurm                 | Specify SLURM configuration
source                | Specify the name of the source file that is in the "src"␣
↪directory relative to yaml configuration
vars                  | Specify a list of environment variables that will be declared␣
↪in the test script

                                        LSF Keys

Keys                  | LSF Equivalents               | Description
------------------------------------------------------------------------------------
↪--------------
M                     | #BSUB -M                      | Memory Limit
R                     | #BSUB -R                      | Runs the job on a host that␣
↪meets the specified resource requirements
T                     | #BSUB -T                      | Sets the limit of the number␣
↪of concurrent threads to thread_limit for the whole job
W                     | #BSUB -W                      | Sets the runtime limit of the␣
↪job.
n                     | #BSUB -n                      | Submits a parallel job and␣
↪specifies the number of processors required to run the job

                                       SLURM Keys

Keys                  | Slurm Equivalents             | Description
------------------------------------------------------------------------------------
↪--------------
constraint            | #SBATCH --constraint          | specify a list of constraints
cpus-per-task         | #SBATCH --cpus-per-task       | number of cpus required per␣
↪task.
mem                   | #SBATCH --mem                 | minimum amount of real memory.
mem-per-cpu           | #SBATCH --mem-per-cpu         | maximum amount of real memory␣
↪per allocated
nodes                 | #SBATCH --nodes               | number of nodes on which to␣
↪run (N = min[-max])
ntasks                | #SBATCH --ntasks              | number of tasks to run.
ntasks-per-node       | #SBATCH --ntasks-per-node     | number of tasks to invoke on␣
↪each node
time                  | #SBATCH --time                | time limit
```

CHAPTER

SEVEN

MODULE OPERATION

Contents

7.1 Module Options (`buildtest module --help`)

```
buildtest module --help
usage: buildtest [options] [COMMANDS] module [-h] [--diff-trees DIFF_TREES] [-eb] [--
↪spack]
                                            {loadtest,tree,collection} ...

optional arguments:
  -h, --help            show this help message and exit
  --diff-trees DIFF_TREES
                        Show difference between two module trees
  -eb, --easybuild      reports modules that are built by easybuild
  --spack               reports modules that are built by spack

subcommands:
  valid subcommands

  {loadtest,tree,collection}
    loadtest            module load test
```

(continues on next page)

(continued from previous page)

```
    tree                module tree operation
    collection          module collection operation
```

7.2 Difference Between Module Trees (`buildtest module --diff-trees`)

buildtest can report differences between two module trees that can be useful if you deploy your software in a **stage/prod** module tree and you want to keep these trees in sync.

If your HPC site builds software stack for each architecture and your environment is heterogeneous then `--diff-trees` option will be helpful.

buildtest takes two trees as argument in the form of `buildtest --diff-tree tree1,tree2` where trees are separated by a comma. The tree must point to the root of the module tree in your system and buildtest will walk through the entire tree. We expect this operation to be quick given that the module tree is on the order of few thousand module files which is a reasonable count of module files in a large HPC facility.

```
buildtest module --diff-trees /clust/app/easybuild/2018/commons/modules/all,/usr/
↪share/lmod/lmod/modulefiles/Core
                        Comparing Module Trees for differences in module files
                        ------------------------------------------------------
Module Tree 1:  /clust/app/easybuild/2018/commons/modules/all
Module Tree 2:  /usr/share/lmod/lmod/modulefiles/Core
ID       |     Module                                                    |   Module␣
↪Tree 1    |   Module Tree 2
---------|---------------------------------------------------------------|-------------
↪-------|----------------------
1        | lmod/6.5.1                                                    | NOT FOUND  ␣
↪        | FOUND
2        | CUDA/9.1.85                                                   | FOUND      ␣
↪        | NOT FOUND
3        | CUDA/7.5.18                                                   | FOUND      ␣
↪        | NOT FOUND
4        | EasyBuild/3.6.0                                               | FOUND      ␣
↪        | NOT FOUND
5        | EasyBuild/3.5.3                                               | FOUND      ␣
↪        | NOT FOUND
6        | git-lfs/2.4.0                                                 | FOUND      ␣
↪        | NOT FOUND
7        | Anaconda2/5.1.0                                               | FOUND      ␣
↪        | NOT FOUND
8        | IGV/2.3.98-Java-1.8.0_152                                     | FOUND      ␣
↪        | NOT FOUND
9        | Anaconda3/5.1.0                                               | FOUND      ␣
↪        | NOT FOUND
10       | CUDA/8.0.61                                                   | FOUND      ␣
↪        | NOT FOUND
11       | settarg/6.5.1                                                 | NOT FOUND  ␣
↪        | FOUND
12       | cuDNN/7.1-CUDA-9.1.85                                         | FOUND      ␣
↪        | NOT FOUND
13       | Java/1.8.0_152                                                | FOUND      ␣
↪        | NOT FOUND
```

If your site supports multiple architecture and you want to find difference between the stacks then you will find `--diff-trees` to be handy. If the stacks are same you will see the following message

```
buildtest module --diff-trees /clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/
↪all,/clust/app/easybuild/2018/IvyBridge/redhat/7.3/modules/all
No difference found between module tree:  /clust/app/easybuild/2018/Broadwell/redhat/
↪7.3/modules/all  and module tree:  /clust/app/easybuild/2018/IvyBridge/redhat/7.3/
↪modules/all
```

7.3 Module Load Testing (`buildtest module loadtest`)

buildtest provides feature to test `module load` functionality on all module files in a module tree. This assumes you have the module tree in `MODULEPATH` in order for `module` command to work properly.

To use this feature specify the appropriate module tree for parameter `BUILDTEST_MODULEPATH` in `settings.yml` or via environment variable. To use this feature you need to use `buildtest module loadtest`

To demonstrate let's start off with an example where we test module load for a single module tree.

```
[siddis14@amrndhl1228 buildtest-framework]$ buildtest --show | grep BUILDTEST_
↪MODULEPATH
BUILDTEST_MODULEPATH                              (C) = /nfs/grid/software/RHEL7/non-
↪easybuild/modules/all
```

Let's start the test

```
buildtest module loadtest
module load RHEL6-apps
RUN: 1/18 STATUS: PASSED - Testing module: RHEL6-apps
_____________________________________________________________________________________
module load cctsoft
RUN: 2/18 STATUS: PASSED - Testing module: cctsoft
_____________________________________________________________________________________
module load clinpharm
RUN: 3/18 STATUS: PASSED - Testing module: clinpharm
_____________________________________________________________________________________
module load deprecated
RUN: 4/18 STATUS: PASSED - Testing module: deprecated
_____________________________________________________________________________________
module load easybuild
RUN: 5/18 STATUS: PASSED - Testing module: easybuild
_____________________________________________________________________________________
module load eb/2017
RUN: 6/18 STATUS: PASSED - Testing module: eb/2017
_____________________________________________________________________________________
module load eb/2018
RUN: 7/18 STATUS: PASSED - Testing module: eb/2018
_____________________________________________________________________________________
module load ljmedchemsoft
RUN: 8/18 STATUS: PASSED - Testing module: ljmedchemsoft
_____________________________________________________________________________________
module load lmod/6.5.1
RUN: 9/18 STATUS: PASSED - Testing module: lmod/6.5.1
_____________________________________________________________________________________
module load medsci
RUN: 10/18 STATUS: PASSED - Testing module: medsci
```

(continues on next page)

(continued from previous page)

```
_____________________________________________________________________________
module load omics
RUN: 11/18 STATUS: PASSED - Testing module: omics
_____________________________________________________________________________
module load pharmsci
RUN: 12/18 STATUS: PASSED - Testing module: pharmsci
_____________________________________________________________________________
module load ru
RUN: 13/18 STATUS: PASSED - Testing module: ru
_____________________________________________________________________________
module load settarg/6.5.1
RUN: 14/18 STATUS: PASSED - Testing module: settarg/6.5.1
_____________________________________________________________________________
module load siterestricted
RUN: 15/18 STATUS: PASSED - Testing module: siterestricted
_____________________________________________________________________________
module load statistics
RUN: 16/18 STATUS: PASSED - Testing module: statistics
_____________________________________________________________________________
module load use.own.eb/append
RUN: 17/18 STATUS: PASSED - Testing module: use.own.eb/append
_____________________________________________________________________________
module load use.own.eb/prepend
RUN: 18/18 STATUS: PASSED - Testing module: use.own.eb/prepend
_____________________________________________________________________________
Writing Results to /tmp/modules-load.out
Writing Results to /tmp/modules-load.err
_____________________________________________________________________________
                     Module Load Summary
Module Trees:                              ['/nfs/grid/software/moduledomains', '/etc/
↪modulefiles', '/usr/share/modulefiles', '/usr/share/lmod/lmod/modulefiles/Core']
PASSED:                                    18
FAILED:                                    0
_____________________________________________________________________________
```

buildtest will attempt to run `module load` against each module to verify modules are working properly.

You may specify additional module trees using `BUILDTEST_MODULEPATH` for module testing.

If you want to test all modules that were detected by `spider` utility, you can set `BUILDTEST_SPIDER_VIEW=all` in your configuration or environment variable or just run as follows:

```
BUILDTEST_SPIDER_VIEW=all buildtest module loadtest
```

This will test all modules retrieved by spider utility.

7.4 Module Trees Operation

buildtest supports adding, removing and listing module trees. Internally, buildtest is modifying BUILDTEST_MODULEPATH which is synonymous to MODULEPATH though, buildtest makes use of `BUILDTEST_MODULEPATH` when querying modules using `spider` command.

At your site, you may be interested in testing software by each stack.

By default, `BUILDTEST_MODULEPATH` is set to an empty list `[]` in configuration file `$HOME/.buildtest/settings.yml`. In this case, `BUILDTEST_MODULEPATH` will read from `MODULEPATH`.

7.4.1 Listing Module Tree (`buildtest module tree -l`)

To list the module trees in buildtest you can run `buildtest module tree -l` which shows one module tree per line

```
buildtest module tree -l
/nfs/grid/software/moduledomains
/etc/modulefiles
/usr/share/modulefiles
/usr/share/lmod/lmod/modulefiles/Core
```

For this run, `BUILDTEST_MODULEPATH` is not set in configuration file so it is reading from `MODULEPATH`

```
$ cat ~/.buildtest/settings.yml  | grep -i BUILDTEST_MODULEPATH
BUILDTEST_MODULEPATH: []
```

7.4.2 Adding Module Tree (`buildtest module tree -a`)

You can add new module tree through command line using `buildtest module tree -a /path/to/tree` which will update the configuration file

```
buildtest module tree -a /usr/share/lmod/lmod/modulefiles/Core
Adding module tree: ['/usr/share/lmod/lmod/modulefiles/Core']
Configuration File: /home/siddis14/.buildtest/settings.yml has been updated
```

7.4.3 Removing Module Tree (`buildtest module tree -r`)

Similarly you can remove module tree from your configuration via `buildtest module tree -r /path/to/tree`

```
buildtest module tree -r /usr/share/lmod/lmod/modulefiles/Core
Removing module tree: ['/usr/share/lmod/lmod/modulefiles/Core']
Configuration File: /home/siddis14/.buildtest/settings.yml has been updated
```

7.5 Report Easybuild Modules (`buildtest module --easybuild`)

buildtest can detect modules that are built by Easybuild. An easybuild module will contain a string in module file as follows:

```
Built with EasyBuild version 3.7.1
```

buildtest will check all module trees defined by `BUILDTEST_MODULEPATH` and search for string without the version number. To enable this feature use `buildtest module --easybuild` or short option `buildtest module -eb`.

Shown below is the output of easybuild retrieval.

```
buildtest module --easybuild


Total Easybuild Modules: 0
Total Modules Searched: 18
```

If you want buildtest to retrieve all records from `spider` to seek out all easybuild modules consider setting `BUILDTEST_SPIDER_VIEW=all` in configuration or environment variable. Shown below is an output when running `BUILDTEST_SPIDER_VIEW=all buildtest module --easybuild`

```
Module: /nfs/grid/software/RHEL7/easybuild/modules/all/Compiler/GCCcore/5.4.0/zlib/.1.
↪2.8.lua is built with Easybuild
Module: /nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/5.4.0-2.27/OpenMPI/2.0.
↪0/zlib/.1.2.8.lua is built with Easybuild
Module: /nfs/grid/software/RHEL7/easybuild/modules/all/Compiler/GCCcore/6.2.0/zlib/.1.
↪2.8.lua is built with Easybuild
Module: /nfs/grid/software/RHEL7/easybuild/modules/all/MPI/intel/2017.1.132-GCC-5.4.0-
↪2.27/impi/2017.1.132/zlib/.1.2.8.lua is built with Easybuild
Module: /nfs/grid/software/RHEL7/easybuild/modules/all/MPI/intel-CUDA/2017.1.132-GCC-
↪5.4.0-2.27-8.0.44/impi/2017.1.132/zlib/.1.2.8.lua is built with Easybuild
Module: /clust/app/easybuild/2018/Broadwell/redhat/7.3/modules/all/zlib/1.2.11-
↪GCCcore-6.4.0.lua is built with Easybuild


Total Easybuild Modules: 404
Total Modules Searched: 824
```

7.6 Report Spack Modules (`buildtest module --spack`)

buildtest can detect Spack modules. A spack module has a string to denote this module was created by spack with timestamp of module creation. Shown below is an example:

```
Module file created by spack (https://github.com/spack/spack) on 2019-04-11 11:38:31.
↪191604
```

buildtest will search for string `Module file created by spack` in modulefile. buildtest will run this for all modules in module trees defined by `BUILDTEST_MODULEPATH`.

```
buildtest module --spack


Total Spack Modules: 0
Total Modules Searched: 18
```

To retrieve all records `spider` to find all spack modules in your system consider running `BUILDTEST_SPIDER_VIEW=all buildtest module --spack`.

```
BUILDTEST_SPIDER_VIEW=all buildtest module --spack
Module: /nfs/grid/software/RHEL7/medsci/modules/all/ffmpeg/3.2.4-n6ulc43.lua is built␣
↪with Spack
Module: /nfs/grid/software/RHEL7/medsci/software/spack/dev/share/spack/lmod/linux-
↪rhel7-x86_64/openmpi/3.1.3-bs5h3cj/Core/fftw/3.3.8-67xoq34.lua is built with Spack
Module: /nfs/grid/software/RHEL7/medsci/software/spack/dev/share/spack/lmod/linux-
↪rhel7-x86_64/openmpi/3.1.3-bs5h3cj/Core/gromacs/2018.1-orqoszc.lua is built with␣
↪Spack
Module: /nfs/grid/software/RHEL7/medsci/modules/all/gromacs/2018.1-orqoszc.lua is␣
↪built with Spack
Module: /nfs/grid/software/RHEL7/medsci/modules/all/gromacs/2018.4-45ev4jn.lua is␣
↪built with Spack
Module: /nfs/grid/software/RHEL7/medsci/software/spack/dev/share/spack/lmod/linux-
↪rhel7-x86_64/openmpi/3.1.3-bs5h3cj/Core/gromacs/2018.4-unxhn3r.lua is built with␣
↪Spack
```

(continues on next page)

(continued from previous page)

```
Module: /nfs/grid/software/RHEL7/medsci/modules/all/openmpi/.3.1.3-bs5h3cj.lua is␣
↪built with Spack
Module: /nfs/grid/software/RHEL7/medsci/software/spack/dev/share/spack/lmod/linux-
↪rhel7-x86_64/openmpi/3.1.3-bs5h3cj/Core/plumed/2.4.2-kgdtnfu.lua is built with Spack


Total Spack Modules: 8
Total Modules Searched: 824
```

7.7 Parent Modules (`buildtest module --module-deps`)

Parent modules are modules that set **MODULEPATH** in the modulefile. This technique is used in **Hierarchical Module Naming Scheme** where modules like compilers, mpi, numlibs expose new module trees. These modules are called parent modules.

buildtest can report list of modules depended on a parent module. First, buildtest will seek out all parent module from file `BUILDTEST_ROOT/vars/modules.json`.

To seek out modules that depend on parent modules use the option `buildtest module --module-deps` or short option `buildtest module -d`.

Shown below are the list of parent modules that can be used with `buildtest module -d` upon tab completion.

```
(siddis14-TgVBs13r) buildtest-framework[master ?] $ buildtest module -d
cctsoft                         eb/2018                         GCCcore/.6.2.0    ␣
↪               impi/2017.1.132                 OpenMPI/2.0.1                     ␣
↪RHEL6-apps
CUDA/8.0.44                     GCC/5.4.0-2.27                  hpc/eb-2017-core  ␣
↪               medsci                          OpenMPI/2.0.2
deprecated                      GCC/6.2.0-2.27                  icc/.2017.1.132-
↪GCC-5.4.0-2.27    omics                          openmpi/.3.1.3-bs5h3cj
eb/2017                         GCCcore/.5.4.0                  ifort/.2017.1.132-
↪GCC-5.4.0-2.27  OpenMPI/2.0.0                  pharmsci
```

Shown below is a sample run for parent module `OpenMPI/2.0.1`. buildtest will report the content of the module file and list of modules that are depended upon the module.

```
buildtest module -d OpenMPI/2.0.1
Module File: /nfs/grid/software/RHEL7/easybuild/modules/all/Compiler/GCC/6.2.0-2.27/
↪OpenMPI/2.0.1.lua
------------------------------------------------------------------------------
help([[The Open MPI Project is an open source MPI-2 implementation. - Homepage: http:/
↪/www.open-mpi.org/]])

whatis([[Description: The Open MPI Project is an open source MPI-2 implementation. -␣
↪Homepage: http://www.open-mpi.org/]])

local root = "/nfs/grid/software/RHEL7/easybuild/software/Compiler/GCC/6.2.0-2.27/
↪OpenMPI/2.0.1"

conflict("OpenMPI")

load("hwloc/1.11.3")
prepend_path("MODULEPATH", "/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/6.
↪2.0-2.27/OpenMPI/2.0.1")
```

(continues on next page)

(continued from previous page)

```
prepend_path("CPATH", pathJoin(root, "include"))
prepend_path("LD_LIBRARY_PATH", pathJoin(root, "lib"))
prepend_path("LIBRARY_PATH", pathJoin(root, "lib"))
prepend_path("MANPATH", pathJoin(root, "share/man"))
prepend_path("PATH", pathJoin(root, "bin"))
prepend_path("PKG_CONFIG_PATH", pathJoin(root, "lib/pkgconfig"))
setenv("EBROOTOPENMPI", root)
setenv("EBVERSIONOPENMPI", "2.0.1")
setenv("EBDEVELOPENMPI", pathJoin(root, "easybuild/Compiler-GCC-6.2.0-2.27-OpenMPI-2.
↪0.1-easybuild-devel"))


family("MPI")
local lsf_libdir = os.getenv("LSF_LIBDIR")
prepend_path("LD_LIBRARY_PATH",lsf_libdir)

-- Built with EasyBuild version 3.1.2


-------------------------------------------------------------------------------
Modules that depend on OpenMPI/2.0.1
/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/6.2.0-2.27/OpenMPI/2.0.1/
↪ScaLAPACK/2.0.2-OpenBLAS-0.2.19-LAPACK-3.6.0.lua
/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/6.2.0-2.27/OpenMPI/2.0.1/GLPK/
↪4.60.lua
/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/6.2.0-2.27/OpenMPI/2.0.1/
↪OpenBLAS/0.2.19-LAPACK-3.6.0.lua
/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/6.2.0-2.27/OpenMPI/2.0.1/FFTW/
↪3.3.5.lua
/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/GCC/6.2.0-2.27/OpenMPI/2.0.1/Boost/
↪1.60.0.lua


Total Modules Found: 5
```

CHAPTER

EIGHT

BUILDING TEST

8.1 Overview

Contents

8.1.1 Build Options (`buildtest build --help`)

```
buildtest build --help
usage: buildtest [options] [COMMANDS] build [-h] [-s INSTALLED-SOFTWARE] [-p SYSTEM-
↪PACKAGE] [--shell {sh,bash,csh}]
                                            [-c TEST CONFIGURATION] [-b] [-S
↪{compilers,mpi,openmp,cuda}]
                                            [--clean-tests] [--testdir TESTDIR] [--
↪clean-build] [-v]
                                            [-m MODULES | -co {corrupt,corrupt2,
↪gcc540,intelmpi} | -mc COLLECTION-ID]
                                            [-pms {first,all}]

optional arguments:
  -h, --help            show this help message and exit
  -s INSTALLED-SOFTWARE, --software INSTALLED-SOFTWARE
                        Specify software package to test
  -p SYSTEM-PACKAGE, --package SYSTEM-PACKAGE
                        Build test for system packages
  --shell {sh,bash,csh}
```

(continues on next page)

(continued from previous page)

```
                        Select the type of shell for testscript
  -c TEST CONFIGURATION, --config TEST CONFIGURATION
                        Specify test configuration
  -b, --binary          Conduct binary test for a package
  -S {compilers,mpi,openmp,cuda}, --suite {compilers,mpi,openmp,cuda}
                        specify test suite
  --clean-tests         delete test directory $BUILDTEST_TESTDIR
  --testdir TESTDIR     Specify alternate test directory to write test. This␣
↪overrides configuration BUILDTEST_TESTDIR
  --clean-build         delete software test directory before writing test scripts
  -v, --verbose         verbosity level (default: 0)
  -m MODULES, --modules MODULES
                        Specify comma separated list of modules to build a test for␣
↪every module version.
  -co {corrupt,corrupt2,gcc540,intelmpi}, --collection {corrupt,corrupt2,gcc540,
↪intelmpi}
                        Use user Lmod module collection whenbuilding test
  -mc COLLECTION-ID, --module-collection COLLECTION-ID
                        Use internal buildtest module collection when building test.
↪Run buildtest collection -l to
                        view list of collection ids
  -pms {first,all}, --parent-module-search {first,all}
                        control how many parent module combination to search
```

8.1.2 Test Configuration

buildtest makes use of test configuration to generate the test script. This can be done by running `buildtest build -c /path/to/configuration`

Shown below is an example.

```
$ buildtest build -c $BUILDTEST_ROOT/toolkit/buildtest/suite/compilers/helloworld/
↪hello_gnu.yml
Writing Test: /home/siddis14/buildtest/suite/compilers/helloworld/hello_gnu.yml.sh
```

buildtest has two levels of verbosity that can be set by using `-v` option. buildtest will check the programming language, compiler and verify all the keys in configuration file before building the test.

buildtest will set the permission of test script to `755`.

```
$ buildtest build -c $BUILDTEST_ROOT/toolkit/buildtest/suite/compilers/helloworld/
↪hello_gnu.yml -v
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml
Programming Language Detected: c++
Compiler Check Passed
Writing Test: /home/siddis14/buildtest/suite/compilers/helloworld/hello_gnu.yml.sh
Changing permission to 755 for test: /home/siddis14/buildtest/suite/compilers/
↪helloworld/hello_gnu.yml.sh
```

You may specify additional level verbosity by `-vv` or specify `-v -v` which will give additional output including the output of configuration file and test script.

```
$ buildtest build -c $BUILDTEST_ROOT/toolkit/buildtest/suite/compilers/helloworld/
↪hello_gnu.yml -v -v
_____________________________________________________________________________
compiler: gnu
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.cpp
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.cpp exists!
Programming Language Detected: c++
Compiler Check Passed
Writing Test: /home/siddis14/buildtest/suite/compilers/helloworld/hello_gnu.yml.sh
Changing permission to 755 for test: /home/siddis14/buildtest/suite/compilers/
↪helloworld/hello_gnu.yml.sh
_____________________________________________________________________________
#!/bin/sh
module purge
module load eb/2018
cd /home/siddis14/buildtest/suite/compilers/helloworld
g++ -O3 -o hello.cpp.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.cpp
./hello.cpp.exe
rm ./hello.cpp.exe
_____________________________________________________________________________
```

8.1.3 Test Suites

Test Suite is a collection of test configuration that is meant for organizing tests. Test suite can be found at https://github.com/HPC-buildtest/buildtest-framework/tree/master/toolkit/buildtest/suite. and each sub-directory is a separate test suite.

A test suite is capable of building all test configuration (`.yml` files) found in its subdirectories. To build a test suite you can execute `buildtest build -S <suite>`

To know more about test suite see *Test Suite*

8.1.4 Sanity Check for System Packages

buildtest can perform sanity check for all binaries defined by a system package. This may be useful when running test periodically to monitor system changes.

To build test for system package you will want to use `buildtest build --package <package>` and specify the name of the installed system package.

For instance, lets build the tests for `coreutils` package by running `buildtest build --package coreutils`

The output will be the following

```
buildtest build --package coreutils
Detecting System Package: coreutils
Generating  102  binary tests
Binary Tests are written in  /tmp/siddis14/buildtest/tests/system/coreutils
Writing Log file to:  /tmp/siddis14/buildtest/logs/system/coreutils/buildtest_08_29_
↪14_05_2019.log
```

8.1.5 Sanity Check for Modules

buildtest can conduct sanity check for all active modules by running `-b`, `--binary` option or setting `BUILDTEST_BINARY=True` in your configuration file.

For instance let's assume the following modules are active modules in your shell

```
$ ml

Currently Loaded Modules:
  1) eb/2018   2) GCCcore/6.4.0   3) binutils/2.28-GCCcore-6.4.0   4) GCC/6.4.0-2.28
```

buildtest will seek out all binary executables in each module file and run `which` command against the binary and load the appropriate modules

Shown below is an example.

```
$ buildtest build -b
Detecting Software:eb/2018
No $PATH set in your module  eb/2018   so no possible binaries can be found
There are no binaries for package: eb/2018
Detecting Software:GCCcore/6.4.0
Generating  19  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/GCCcore/6.4.0
Detecting Software:binutils/2.28-GCCcore-6.4.0
Generating  18  binary tests
Binary Tests are written in  /home/siddis14/buildtest/software/binutils/2.28-GCCcore-
↪6.4.0
Detecting Software:GCC/6.4.0-2.28
No $PATH set in your module  GCC/6.4.0-2.28   so no possible binaries can be found
There are no binaries for package: GCC/6.4.0-2.28
```

modules that dont have `PATH` set or no binary executables are found in the directory, then buildtest will not generate any test.

Shown below is an example test script for gcc binary

```
#!/bin/sh


module load GCCcore/6.4.0
which gcc
```

8.1.6 Clean build (`buildtest build --clean-build`)

buildtest will preserve the testing directory when tests are generated. For example, if you run the following

```
buildtest build --package gcc --shell sh
buildtest build --package gcc --shell csh
buildtest build --package gcc --shell bash
```

This will write the test for shell ("sh", "bash", "csh") in the same directory. If you want to remove the directory prior to running test you can do the following

```
buildtest build --package gcc --clean-build
```

8.1.7 Customize Test Directory (`buildtest build --testdir`)

If you want to customize the path to BUILDTEST_TESTDIR you may use the option `--testdir` or update the environment variable `BUILDTEST_TESTDIR`. The command line option will override environment variable and environment variable will override configuration value.

```
buildtest build --package gcc --testdir /home/siddis14/tmp
Detecting System Package: gcc
Generating  8  binary tests
Binary Tests are written in  /home/siddis14/tmp/system/gcc
Writing Log file to:  /tmp/siddis14/buildtest/logs/system/gcc/buildtest_08_29_14_05_
↪2019.log
```

8.1.8 Shell Types

Currently buildtest supports `sh`, `bash`, `csh` shell for creating test scripts. buildtest defaults to `sh` but this can be tweaked

To create tests for different shell types try `buildtest build --shell <shell>` or set the variable `BUILDTEST_SHELL` in your configuration file or via environment variable

Let's build test with `csh`

```
buildtest build -c /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml --shell csh
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.
↪csh
```

buildtest will add the appropriate shell extension for the test script to avoid name conflicts.

Another way to build for different shell is to set `BUILDTEST_SHELL` as we see in example below

```
BUILDTEST_SHELL=bash buildtest build -c /home/siddis14/buildtest-framework/toolkit/
↪buildtest/suite/compilers/helloworld/hello_gnu.yml
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.
↪bash
```

8.2 Module Collections

8.2.1 Lmod Module Collection

buildtest supports building test with Lmod user collections module collection comes in handy when testing software with different module sets.

You can run `module -t savelist` to see a list of collection. Shown below is an example:

```
$ module -t savelist
corrupt
corrupt2
default
intelmpi
```

To restore a module collection you can run:

```
module restore <collection-name>
```

To build a test with a module collection use the `--collection` option which is a choice field that is the name of the module collection. To demonstrate, lets build a test with a module collection `intelmpi`.

```
buildtest build -c /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
 ↪compilers/helloworld/hello_intel_fortran.yml -co intelmpi -vv
_____________________________________________________________________________
compiler: intel
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.f90
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
 ↪compilers/helloworld/hello_intel_fortran.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
 ↪helloworld/src/hello.f90 exists!
Programming Language Detected: fortran
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
 ↪fortran.yml.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
 ↪helloworld/hello_intel_fortran.yml.sh
_____________________________________________________________________________
#!/bin/sh
module restore intelmpi
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
 ↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
```

Note the `module restore` command will load the modules from the collection. To use this feature, you will need to have a module collection which are typically stored in `$HOME/.lmod.d/`

8.2.2 buildtest module collection

buildtest keeps track of its own module collection which is stored in `BUILDTEST_ROOT/vars/default.json`. This file is automatically generated at start of buildtest.

buildtest module collection works as follows.

First you load a set of modules:

```
$ ml

Currently Loaded Modules:
  1) eb/2018   2) CUDA/9.1.85
```

Next add the modules to your collection by running:

```
$ buildtest module collection -a
{
    "collection": [
        [
            "eb/2018",
            "CUDA/9.1.85"
        ]
    ]
}

Updating collection file: /home/siddis14/buildtest-framework/var/default.json
```

Next, view the module collection by running:

```
$ buildtest module collection -l
0: ['eb/2018', 'CUDA/9.1.85']
```

buildtest will display module collection with index starting **0** followed by name of modules. Let's add a couple more collections by repeating same step by adding different modules.

Shown below, we have 4 module collection that shows the index followed by a list of modules

```
$ buildtest module collection -l
0: ['eb/2018', 'CUDA/9.1.85']
1: ['eb/2018', 'GCCcore/6.4.0', 'binutils/2.28-GCCcore-6.4.0', 'GCC/6.4.0-2.28']
2: ['eb/2017', 'icc/.2017.1.132-GCC-5.4.0-2.27', 'GCCcore/.5.4.0', 'binutils/.2.27',
↪'ifort/.2017.1.132-GCC-5.4.0-2.27', 'impi/2017.1.132', 'imkl/2017.1.132', 'intel/
↪2017.01']
3: ['eb/2017', 'GCCcore/.6.2.0', 'binutils/.2.27', 'GCC/6.2.0-2.27']
```

Let's build a test with the 2nd module collection by using `--module-collection` option or short option `-mc`.

```
(siddis14-TgVBs13r) buildtest-framework[master !?+] $ buildtest build -c  $(pwd)/
↪toolkit/buildtest/suite/compilers/helloworld/hello_intel_fortran.yml -mc 2 -vv
_____________________________________________________________________________
compiler: intel
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.f90
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_intel_fortran.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.f90 exists!
Programming Language Detected: fortran
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/suite/compilers/helloworld/hello_intel_fortran.
↪yml.sh
```

(continues on next page)

(continued from previous page)

```
Changing permission to 755 for test: /tmp/siddis14/buildtest/suite/compilers/
↪helloworld/hello_intel_fortran.yml.sh

_____________________________________________________________________________

#!/bin/sh
module load eb/2017 icc/.2017.1.132-GCC-5.4.0-2.27 GCCcore/.5.4.0 binutils/.2.27␣
↪ifort/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132 imkl/2017.1.132 intel/2017.01
cd /tmp/siddis14/buildtest/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe

_____________________________________________________________________________
```

The option `--module-collection` takes an integer argument that is a choice field prepopulated by finding the total index in the `collection` key in file `default.json`

If you pass an invalid index, buildtest will report an error as you can see

Error: buildtest build: error: argument -mc/–module-collection: invalid choice: -1 (choose from 0, 1, 2, 3)

8.2.3 Module Permutation

buildtest can build a single test configuration with all version of a module . The `spider` utility by Lmod keeps track of metadata for all modules in your system as a json object. buildtest will formulate a modified json object that is written in `$BUILDTEST_ROOT/var/modules.json`.

Here is an example json object for **intel**.

```
"intel": {
    "/nfs/grid/software/RHEL6/general/intel/16.0.lua": {
        "fullName": "intel/16.0",
        "parent": [
            [
                "RHEL6-apps"
            ]
        ]
    },
    "/nfs/grid/software/RHEL6/general/intel/14.0.lua": {
        "fullName": "intel/14.0",
        "parent": [
            [
                "RHEL6-apps"
            ]
        ]
    },
    "/nfs/grid/software/RHEL7/easybuild/modules/all/Core/intel/2017.01.lua": {
        "fullName": "intel/2017.01",
        "parent": [
            [
                "eb/2017"
            ],
            [
                "medsci",
                "hpc/eb-2017-core"
```

(continues on next page)

(continued from previous page)

```
            ]
        ]
    },
```

Shown below is a list of intel modules available in this system

```
$ module -t spider intel
intel/14.0
intel/15.0
intel/16.0
intel/16.0.1
intel/16.0.3
intel/2017.01
intel/2018a
```

To demonstrate an example, let's build a test using the module permutation option `--modules` on all `intel` modules.

```
 buildtest build -c  /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_intel_fortran.yml --modules intel -vv
Module Permutation Detected.
Each test will be built with 7 module permutations
Module Permutation List
_____________________________________________________
module load RHEL6-apps intel/16.0
module load RHEL6-apps intel/14.0
module load eb/2017 intel/2017.01
module load RHEL6-apps intel/16.0.3
module load RHEL6-apps intel/15.0
module load RHEL6-apps intel/16.0.1
module load eb/2018 intel/2018a
_____________________________________________________________________________
compiler: intel
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.f90
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_intel_fortran.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.f90 exists!
Programming Language Detected: fortran
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0xbb23d23609a7a848b9ebe7ef002a9323.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0xbb23d23609a7a848b9ebe7ef002a9323.sh
_____________________________________________________________________________
#!/bin/sh
module load RHEL6-apps intel/16.0
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
```

(continues on next page)

(continued from previous page)

```
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x710d2b7cfdfb9a935b735f48330554ed.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x710d2b7cfdfb9a935b735f48330554ed.sh
_____________________________________________________________________________________
#!/bin/sh
module load RHEL6-apps intel/14.0
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x88ce320d3c7ed9da689eb4017b9bb8f5.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x88ce320d3c7ed9da689eb4017b9bb8f5.sh
_____________________________________________________________________________________
#!/bin/sh
module load eb/2017 intel/2017.01
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x361f018c8eee60f878f988be7aaee8cd.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x361f018c8eee60f878f988be7aaee8cd.sh
_____________________________________________________________________________________
#!/bin/sh
module load RHEL6-apps intel/16.0.3
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0xbe42206e12a730e3d397c6ee4dee83f8.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0xbe42206e12a730e3d397c6ee4dee83f8.sh
_____________________________________________________________________________________
#!/bin/sh
module load RHEL6-apps intel/15.0
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0xc7eb5591c162974bb90c62b08a26f7d1.sh
```

(continues on next page)

(continued from previous page)

```
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0xc7eb5591c162974bb90c62b08a26f7d1.sh
_____________________________________________________________________________
#!/bin/sh
module load RHEL6-apps intel/16.0.1
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x737e4ea256880ff1f5380bca5f2788ca.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x737e4ea256880ff1f5380bca5f2788ca.sh
_____________________________________________________________________________
#!/bin/sh
module load eb/2018 intel/2018a
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
Writing 7 tests for /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_intel_fortran.yml
```

Each test will be uniquely identified with a 128 random number in the test script to avoid name conflicts.

In this example, buildtest is building the test for every `intel` modules found in the system.

buildtest will select the `first` parent combination should there be multiple parent combination to load the module. This is controlled by variable `BUILDTEST_PARENT_MODULE_SEARCH` that is defined in configuration file.

The default configuration for `BUILDTEST_PARENT_MODULE_SEARCH` is `first` which will select the first parent combination. The other option is `all` which will select all parent combination when building test.

Shown below is a snapshot of `VMD` record from `modules.json`

```
"VMD": {
    "/nfs/grid/software/RHEL7/easybuild/modules/all/MPI/intel/2017.1.132-GCC-5.4.0-2.
↪27/impi/2017.1.132/VMD/1.9.3-Python-2.7.12.lua": {
        "fullName": "VMD/1.9.3-Python-2.7.12",
        "parent": [
            [
                "eb/2017",
                "icc/.2017.1.132-GCC-5.4.0-2.27",
                "impi/2017.1.132"
            ],
            [
                "eb/2017",
                "ifort/.2017.1.132-GCC-5.4.0-2.27",
                "impi/2017.1.132"
            ],
            [
                "icc/.2017.1.132-GCC-5.4.0-2.27",
                "impi/2017.1.132"
            ],
```

(continues on next page)

(continued from previous page)

```
            [
                "ifort/.2017.1.132-GCC-5.4.0-2.27",
                "impi/2017.1.132"
            ],
            [
                "medsci",
                "hpc/eb-2017-core",
                "icc/.2017.1.132-GCC-5.4.0-2.27",
                "impi/2017.1.132"
            ],
            [
                "medsci",
                "hpc/eb-2017-core",
                "ifort/.2017.1.132-GCC-5.4.0-2.27",
                "impi/2017.1.132"
            ]
        ]
    }
},
```

The `fullName` and `parent` key define how to load a module with all the parent combinations which you are required in order to load the desired module.

To demonstrate let's build a test with all parent combination for `VMD` module.

```
BUILDTEST_PARENT_MODULE_SEARCH=all buildtest build -c  /home/siddis14/buildtest-
↪framework/toolkit/buildtest/suite/compilers/helloworld/hello_intel_fortran.yml --
↪modules VMD -vv
Module Permutation Detected.
Each test will be built with 4 module permutations
Module Permutation List
_____________________________________________________
module load eb/2017 icc/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132 VMD/1.9.3-Python-2.
↪7.12
module load eb/2017 ifort/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132 VMD/1.9.3-Python-
↪2.7.12
module load medsci hpc/eb-2017-core icc/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132␣
↪VMD/1.9.3-Python-2.7.12
module load medsci hpc/eb-2017-core ifort/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132␣
↪VMD/1.9.3-Python-2.7.12
_____________________________________________________________________________________
compiler: intel
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.f90
testblock: singlesource


_____________________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_intel_fortran.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.f90 exists!
Programming Language Detected: fortran
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0xc80e042db904de9a3fa899d7e58ed78e.sh
```

(continues on next page)

(continued from previous page)

```
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0xc80e042db904de9a3fa899d7e58ed78e.sh
_____________________________________________________________________________
#!/bin/sh
module load eb/2017 icc/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132 VMD/1.9.3-Python-2.
↪7.12
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x33ee0f531d623d52cde69680dd973ee3.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x33ee0f531d623d52cde69680dd973ee3.sh
_____________________________________________________________________________
#!/bin/sh
module load eb/2017 ifort/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132 VMD/1.9.3-Python-
↪2.7.12
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x973a918edc31a04c62638eb390bba18d.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x973a918edc31a04c62638eb390bba18d.sh
_____________________________________________________________________________
#!/bin/sh
module load medsci hpc/eb-2017-core icc/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132␣
↪VMD/1.9.3-Python-2.7.12
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml_0x412dfac1afd2b8092e45330602c4f9d4.sh
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_intel_fortran.yml_0x412dfac1afd2b8092e45330602c4f9d4.sh
_____________________________________________________________________________
#!/bin/sh
module load medsci hpc/eb-2017-core ifort/.2017.1.132-GCC-5.4.0-2.27 impi/2017.1.132␣
↪VMD/1.9.3-Python-2.7.12
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
ifort -O3 -o hello.f90.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.f90
./hello.f90.exe
 rm ./hello.f90.exe
_____________________________________________________________________________
Writing 4 tests for /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_intel_fortran.yml
```

Note all 6 parent combination for module `VMD/1.9.3-Python-2.7.12` were used when writing the test. It is worth noting, that *any parent combination is sufficient* when loading the desired module.

8.3 Test Suite

Test suite is a collection of yaml files and source code grouped in a directory of similar types of test.

Currently, there is 4 test suites **compilers**, **cuda**, **openmp** and **mpi**. The test suites can be found at https://github.com/HPC-buildtest/buildtest-framework/tree/master/toolkit/buildtest/suite

8.3.1 Hello World C++

Let's take a look at a hello world C++ example that will be compiled with gcc

```
compiler: gnu
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.cpp
testblock: singlesource
```

The first line `compiler: gnu` is to indicate we will use the gnu compiler during compilation.

The `flags: -O3` will insert the build flag **-O3** during compilation

The key `maintainer` is a list of maintainers that are primary contact for the test & configuration file

The key `source: hello.cpp` is the source file, this file will need to reside in **src** directory wherever you have your yml file

Finally, `testblock: singlesource` inform buildtest that this is a single source compilation and buildtest will use the appropriate Class to build this test. Currently, `testblock` only supports singlesource at this moment.

Next let's see the generated test script

```
#!/bin/sh
module purge
module load eb/2018
cd /home/siddis14/buildtest/suite/compilers/helloworld
g++ -O3 -o hello.cpp.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.cpp
./hello.cpp.exe
rm ./hello.cpp.exe
```

Couple things to note.

- buildtest will purge and load the module that is active in your shell
- The test script will be named with the yml file and the appropriate shell extension `.sh`, `.bash`, `.csh`.
- buildtest will `cd` into the test directory where test script is found
- buildtest will detect the compiler based on extension type specified in `source` tag. In this case it will be `g++` since we specified `compiler: gnu`
- buildtest will compile the source file that was defined in `source` tag. buildtest will figure out the full path to file.
- The name of the executable will be the name of the source code with `.exe` extension.

- Finally buildtest will run executable and remove it upon completion.

This test is found in `compilers` suite and you can build the test suite by running `buildtest build -S compilers`

Shown below is the output

```
buildtest build -S compilers
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_slurm.
↪yml.slurm
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_lsf.yml.
↪lsf
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_gnu.yml.
↪sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_args.c.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_intel_
↪fortran.yml.sh
```

To run the compiler test suite you can run `buildtest run -S compilers`

Below is the output of the run

```
buildtest run -S compilers
Running All Tests from Test Directory: /tmp/siddis14/buildtest/tests/suite/compilers
============================================================
                         Test summary
Package:  compilers
Executed 5 tests
Passed Tests: 4 Percentage: 80.0%
Failed Tests: 1 Percentage: 20.0%
WARNING: Threshold of 100.0% was not satisfied
compilers has a 80.0% passed rate with a difference of 0.2 from the threshold
Writing results to /tmp/siddis14/buildtest/run/buildtest_08_29_14_05_2019.run
```

All test will be run and the output will be stored in the `.run` file which contains output of test run along with additional details buildtest was able to capture for your site.

8.3.2 OpenMP Example

Let's take a look at a OpenMP yml example for computing vector dot product

```
compiler: gnu
ldflags: -fopenmp
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: omp_dotprod.c
testblock: singlesource
vars:
  OMP_NUM_THREADS: 2
```

To run a OpenMP example you typically set the environment variable `OMP_NUM_THREADS` to declare number of threads during execution.

This can be configured used `vars:` keyword that takes a list of of key-value to set environment variable in the test script. In this example we set `OMP_NUM_THREADS=2`

To specify flags to linker `(ld)` then use key `ldflags`. In this case, to compile openmp with gnu compiler you need to specify `-fopenmp`.

Let's see the test script

```
#!/bin/sh
module purge
module load eb/2018
export OMP_NUM_THREADS=2
cd /home/siddis14/buildtest/suite/openmp/dotprod
gcc -o omp_dotprod.c.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪openmp/dotprod/src/omp_dotprod.c -fopenmp
./omp_dotprod.c.exe
rm ./omp_dotprod.c.exe
```

Let's build `openmp` test suite by running the following `buildtest build -S openmp`

```
buildtest build -S openmp
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/matrixmultiplication/omp_mm.
↪f.yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/matrixmultiplication/omp_mm.
↪c.yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/workshare/omp_workshare2.c.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/workshare/omp_workshare1.c.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/workshare/omp_workshare1.f.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/workshare/omp_workshare2.f.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/dotprod/omp_dotprod.c.yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/dotprod/omp_dotprod.f.yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/dotprod/omp_dotprod_serial.c.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/dotprod/omp_dotprod_serial.f.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/hello/omp_hello.c.yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/hello/omp_hello.f.yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/reduction/omp_reduction.f.
↪yml.sh
Writing Test: /tmp/siddis14/buildtest/tests/suite/openmp/reduction/omp_reduction.c.
↪yml.sh
```

Next let's run the test suite by running `buildtest run -S openmp`

```
buildtest run -S openmp
Running All Tests from Test Directory: /tmp/siddis14/buildtest/tests/suite/openmp
================================================================
                          Test summary
Package:  openmp
Executed 14 tests
Passed Tests: 14 Percentage: 100.0%
Failed Tests: 0 Percentage: 0.0%
SUCCESS: Threshold of 100.0% was achieved
Writing results to /tmp/siddis14/buildtest/run/buildtest_08_29_14_05_2019.run
```

8.3.3 Testing with modules

Now that we have built a couple test, we want to leverage modules to test a particular test with different modules. This may be particularly useful if you have some test that you want to compare with different compilers, MPI, etc...

Let's take the same hello world example and build it with different gcc compilers.

Recall the first test was the following

```
#!/bin/sh
module purge
module load eb/2018
cd /home/siddis14/buildtest/suite/compilers/helloworld
g++ -O3 -o hello.cpp.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.cpp
./hello.cpp.exe
rm ./hello.cpp.exe
```

In buildtest, just load the modules of interest before you build the test and it will insert all the modules in the test script.

For this example we have the following modules loaded

```
$ ml

Currently Loaded Modules:
  1) eb/2018   2) GCCcore/6.4.0   3) binutils/2.28-GCCcore-6.4.0   4) GCC/6.4.0-2.28
```

Let's rebuild the test and notice how the modules are loaded in the test

```
$ buildtest build -c $BUILDTEST_ROOT/toolkit/buildtest/suite/compilers/helloworld/
↪hello_gnu.yml -vv
_____________________________________________________________________________
compiler: gnu
flags: -O3
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.cpp
testblock: singlesource

_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_gnu.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.cpp exists!
Programming Language Detected: c++
Compiler Check Passed
Writing Test: /home/siddis14/buildtest/suite/compilers/helloworld/hello_gnu.yml.sh
Changing permission to 755 for test: /home/siddis14/buildtest/suite/compilers/
↪helloworld/hello_gnu.yml.sh
_____________________________________________________________________________
#!/bin/sh
module purge
module load eb/2018
module load GCCcore/6.4.0
module load binutils/2.28-GCCcore-6.4.0
module load GCC/6.4.0-2.28
cd /home/siddis14/buildtest/suite/compilers/helloworld
g++ -O3 -o hello.cpp.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.cpp
./hello.cpp.exe
rm ./hello.cpp.exe
_____________________________________________________________________________
```

buildtest will run `module purge` and load all the active modules by running `module -t list` and insert each

module in a separate line. This gives user freedom to load whatever module they want when creating test, though this puts responsibility on user to understand the testscript.

8.3.4 LSF Job Example

buildtest supports creation of job scripts for LSF batch scheduler. Test scripts with the extension **.lsf** will denote the test script is LSF job scripts.

Let's see an example configuration for LSF job

```
1     compiler: gnu
2     flags: -O2
3     lsf:
4       M: 200M
5       R: sandybridge
6       W: 01:00
7       n: '4'
8     maintainer:
9     - shahzeb siddiqui shahzebmsiddiqui@gmail.com
10    source: hello.c
11    testblock: singlesource
```

The lsf section starts with keyword `lsf:` defined in line 3. The LSF keys are named based on `bsub` options which makes it easy to associate the key to the equivalent bsub command. In this example above lines **4-7** describe the LSF parameters . These include `200MB` of memory with `1hr` walltime, `4` tasks and requesting `sandybridge` resource.

Note: Only a subset of lsf keys are exposed in yaml

You can run `buildtest show -k singlesource` to see description of all keys or refer to *Show Keys*

We can run this as follows

```
buildtest build -c /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_lsf.yml -vv
_____________________________________________________________________________
compiler: gnu
flags: -O2
lsf:
  M: 200M
  R: sandybridge
  W: 01:00
  n: '4'
maintainer:
- shahzeb siddiqui shahzebmsiddiqui@gmail.com
source: hello.c
testblock: singlesource


_____________________________________________________________________________
Key Check PASSED for file /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/hello_lsf.yml
Source File /home/siddis14/buildtest-framework/toolkit/buildtest/suite/compilers/
↪helloworld/src/hello.c exists!
Programming Language Detected: c
LSF Keys Passed
Compiler Check Passed
Writing Test: /tmp/siddis14/buildtest/tests/suite/compilers/helloworld/hello_lsf.yml.
↪lsf
```

(continues on next page)

(continued from previous page)

```
Changing permission to 755 for test: /tmp/siddis14/buildtest/tests/suite/compilers/
↪helloworld/hello_lsf.yml.lsf
_____________________________________________________________________________
#!/bin/sh
#BSUB -M 200M
#BSUB -R sandybridge
#BSUB -W 01:00
#BSUB -n 4
module purge
cd /tmp/siddis14/buildtest/tests/suite/compilers/helloworld
gcc -O2 -o hello.c.exe /home/siddis14/buildtest-framework/toolkit/buildtest/suite/
↪compilers/helloworld/src/hello.c
./hello.c.exe
 rm ./hello.c.exe
_____________________________________________________________________________
```

8.3.5 SLURM Job Example

buildtest supports job script for SLURM. Similar to LSF, the test script will be denoted with extension **.slurm** and start with the keyword `slurm:` in configuration. For more details on slurm keys in yaml file see *Show Keys*

Here is an example configuration for SLURM job

```
1     compiler: gnu
2     flags: -O2
3     maintainer:
4     - shahzeb siddiqui shahzebmsiddiqui@gmail.com
5     slurm:
6       mem: 200M
7       nodes: '4'
8       time: 01:00
9     source: hello.cpp
10    testblock: singlesource
```

Line 5 `slurm:` starts the slurm section. Slurm keys are named based on `sbatch` options so it is easy to remember.

Note: Only a subset of slurm keys are exposed in yaml

Line 6-8 define the slurm parameters, in this case we request for `200M` memory, with `4` nodes and walltime of `1h`.

We can run this as follows

```
cat: scripts/build-slurm-example.txt: No such file or directory
```

CHAPTER

NINE

RUNNING TEST

```
usage: buildtest run [-h] [-s SOFTWARE-TEST-SUITE] [-p PACKAGE-TEST-SUITE] [-S
↪{compilers,openmp}] [-j]

optional arguments:
  -h, --help            show this help message and exit
  -s SOFTWARE-TEST-SUITE, --software SOFTWARE-TEST-SUITE
                        Run test suite for application
  -p PACKAGE-TEST-SUITE, --package PACKAGE-TEST-SUITE
                        Run test suite for system package
  -S {compilers,openmp}, --suite {compilers,openmp}
                        Run the test suite
  -j, --job             Submit jobs to resource scheduler
```

9.1 Run an Application Test Suite (`buildtest run --software`)

buildtest can run test written in `$BUILDTEST_TESTDIR` for a particular application specified by option `--software`. The choice field for this option is populated based on directories found in `$BUILDTEST_TESTDIR` which were created by subsequent runs of `buildtest build -s <application>`.

```
(buildtest) [siddis14@adwnode11 buildtest-framework]$ buildtest run --software
GCC/6.4.0-2.28             GCCcore/6.4.0              Perl/5.26.0-GCCcore-6.4.0
```

Shown below is an output of `buildtest run --software GCCcore/6.4.0` which attempts to run all tests for application `GCCcore/6.4.0`

```
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/ebapp/GCCcore/6.4.0/omp_orphan.f.csh >/dev/null␣
↪2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/ebapp/GCCcore/6.4.0/omp_dotprod_serial.f.csh >/
↪dev/null 2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
============================================================
                         Test summary
Application:  GCCcore/6.4.0
Executed 126 tests
```

(continues on next page)

(continued from previous page)

```
Passed Tests: 126    Percentage: 100.0%
Failed Tests: 0    Percentage: 0.0%
```

9.2 Run a System Package Test Suite (`buildtest run --package`)

Similarly, `buildtest run --package` is used to run test suite for system packages that were built by option `buildtest build --package <package>`

Shown below is an output of `buildtest run --package gcc`

```
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_c89.sh >/dev/null 2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_c99.sh >/dev/null 2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_gcc.sh >/dev/null 2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_gcc-ar.sh >/dev/null 2>
↪&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_gcc-nm.sh >/dev/null 2>
↪&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_gcc-ranlib.sh >/dev/
↪null 2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
Executing Test: /tmp/buildtest-tests/system/gcc/which__usr_bin_gcov.sh >/dev/null 2>&1
------------------------------------------------------------
Test Successful
------------------------------------------------------------
============================================================
                        Test summary
System Package:  gcc
Executed 7 tests
Passed Tests: 7    Percentage: 100.0%
Failed Tests: 0    Percentage: 0.0%
```

CHAPTER

TEN

BENCHMARK SUBCOMMANDS

10.1 OSU MicroBenchmark

OSU MicroBenchmark is a benchmark suite by Ohio State University that is commonly packaged as part of mvapich2 software tool. The OSU MicroBenchmark is designed to test MPI communication between process. For more information about this benchmark check out http://mvapich.cse.ohio-state.edu/benchmarks/

```
(siddis14-TgVBs13r) buildtest-framework[master !?] $ buildtest benchmark osu --help
usage: buildtest [options] benchmark osu [-h] [-r] [-i] [-l] [-c CONFIG]

optional arguments:
-h, --help            show this help message and exit
-r, --run             Run Benchmark
-i, --info            show yaml key description
-l, --list            List of tests available for OSU Benchmark
-c CONFIG, --config CONFIG
                      OSU Yaml Configuration File
```

To get a sense of all the test available from this benchmark supported by buildtest you can run `buildtest benchmark osu -l` which will give a listing all of tests.

```
(buildtest) [siddis14@adwnode1 buildtest-framework]$ buildtest benchmark osu -l
Check Configuration

-------------------- TEST BREAKDOWN ---------------------------------
osu_bibw - Bidirectional Bandwidth Test
osu_bw - Bandwidth Test
osu_latency - Latency Test
osu_put_latency - Latency Test for Put
osu_get_latency - Latency Test for Get
osu_put_bw - Bandwidth Test for Put
osu_get_bw - Bandwidth Test for Get
osu_put_bibw - Bidirectional Bandwidth Test for Put
osu_acc_latency - Latency Test for Accumulate
osu_cas_latency - Latency Test for Compare and Swap
osu_fop_latency - Latency Test for Fetch and Op
osu_allgather - MPI_Allgather Latency Test
osu_allgatherv - MPI_Allgatherv Latency Test
osu_allreduce - MPI_Allreduce Latency Test
osu_alltoall - MPI_Alltoall Latency Test
osu_alltoallv - MPI_Alltoallv Latency Test
osu_bcast - MPI_Bcast Latency Test
osu_gather - MPI_Gather Latency Test
```

(continues on next page)

(continued from previous page)

```
osu_gatherv - MPI_Gatherv Latency Test
osu_reduce - MPI_Reduce Latency Test
osu_reduce_scatter - MPI_Reduce_scatter Latency Test
osu_scatter - MPI_Scatter Latency Test
osu_scatterv - MPI_Scatterv Latency Test
osu_iallgather - MPI_Iallgather Latency Test
osu_iallreduce - MPI_Iallreduce Latency Test
osu_ialltoall - MPI_Ialltoall Latency Test
osu_ibcast - MPI_Ibcast Latency Test
osu_igather - MPI_Igather Latency Test
osu_ireduce - MPI_Iallreduce Latency Test
osu_iscatter - MPI_Iscatter Latency Test
----------------------------------------------------------------

------------------------- POINT-TO-POINT MPI BENCHMARKS -------------------------
osu_latency - Latency Test
osu_latency_mt - Multi-threaded Latency Test
osu_bw - Bandwidth Test
osu_bibw - Bidirectional Bandwidth Test
osu_mbw_mr - Multiple Bandwidth / Message Rate Test
osu_multi_lat - Multi-pair Latency Test
----------------------------------------------------------------------------

------------------------- COLLECTIVE MPI BENCHMARKS -------------------------
osu_allgather - MPI_Allgather Latency Test
osu_allgatherv - MPI_Allgatherv Latency Test
osu_allreduce - MPI_Allreduce Latency Test
osu_alltoall - MPI_Alltoall Latency Test
osu_alltoallv - MPI_Alltoallv Latency Test
osu_barrier - MPI_Barrier Latency Test
osu_bcast - MPI_Bcast Latency Test
osu_gather - MPI_Gather Latency Test
osu_gatherv - MPI_Gatherv Latency Test
osu_reduce - MPI_Reduce Latency Test
osu_reduce_scatter - MPI_Reduce_scatter Latency Test
osu_scatter - MPI_Scatter Latency Test
osu_scatterv - MPI_Scatterv Latency Test
----------------------------------------------------------------------------

----------------- NON-BLOCKING COLLECTIVE MPI BENCHMARKS -----------------
osu_iallgather - MPI_Iallgather Latency Test
osu_iallgatherv - MPI_Iallgatherv Latency Test
osu_iallreduce - MPI_Iallreduce Latency Test
osu_ialltoall - MPI_Ialltoall Latency Test
osu_ialltoallv - MPI_Ialltoallv Latency Test
osu_ialltoallw - MPI_Ialltoallw Latency Test
osu_ibarrier - MPI_Ibarrier Latency Test
osu_ibcast - MPI_Ibcast Latency Test
osu_igather - MPI_Igather Latency Test
osu_igatherv - MPI_Igatherv Latency Test
osu_ireduce - MPI_Ireduce Latency Test
osu_iscatter - MPI_Iscatter Latency Test
osu_iscatterv - MPI_Iscatterv Latency Test
----------------------------------------------------------------------------

------------------------- ONE-SIDED MPI BENCHMARKS -------------------------
osu_put_latency - Latency Test for Put with Active/Passive Synchronization
```

(continues on next page)

(continued from previous page)

```
osu_get_latency - Latency Test for Get with Active/Passive Synchronization
osu_put_bw - Bandwidth Test for Put with Active/Passive Synchronization
osu_get_bw - Bandwidth Test for Get with Active/Passive Synchronization
osu_put_bibw - Bi-directional Bandwidth Test for Put with Active Synchronization
osu_acc_latency - Latency Test for Accumulate with Active/Passive Synchronization
osu_cas_latency - Latency Test for Compare and Swap with Active/Passive␣
↪Synchronization
osu_fop_latency - Latency Test for Fetch and Op with Active/Passive Synchronization
osu_get_acc_latency - Latency Test for Get_accumulate with Active/Passive␣
↪Synchronization
----------------------------------------------------------------------------

For more information please refer to http://mvapich.cse.ohio-state.edu/benchmarks/
```

There is a YAML file for the OSU benchmark that can be found at https://raw.githubusercontent.com/HPC-buildtest/buildtest-configs/master/buildtest/benchmark/osu.yaml. This is the default configuration file that will be used, you may specify an alternative file using the *-c* option.

A description of all the yaml keys can be found by using the `-i` or `--info` option. Each of these options correspond to a particular option found in the test suite.

```
(buildtest) [siddis14@adwnode1 buildtest-framework]$ buildtest benchmark osu -i
Check Configuration

  Keys               |    Description
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  proc               |    Number of MPI Processes
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  min_message_size   |    set the minimum and/or the maximum message size to MIN and/
↪or MAX bytes respectively
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  max_message_size   |    set the minimum and/or the maximum message size to MIN and/
↪or MAX bytes respectively
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  max_mem_per_proc   |    set per process maximum memory consumption to SIZE bytes␣
↪(default 536870912)
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  iter_msg_size      |    set iterations per message size to ITER (default 1000 for␣
↪small messages, 100 for large messages)
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  warmup_iter        |    set number of warmup iterations to skip before timing␣
↪(default 200)
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  full_format        |    print full format listing (MIN/MAX latency and ITERATIONS␣
↪displayed in addition to AVERAGE latency)
---------------------+-------------------------------------------------------------
↪---------------------------------------------
  calls              |    set the number of MPI_Test() calls during the dummy␣
↪computation, set CALLS to 100, 1000, or any number > 0.
---------------------+-------------------------------------------------------------
↪---------------------------------------------
```

(continues on next page)

(continued from previous page)

```
  iter                |    number of iterations for timing (default 100)
--------------------+------------------------------------------------------------------
↪-----------------------------------------------
  win_option          |    <win_option> can be any of the follows:
                      |    create             use MPI_Win_create to create an MPI Window␣
↪object
                      |    allocate           use MPI_Win_allocate to create an MPI␣
↪Window object
                      |    dynamic            use MPI_Win_create_dynamic to create an␣
↪MPI Window object
--------------------+------------------------------------------------------------------
↪-----------------------------------------------
  sync_option         |    <sync_option> can be any of the follows:
                      |    pscw               use Post/Start/Complete/Wait␣
↪synchronization calls
                      |    fence              use MPI_Win_fence synchronization call
                      |    lock               use MPI_Win_lock/unlock synchronizations␣
↪calls
                      |    flush              use MPI_Win_flush synchronization call
                      |    flush_local        use MPI_Win_flush_local synchronization␣
↪call
                      |    lock_all           use MPI_Win_lock_all/unlock_all␣
↪synchronization calls
--------------------+------------------------------------------------------------------
↪-----------------------------------------------
  threads             |    number of recv threads to test with (min: 1, default: 2,␣
↪max: 128)
--------------------+------------------------------------------------------------------
↪-----------------------------------------------
```

To run the benchmark just specify the `-r` or `--run` option and it will run all the test and return an output `.run` file with the results.

Note: It will take a couple minutes to finish all of the tests. Please be patient!

```
(buildtest) [siddis14@adwnode1 buildtest-framework]$ buildtest benchmark osu -r
Check Configuration
Reading Yaml file: /home/siddis14/github/buildtest-configs/buildtest/benchmark/osu.
↪yaml

Loading YAML content

Parsing YAML content ...
Tests Generation complete. All tests are written under /tmp/osu*

osu_allgather[ /tmp/osu_allgather.sh ]         [RUNNING]
osu_allgather[ /tmp/osu_allgather.sh ]         [PASSED]
osu_allgatherv[ /tmp/osu_allgatherv.sh ]         [RUNNING]
osu_allgatherv[ /tmp/osu_allgatherv.sh ]         [PASSED]
osu_allreduce[ /tmp/osu_allreduce.sh ]         [RUNNING]
osu_allreduce[ /tmp/osu_allreduce.sh ]         [PASSED]
osu_alltoall[ /tmp/osu_alltoall.sh ]         [RUNNING]
osu_alltoall[ /tmp/osu_alltoall.sh ]         [PASSED]
osu_alltoallv[ /tmp/osu_alltoallv.sh ]         [RUNNING]
osu_alltoallv[ /tmp/osu_alltoallv.sh ]         [PASSED]
```

(continues on next page)

(continued from previous page)

```
osu_barrier[ /tmp/osu_barrier.sh ]          [RUNNING]
osu_barrier[ /tmp/osu_barrier.sh ]          [PASSED]
osu_bcast[ /tmp/osu_bcast.sh ]          [RUNNING]
osu_bcast[ /tmp/osu_bcast.sh ]          [PASSED]
osu_bibw[ /tmp/osu_bibw.sh ]          [RUNNING]
osu_bibw[ /tmp/osu_bibw.sh ]          [PASSED]
osu_bw[ /tmp/osu_bw.sh ]          [RUNNING]
osu_bw[ /tmp/osu_bw.sh ]          [PASSED]
osu_cas_latency[ /tmp/osu_cas_latency.sh ]          [RUNNING]
osu_cas_latency[ /tmp/osu_cas_latency.sh ]          [PASSED]
osu_fop_latency[ /tmp/osu_fop_latency.sh ]          [RUNNING]
osu_fop_latency[ /tmp/osu_fop_latency.sh ]          [PASSED]
osu_gather[ /tmp/osu_gather.sh ]          [RUNNING]
osu_gather[ /tmp/osu_gather.sh ]          [PASSED]
osu_gatherv[ /tmp/osu_gatherv.sh ]          [RUNNING]
osu_gatherv[ /tmp/osu_gatherv.sh ]          [PASSED]
osu_acc_latency[ /tmp/osu_acc_latency.sh ]          [RUNNING]
osu_acc_latency[ /tmp/osu_acc_latency.sh ]          [PASSED]
osu_get_bw[ /tmp/osu_get_bw.sh ]          [RUNNING]
osu_get_bw[ /tmp/osu_get_bw.sh ]          [PASSED]
osu_iallgather[ /tmp/osu_iallgather.sh ]          [RUNNING]
osu_iallgather[ /tmp/osu_iallgather.sh ]          [PASSED]
osu_iallgatherv[ /tmp/osu_iallgatherv.sh ]          [RUNNING]
osu_iallgatherv[ /tmp/osu_iallgatherv.sh ]          [PASSED]
osu_ialltoall[ /tmp/osu_ialltoall.sh ]          [RUNNING]
osu_ialltoall[ /tmp/osu_ialltoall.sh ]          [PASSED]
osu_ialltoallv[ /tmp/osu_ialltoallv.sh ]          [RUNNING]
osu_ialltoallv[ /tmp/osu_ialltoallv.sh ]          [PASSED]
osu_ialltoallw[ /tmp/osu_ialltoallw.sh ]          [RUNNING]
osu_ialltoallw[ /tmp/osu_ialltoallw.sh ]          [PASSED]
osu_ibarrier[ /tmp/osu_ibarrier.sh ]          [RUNNING]
osu_ibarrier[ /tmp/osu_ibarrier.sh ]          [PASSED]
osu_ibcast[ /tmp/osu_ibcast.sh ]          [RUNNING]
osu_ibcast[ /tmp/osu_ibcast.sh ]          [PASSED]
osu_igather[ /tmp/osu_igather.sh ]          [RUNNING]
osu_igather[ /tmp/osu_igather.sh ]          [PASSED]
osu_igatherv[ /tmp/osu_igatherv.sh ]          [RUNNING]
osu_igatherv[ /tmp/osu_igatherv.sh ]          [PASSED]
osu_iscatter[ /tmp/osu_iscatter.sh ]          [RUNNING]
osu_iscatter[ /tmp/osu_iscatter.sh ]          [PASSED]
osu_iscatterv[ /tmp/osu_iscatterv.sh ]          [RUNNING]
osu_iscatterv[ /tmp/osu_iscatterv.sh ]          [PASSED]
osu_latency[ /tmp/osu_latency.sh ]          [RUNNING]
osu_latency[ /tmp/osu_latency.sh ]          [PASSED]
osu_latency_mt[ /tmp/osu_latency_mt.sh ]          [RUNNING]
osu_latency_mt[ /tmp/osu_latency_mt.sh ]          [PASSED]
osu_multi_lat[ /tmp/osu_multi_lat.sh ]          [RUNNING]
osu_multi_lat[ /tmp/osu_multi_lat.sh ]          [PASSED]
osu_put_bibw[ /tmp/osu_put_bibw.sh ]          [RUNNING]
osu_put_bibw[ /tmp/osu_put_bibw.sh ]          [PASSED]
osu_put_bw[ /tmp/osu_put_bw.sh ]          [RUNNING]
osu_put_bw[ /tmp/osu_put_bw.sh ]          [PASSED]
osu_put_latency[ /tmp/osu_put_latency.sh ]          [RUNNING]
osu_put_latency[ /tmp/osu_put_latency.sh ]          [PASSED]
osu_reduce[ /tmp/osu_reduce.sh ]          [RUNNING]
osu_reduce[ /tmp/osu_reduce.sh ]          [PASSED]
osu_reduce_scatter[ /tmp/osu_reduce_scatter.sh ]          [RUNNING]
```

(continues on next page)

(continued from previous page)

```
osu_reduce_scatter[ /tmp/osu_reduce_scatter.sh ]         [PASSED]
osu_scatter[ /tmp/osu_scatter.sh ]          [RUNNING]
osu_scatter[ /tmp/osu_scatter.sh ]          [PASSED]
osu_scatterv[ /tmp/osu_scatterv.sh ]          [RUNNING]
osu_scatterv[ /tmp/osu_scatterv.sh ]          [PASSED]
Writing Test Results to /tmp/buildtest_16_53_25_01_2019.run
```

The benchmark submenu is further broken down into another sub-menu (one per benchmark). Currently, buildtest supports OSU benchmark.

```
(siddis14-TgVBs13r) buildtest-framework[master !?] $ buildtest benchmark --help
usage: buildtest [options] benchmark [-h] {osu,hpl,hpcg} ...

positional arguments:
  {osu,hpl,hpcg}  subcommand help
    osu           OSU MicroBenchmark sub menu
    hpl           High Performance Linpack sub menu
    hpcg          High Performance Conjugate Gradient sub menu

optional arguments:
  -h, --help      show this help message and exit
```

CHAPTER

ELEVEN

YAML SUBCOMMAND

11.1 Yaml Options (`buildtest yaml --help`)

```
usage: buildtest yaml [-h] [--ohpc] [-m {yes,no}]

optional arguments:
  -h, --help            show this help message and exit
  --ohpc                Build YAML configuration for OpenHPC package. YAML files will␣
↪be written in
                        $BUILDTEST_CONFIGS_REPO/ohpc
  -m {yes,no}, --maintainer {yes,no}
                        Add as maintainer to test
```

CHAPTER

TWELVE

COMMAND REFERENCE

CHAPTER

THIRTEEN

CONTRIBUTING GUIDE

This document explains how you can contribute back to buildtest.

There are many ways you can help contribute to buildtest that may include

- File an issue with the framework
- Proofread documentation and report or fix issues
- Participate in discussions and join the slack channel
- Share your tests
- Provide feedback on buildtest options.
    1. What features you *like/dislike*
    2. What features you would like to have
    3. What testing capabilities matter most for you

13.1 Reporting an Issue

Please report all issues regarding the framework at https://github.com/HPC-buildtest/buildtest-framework/issues

13.2 Reference to CONTRIBUTING guides

Please see the following contributing guide for additional reference

- Contributing To buildtest Framework
- Test Contribution Guide

CHAPTER

FOURTEEN

REFERENCES

14.1 Conference

Title	Conference
Software Stack Testing with buildtest	HPCKP18
HPC Application Testing Framework - buildtest	HPCKP17

14.2 Article

- https://www.hpcwire.com/2019/01/17/pfizer-hpc-engineer-aims-to-automate-software-stack-testing/

CHAPTER

FIFTEEN

SLACK

Join the Slack Channel to get connected. The slack channel can be accessed at http://hpcbuildtest.slack.com

CHAPTER

SIXTEEN

RELATED PROJECTS

- ReFrame: Re gression FRAME work for Software Testing
- EasyBuild: end-end software build framework with 1000+ software used for building tuned application for HPC clusters
- Spack: is S upercomputing PACK age Manager that supports 1000+ software packages and extremely efficient in combinatorial builds

CHAPTER

SEVENTEEN

INDICES AND TABLES

- genindex
- modindex
- search