

EasyBuild 2.1 on Cray Linux Environment (Experimental)

Petar Forai

Current Status as of 2.1

- Some big apps work and most of the machinery is there
- Marked as ‘experimental’, ie current interfaces and functionality could be changed in future (waiting for feedback from you guys!)
- Needs —experimental flag to get EB to use this
- Updated status page at

<https://github.com/hpcugent/easybuild/wiki/EasyBuild-on-Cray>

Short Introduction to Cray Development Environment

- Cray provides module files for the “products” that are shipped with the system (like compilers, Cray libs, MPI, network drivers, etc.)
- Out of box install contains several hundred of those. The module files are quite sophisticated, complex and sometimes buggy (syntax errors)
- Cray Linux Environment comes with PrgEnv-* meta-modules for ‘whole stack’ environment setup (compiler, libs, MPI, et al)
- One for each of GNU/Intel/Cray/PGI
- PrgEnv modules provide compiler ‘drivers’/wrappers that provide the user with `cc/cc/ftn` executables (C/C++/Fortran respectively)

- The PrgEnv wrapper drivers also contain the real compiler binaries and make them available for use as well (ifort, icpc, (crayftn, craycc,crayCC in older PrgEnv), gcc, g++,gfortran, etc.)
- System GCC (SLES11 GCC 4.3.x for CLE 5.x) is usually visible in the non PrgEnv-GNU modules (on SuSE cc is a symlink to gcc) but is pushed back in \$PATH
- Invoking the compilers directly means that you need to explicitly set libarriers and include paths. Like you would on any regular Linux cluster.
- Compiler drivers read pkg-config .pc files that are shipped with the installed products to generate appropriate linker and include file search paths

default set of modules output from Prg-Env-cray's cc

```
forai@sisu-login1:~> cc -craype-verbose
```

```
driver.cc -hcpu=haswell -hstatic -D__CRAYXC -D__CRAY_HASWELL -D__CRAYXT_COMPUTE_LINUX_TARGET -hnetwork=aries -wl,--rpath=/opt/cray/cce/8.3.7/craylibs/x86-64 -hlast_user_arg -nostdinc -ibase-compiler /opt/cray/cce/8.3.7/CC/x86-64/compiler_include_base -isystem /opt/cray/cce/8.3.7/craylibs/x86-64/include -I/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/include -I/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/include-fixed -L/opt/cray/cce/8.3.7/CC/x86-64/lib/x86-64 -wl,-rpath=/opt/cray/cce/8.3.7/CC/x86-64/lib/x86-64 -L/opt/gcc/4.8.1/snos/lib64 -wl,-rpath=/opt/cray/gcc-libs/ -ugcc_base=/opt/gcc/4.8.1/snos -usysroot=/ -uno_driver_libs -I/opt/cray/mpt/7.1.1/gni/mpich2-cray/83/include -I/opt/cray/libsci/13.0.1/CRAY/83/haswell/include -I/opt/cray/rca/1.0.0-2.0502.51491.3.92.ari/include -I/opt/cray/alps/5.2.1-2.0502.8712.10.32.ari/include -I/opt/cray/xpmem/0.1-2.0502.51169.1.11.ari/include -I/opt/cray/gni-headers/3.0-1.0502.9038.7.4.ari/include -I/opt/cray/dmapp/7.0.1-1.0502.9080.9.32.ari/include -I/opt/cray/pmi/5.0.6-1.0000.10439.140.2.ari/include -I/opt/cray/ugni/5.0-1.0502.9037.7.26.ari/include -I/opt/cray/udreg/2.3.2-1.0502.8763.1.11.ari/include -I/opt/cray/cce/8.3.7/craylibs/x86-64/pkgconfig/./include -I/opt/cray/cce/8.3.7/craylibs/x86-64/include -I/opt/cray/wlm_detect/1.0-1.0502.51217.1.1.ari/include -I/opt/cray/krca/1.0.0-2.0502.52773.6.7.ari/include -I/opt/cray-hss-devel/7.2.0/include -L/opt/cray/dmapp/default/lib64 -L/opt/cray/mpt/7.1.1/gni/mpich2-cray/83/lib -L/opt/cray/libsci/13.0.1/CRAY/83/haswell/lib -L/opt/cray/rca/1.0.0-2.0502.51491.3.92.ari/lib64 -L/opt/cray/alps/5.2.1-2.0502.8712.10.32.ari/lib64 -L/opt/cray/xpmem/0.1-2.0502.51169.1.11.ari/lib64 -L/opt/cray/dmapp/7.0.1-1.0502.9080.9.32.ari/lib64 -L/opt/cray/pmi/5.0.6-1.0000.10439.140.2.ari/lib64 -L/opt/cray/ugni/5.0-1.0502.9037.7.26.ari/lib64 -L/opt/cray/udreg/2.3.2-1.0502.8763.1.11.ari/lib64 -L/opt/cray/atp/1.7.5/lib -L/opt/cray/cce/8.3.7/craylibs/x86-64 -L/opt/cray/wlm_detect/1.0-1.0502.51217.1.1.ari/lib64 -lAtpSigHandler -lAtpSigHCommData -wl,--undefined=__ATP_Data_Globals -wl,--undefined=__atpHandlerInstall -lpthread -lsci_cray_mpi_mp -lm -lf -lmpich_cray -lrt -lpthread -lugni -lpmi -lsci_cray_mp -lcraymp -lm -lpthread -lf -lpgas-dmapp -lfi -lu -lrt -ldmapp -lugni -ludreg -lpthread -lm -lcray-c++-rts -lcraystdc++ -lxpmem -ldmapp -lpthread -lpmi -lpthread -lalpslli -lpthread -lwlm_detect -lugni -lpthread -lalpsutil -lpthread -lrca -ludreg -lomp -lcraymp -lpthread -lrt -lmodules -lm -lfi -lm -lquadmath -lcraymath -lm -lgfortran -lquadmath -lf -lm -lpthread -lu -lrt -lcsup -wl,--whole-archive,-ltcmalloc_minimal,--no-whole-archive -lstdc++ -lpthread -wl,--start-group,-lc,-lcsup,-lgcc_eh,-lm,-lgcc,--end-group -wl,-T/opt/cray/cce/8.3.7/craylibs/x86-64/2.23.1.cce.ld
```

after module load cray-hdf5/1.8.13

```
forai@sisu-login1:~> cc -craype-verbose
```

```
driver.cc -hcpu=haswell -hstatic -D__CRAYXC -D__CRAY_HASWELL -D__CRAYXT_COMPUTE_LINUX_TARGET -hnetwork=aries -wl,--rpath=/opt/cray/cce/8.3.7/craylibs/x86-64 -hlast_user_arg -nostdinc -ibase-compiler /opt/cray/cce/8.3.7/CC/x86-64/compiler_include_base -isystem /opt/cray/cce/8.3.7/craylibs/x86-64/include -I/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/include -I/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/include-fixed -L/opt/cray/cce/8.3.7/CC/x86-64/lib/x86-64 -wl,-rpath=/opt/cray/cce/8.3.7/CC/x86-64/lib/x86-64 -L/opt/gcc/4.8.1/snos/lib64 -wl,-rpath=/opt/cray/gcc-libs/ -ugcc_base=/opt/gcc/4.8.1/snos -usysroot=/ -uno_driver_libs -I/opt/cray/hdf5/1.8.13/CRAY/83/include -I/opt/cray/mpt/7.1.1/gni/mpich2-cray/83/include -I/opt/cray/libsci/13.0.1/CRAY/83/haswell/include -I/opt/cray/rca/1.0.0-2.0502.51491.3.92.ari/include -I/opt/cray/alps/5.2.1-2.0502.8712.10.32.ari/include -I/opt/cray/xpmem/0.1-2.0502.51169.1.11.ari/include -I/opt/cray/gni-headers/3.0-1.0502.9038.7.4.ari/include -I/opt/cray/dmapp/7.0.1-1.0502.9080.9.32.ari/include -I/opt/cray/pmi/5.0.6-1.0000.10439.140.2.ari/include -I/opt/cray/ugni/5.0-1.0502.9037.7.26.ari/include -I/opt/cray/udreg/2.3.2-1.0502.8763.1.11.ari/include -I/opt/cray/cce/8.3.7/craylibs/x86-64/pkgconfig/./include -I/opt/cray/cce/8.3.7/craylibs/x86-64/include -I/opt/cray/wlm_detect/1.0-1.0502.51217.1.1.ari/include -I/opt/cray/krca/1.0.0-2.0502.52773.6.7.ari/include -I/opt/cray-hss-devel/7.2.0/include -L/opt/cray/hdf5/1.8.13/CRAY/83/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/mpt/7.1.1/gni/mpich2-cray/83/lib -L/opt/cray/libsci/13.0.1/CRAY/83/haswell/lib -L/opt/cray/rca/1.0.0-2.0502.51491.3.92.ari/lib64 -L/opt/cray/alps/5.2.1-2.0502.8712.10.32.ari/lib64 -L/opt/cray/xpmem/0.1-2.0502.51169.1.11.ari/lib64 -L/opt/cray/dmapp/7.0.1-1.0502.9080.9.32.ari/lib64 -L/opt/cray/pmi/5.0.6-1.0000.10439.140.2.ari/lib64 -L/opt/cray/ugni/5.0-1.0502.9037.7.26.ari/lib64 -L/opt/cray/udreg/2.3.2-1.0502.8763.1.11.ari/lib64 -L/opt/cray/atp/1.7.5/lib -L/opt/cray/cce/8.3.7/craylibs/x86-64 -L/opt/cray/wlm_detect/1.0-1.0502.51217.1.1.ari/lib64 -lhdf5_hl -lz -ldl -lm -lrt -lhdf5 -lz -ldl -lm -lrt -lAtpSigHandler -lAtpSigHCommData -wl,--undefined=__ATP_Data_Globals -wl,--undefined=__atpHandlerInstall -lpthread -lsci_cray_mpi_mp -lm -lf -lmpich_cray -lrt -lpthread -lugni -lpmi -lsci_cray_mp -lcraymp -lm -lpthread -lf -lpgas-dmapp -lfi -lu -lrt -ldmapp -lugni -ludreg -lpthread -lm -lcray-c++-rts -lcraystdc++ -lxpmem -ldmapp -lpthread -lpmi -lpthread -lalpslli -lpthread -lwlm_detect -lugni -lpthread -lalpsutil -lpthread -lrca -ludreg -lomp -lcraymp -lpthread -lrt -lmodules -lm -lfi -lm -lquadmath -lcraymath -lm -lgfortran -lquadmath -lf -lm -lpthread -lu -lrt -lcsup -wl,--whole-archive,-ltcmalloc_minimal,--no-whole-archive -lstdc++ -lpthread -wl,--start-group,-lc,-lcsup,-lgcc_eh,-lm,-lgcc,--end-group -wl,-T/opt/cray/cce/8.3.7/craylibs/x86-64/2.23.1.cce.ld
```

- They accept environment variables and flags for setting build options like backend code generator (Haswell vs IvyBridge or static vs dynamic building)
- Cray provides module files for some settings of the compiler like backend code generator targets
- The PrgEnv wrappers are very similar to EasyBuild toolchains except that they make it easier for the user to run the compilers and not have to specify linker and search paths (EB sets `$C_PATH` and `$LIBRARY_PATH` so for GCC and Intel this is pretty much the same)

The EasyBuild way

EasyBuild approach for toolchains

- Toolchain definitions are composition of ‘full stack’ development environments (compilers, MPI, BLAS/LAPACK, FFTW, etc.)
- Toolchain code in framework exposes things like `$CC`, `$CXX`, `$FTN`, `$CFLAGS`, `$CXXFLAGS`, etc. when running
- There’s also API in the toolchains to construct compiler call strings - mostly used in complex blocks
- Got to <https://github.com/hpcugent/easybuild-easyconfigs/search?utf8=%E2%9C%93&q=CFLAGS>
- and <https://github.com/hpcugent/easybuild-easyblocks/search?utf8=%E2%9C%93&q=CFLAGS>

How are EB deps handled found while building?

- EasyBuild sets `$CPATH` and `$LIBRARY_PATH` in the module files it creates
- Honored by GCC and Intel and ignored by Cray's compilers
- use `$EBROOT<FOO>` with `-I`
`$EBROOT<FOO>/include` and `-L`
`$EBROOT<FOO>/lib` when setting
`$CFLAGS/$CXXFLAGS` either in configs
or `get_software_root('Python')`
in blocks

```
whatis("Description: HDF5 is a unique
technology suite that makes possible the
management of extremely large and complex
data co
llections. - Homepage: http://
www.hdfgroup.org/HDF5/ ")
conflict("HDF5")
load("cpeg/5.2.25")
load("zlib/1.2.8-cpeg-5.2.25")
load("Szip/2.1-cpeg-5.2.25")
prepend_path("CPATH", "/proj/cle/apps/
software/HDF5/1.8.13-cpeg-5.2.25/include")
prepend_path("LD_LIBRARY_PATH", "/proj/cle/
apps/software/HDF5/1.8.13-cpeg-5.2.25/lib")
prepend_path("LIBRARY_PATH", "/proj/cle/
apps/software/HDF5/1.8.13-cpeg-5.2.25/lib")
prepend_path("PATH", "/proj/cle/apps/
software/HDF5/1.8.13-cpeg-5.2.25/bin")
setenv("EBROOTHDF5", "/proj/cle/apps/
software/HDF5/1.8.13-cpeg-5.2.25")
setenv("EBVERSIONHDF5", "1.8.13")
setenv("EBDEVELHDF5", "/proj/cle/apps/
software/HDF5/1.8.13-cpeg-5.2.25/easybuild/
HDF5-1.8.13-cpeg-5.2.25-easybuild-devel")
help([[ HDF5 is a unique technology suite
that makes possible the management of
extremely large and complex data
collections. - Homepage: http://
www.hdfgroup.org/HDF5/

]])
```

Mapping Cray PrgEnv(s) to EasyBuild concepts

- New proper toolchains for `PrgEnv-{gnu|intel|cray}`, they just load `PrgEnv-*` and `fftw` - which makes this similar to other TCs in terms of compiler, MPI, BLAS/LAPACK, `fftw`
- Named `Cray{Gnu|Intel|CCE}-<PrgEnv-version>`
- Install the resulting configs to have the toolchains available for building
- Setup EB env vars that control which code gen/arch specific optimizations the compiler is going to use

External Modules

- External modules supported in 2.1 via meta data description
- Maps software available on system to software available within EasyBuild useful to integrate Cray provided software in EB (HDF5, third party libraries, etc.)
- Example <https://gist.github.com/pforai/d0f25022b7925792f9f7>

'Conceptual' Issues with EB

- EB assumes it can execute tests where it is running (can be disabled)
- Executing and/or benchmarking build artefacts (ie numpy for np.dot()) needs to run <1s) after the build
- “Cross compilation” works with loading craype-haswell or craype-ivybridge or others if machine has different CPUs in login nodes vs backend nodes via the `—operarch=<target>` flag

What works

- Some large apps work as expected, like CP2K, GROMACS, Python with numpy and scipy linked against libsci and cray-mpich for mpi4py (using cray supplied software as dependencies)
- All login nodes software (git, mercurial, Autotools, cURL, Qt4/5, ...)
- Simple to medium simple software for Bioinformatics, Math libraries, etc.

What is missing

- Most work has gone into getting PrgEnv-gnu and PrgEnv-intel to run, other PrgEnvs are more special
- Generating pkg-config .pc files at end of build to have the wrappers set proper -I and -L -I for 3rd party software (WIP) (workaround available)
- Cleaning ups configs and blocks - see <https://github.com/hpcugent/easybuild-easyconfigs/search?utf8=%E2%9C%93&q=CFLAGS> (there are assumptions in (some) places that the compilers running accept GCC flags)

Requirements

- Tcl/C Environment Modules `3.2.10` or `>= Lmod 5.6.3`
- This is basically a requirement of EasyBuild as default Tcl/C modules `=< 3.2.10` produce a **SIGSEGV** when loading a lot of modules at once (see https://bugzilla.redhat.com/show_bug.cgi?id=834580 if you suffer from this)
- This can be relaxed in EB code if case you know that your modules implementation is fine and will not segfault
- Lmod can consume Cray module files! (Thanks and Kudos to Robert McLay for this!)
- CLE 5.x with `>=craype-2.0` because we only have XC40 systems for developing EB

Open Questions

- Switch between wrapper and true compiler binary - How frequently do you use this?
- How to link dynamically - expose `$CRAYPE_LINK_TYPE=dynamic/static` or `cc -dynamic` or `-shared`
- `PrgEnv-foo` doesn't fix the `libsci` version, just loads system's default so versions of `TC` are not fixed

Open Questions II

- What to do with libsci? EB normally pins all the version of all dependencies. PrgEnv-foo just loads system default for libsci (unlinke other deps in PrgEnv modules)
- On one machine this means that CrayGNU-5.2.25 contains ie libsci 13.0.3 and on another it is 13.0.1
- Current assumption is 'site knows what it's doing and is checking for compatiblity'
- Should be pin or parametrise?
- Provide feedback for EB built software in terms of performance vs your hand compiled stacks (like for large CP2K runs, GROMACs, etc.)