Building Applications on Edison

Jack Deslippe
NERSC User Services
10/10/2013
Outline

- Applications Already Available at NERSC
- Available Compilers
- Available Libraries
Did you know that NERSC offers precompiled executables for more than 100 applications?
Did you know that NERSC offers precompiled executables for more than 100 applications?

Example, Materials Science:

Quantum ESPRESSO, NAMD, LAMMPS, NWChem, VASP, BerkeleyGW, SIESTA, Abinit, Gamess, GROMACS, GPAW MEEP, cpmd, libxc, etsf_io, atompaw, Wiek2K, Gaussian, PARATEC, cp2k, Wannier90, Amber, Yambo, XCrysdan, Q-Chem
The NERSC Software Database on the web shows all of our available pre-compiled applications for Hopper, Edison, Carver.

### Pre-compiled Applications

<table>
<thead>
<tr>
<th>#</th>
<th>Package</th>
<th>Platform</th>
<th>Category</th>
<th>Version</th>
<th>Module</th>
<th>Install Date</th>
<th>Date Made Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ABINIT</td>
<td>carver</td>
<td>applications/material sciences</td>
<td>6.0.3</td>
<td>abinit/6.0.3</td>
<td>2010-05-06</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ABINIT</td>
<td>carver</td>
<td>applications/material sciences</td>
<td>6.10.3</td>
<td>abinit/6.10.3</td>
<td>2012-01-10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ABINIT</td>
<td>carver</td>
<td>applications/material sciences</td>
<td>6.2.2</td>
<td>abinit/6.2.2</td>
<td>2010-08-16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ABINIT</td>
<td>carver</td>
<td>applications/material sciences</td>
<td>6.4.1</td>
<td>abinit/6.4.1</td>
<td>2010-11-24</td>
<td>2012-01-12</td>
</tr>
<tr>
<td>4</td>
<td>ABINIT</td>
<td>franklin</td>
<td>applications/material sciences</td>
<td>5.6</td>
<td>abinit/5.6</td>
<td>2008-12-03</td>
<td>2008-12-03</td>
</tr>
<tr>
<td>5</td>
<td>ABINIT</td>
<td>franklin</td>
<td>applications/material sciences</td>
<td>5.8.4</td>
<td>abinit/5.8.4</td>
<td>2009-12-06</td>
<td>2009-12-15</td>
</tr>
<tr>
<td>6</td>
<td>ABINIT</td>
<td>franklin</td>
<td>applications/material sciences</td>
<td>6.10.3</td>
<td>abinit/6.10.3</td>
<td>2012-01-10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ABINIT</td>
<td>franklin</td>
<td>applications/material sciences</td>
<td>6.4.3</td>
<td>abinit/6.4.3</td>
<td>2011-02-22</td>
<td>2011-03-24</td>
</tr>
<tr>
<td>8</td>
<td>ABINIT</td>
<td>hopper</td>
<td>applications/material sciences</td>
<td>6.10.3</td>
<td>abinit/6.10.3</td>
<td>2012-01-09</td>
<td></td>
</tr>
</tbody>
</table>
Many application pages contain compilation instructions

e.g. Abinit

Compilation Instructions

Some advanced users may be interested in tweaking the Abinit build parameters and building Abinit themselves in their own directory. In order to aid in this process, and to provide a greater degree of transparency, the build instructions for the Abinit module are listed below. The following procedure was used to build Abinit 6.8.2 on Hopper.

% module swap PrgEnv-pgi PrgEnv-gnu

% module swap gcc gcc/4.5.2

% module load netcdf atompaw etsf_io wannier90 libxc

% ./configure --prefix="pwd/.." FC=ftn CC=CC CXX=CC FCFLAGS="-O3" CFLAGS="-O3" CXXFLAGS="-O3" --with-fc-vendor=gn

% make

% make install
Available Compilers

Compilers vs. Compiler Wrappers

ifort, icc vs ftn, cc, CC

The compiler wrappers are the same as the underlying compilers with the addition of flags included by default and libraries linked by default (like MPI libraries for example)

The same compiler wrapper command (e.g. ftn) can refer to any underlying compiler available on the system (e.g. pgi, gnu, intel etc...)
## Available Compilers Across Machines:

<table>
<thead>
<tr>
<th></th>
<th>Edison</th>
<th>Hopper</th>
<th>Carver</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGI</td>
<td></td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>GNU</td>
<td>❌</td>
<td></td>
<td>❌</td>
</tr>
<tr>
<td>Intel</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Pathscale</td>
<td></td>
<td>❌</td>
<td></td>
</tr>
<tr>
<td>Cray</td>
<td>❌</td>
<td>❌</td>
<td></td>
</tr>
</tbody>
</table>

Default
### Available Compilers

#### Available Compilers Across Machines:

<table>
<thead>
<tr>
<th></th>
<th>Edison</th>
<th>Hopper</th>
<th>Carver</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGI</td>
<td></td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>GNU</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Intel</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Pathscale</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Cray</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>

**Default**

Edison Swap between Compilers With Modules:

```
% module swap PrgEnv-intel PrgEnv-gnu
```
# Available Compilers

## Useful Compiler Options on Edison:

<table>
<thead>
<tr>
<th></th>
<th>Intel</th>
<th>GNU</th>
<th>Cray</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimization</strong></td>
<td>-fast -no-ipo</td>
<td>-O3 -fast-math</td>
<td></td>
</tr>
<tr>
<td><strong>OpenMP</strong></td>
<td>-openmp</td>
<td>-fopenmp</td>
<td></td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>-V</td>
<td>--version</td>
<td>-V</td>
</tr>
<tr>
<td><strong>Verbose</strong></td>
<td>-v</td>
<td>-v</td>
<td>-v</td>
</tr>
<tr>
<td><strong>Debugging</strong></td>
<td>-g -warn all</td>
<td>-g -fbounds-check</td>
<td>-g -R bs</td>
</tr>
<tr>
<td><strong>Math Libs</strong></td>
<td>-mkl=cluster</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The "-v" option displays the complete link and compile line unwrapped from ftn
- The "-Wl,-y<symbol_name>" reports which library the linker is currently using for the symbol <symbol_name>"
The Intel and PGI compiler -fast options have different effects:

**PGI**
- "A generally optimal set of options is chosen for targets that support SSE capability."
- Same as -fastsse.

**Intel**
- Includes interprocedural optimization which can increase compile time by an order of magnitude or cause it to fail.
- Always turn it off with -no-ipo when using -fast.
- -fast -no-ipo generally improves performance over the default (no optimization arguments) on Edison.
- Includes vectorization. AVX (4 Double-Precision operations).
Intel and Cray compilers generally provide best performance.

MKL on Edison

Using MKL on Edison:

Add to the end of your link-line

- **-mkl=cluster** : Single-Threaded MKL with BLAS/LAPACK/scALAPACK/FFTW

- **-mkl** (same as -**mkl=parallel**) : Multi-Threaded MKL (No ScALAPACK)

- **-mkl=sequential** : Single-Threaded MKL (No ScALAPACK)
MKL on Edison

Using MKL on Edison:

Add to the end of your link-line

-mkl=cluster : Single-Threaded MKL with BLAS/LAPACK/scalAPACK/FFTW

-mkl (same as -mkl=parallel) : Multi-Threaded MKL (No ScalAPACK)

-mkl=sequential : Single-Threaded MKL (No ScalAPACK)

In order to link to multi-threaded MKL libraries including ScalAPACK must add:

$(MKLROOT)/lib/intel64/libmkl_scalapack_lp64.a -Wl,--start-group $(MKLROOT)
/lib/intel64/libmkl_intel_lp64.a $(MKLROOT)/lib/intel64/libmkl_intel_thread.a $(MKLROOT)
/lib/intel64/libmkl_core.a $(MKLROOT)/lib/intel64/libmkl_blacs_intelmpi_lp64.a  -Wl,--end-group -lpthread -lm

For linking with other compilers than intel. Use the following online tool: http://software.intel.com/sites/products/mkl/MKL_Link_Line_Advisor.html
MKL on Edison

Using MKL on Edison:

Add to the end of your link-line

-mkl=cluster

-Single-Threaded MKL with BLAS/LAPACK/scaLAPACK/FFTW

-mkl (same as -mkl=parallel)

-Multi-Threaded MKL (No ScaLAPACK)

-mkl=sequential

-Single-Threaded MKL (No ScaLAPACK)

In order to link to multi-threaded MKL libraries including ScaLAPACK must add:

$(MKLROOT)/lib/intel64/libmkl_scalapack_lp64.a -Wl,--start-group $(MKLROOT)/lib/intel64/libmkl_intel_lp64.a $(MKLROOT)/lib/intel64/libmkl_intel_thread.a $(MKLROOT)/lib/intel64/libmkl_core.a $(MKLROOT)/lib/intel64/libmkl_blacs_intelmpi_lp64.a  -Wl,--end-group -lpthread -lm

For linking with other compilers than intel. Use the following online tool:

http://software.intel.com/sites/products/mkl/MKL_Link_Line_Advisor.html
MKL and LibSci give comparable dense linear algebra performance.
◆ Both provide good support with threads.

MKL FFTs consistently outperform FFTW.
◆ MKL FFTs scale well with threads. FFTW does not.
MKL gives better threaded FFTW performance than Cray provided FFTW pthread in codes that use OpenMP.
Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  edison: ftn, cc, CC not ifort, icc, gcc, gfortran ...
Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  edison: **ftn**, **cc**, **CC** not **ifort**, **icc**, **gcc**, **gfortran** ...

- **ftn** links libraries (blas etc...) by default on Hopper (need -mkl=cluster on edison with intel). mpif90 does not.
Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  edison: ftn, cc, CC not ifort, icc, gcc, gfortran ...

- ftn links libraries (blas etc...) by default on Hopper (need -mkl=cluster on edison with intel). mpif90 does not.

- Edison statically links by default (carver dynamically links)
Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  edison: ftn, cc, CC not ifort, icc, gcc, gfortran ...

- ftn links libraries (blas etc...) by default on Hopper (need -mkl=cluster on edison with intel). mpif90 does not.

- Edison statically links by default (carver dynamically links)

You can dynamically link on Edison by adding -dynamic flag to compilation and "setenv CRAY_ROOTFS DSL" in batch script and/or use CCM.
Available Compilers

Tips for using compilers:

To show libraries linked automatically with ftn

% ftn -v test.f90 -o test.x
...
Using built-in specs.

To show which library "dgemm" is used from:

% ftn -lWl,-ydgemm_ test.f90 -o test.x
/scratch/scratchdirs/jdeslip/cc8lmteH.o: reference to dgemm_
/opt/xt-libsci/11.0.03-gnu/46/mc12/lib/libsci_gnu_mp.a(dgemm.o): definition of dgemm_

Available Libraries

Modules

Software libraries at NERSC are managed in modules.

Modules add and remove executables and libraries from your $PATH and $LD_LIBRARY_PATH as well as define environment variables.

They are used by doing "module load" command

e.g. "% module load fftw"
Available Libraries

Most math and science libraries are available

e.g.:

% module avail fftw

------------------------ /opt/cray/modulefiles
------------------------
fftw/2.1.5.5          fftw/2.1.5.6          fftw/3.3.0.1
fftw/3.3.0.3          fftw/3.3.0.4(default)

% module show fftw/2.1.5.5

setenv       FFTW_DIR /opt/fftw/2.1.5.5/lib
setenv       FFTW_INC /opt/fftw/2.1.5.5/include
setenv       FFTW_POST_LINK_OPTS -L/opt/fftw/2.1.5.5/lib
setenv       FFTW_INCLUDE_OPTS -I/opt/fftw/2.1.5.5/include
prepend-path  CRAY_LD_LIBRARY_PATH /opt/fftw/2.1.5.5/lib
prepend-path  INFOPATH /opt/fftw/2.1.5.5/info
append-path   PE_PRODUCT_LIST FFTW
setenv       CRAY_FFTW_DIR /opt/fftw/2.1.5.5
Summary of Good Practices

- Use developer recommended compiler and compiler options
- Use compiler wrappers
- Test your application against lower optimization levels and included tests
- Use system provided libraries
More Information

NERSC Website:

- http://www.nersc.gov/users/computational-systems/edison/programming/

Man Pages:

% man ifort

eMail Us:

consult@nersc.gov
EXTRA SLIDES
Jack is having trouble building and running some of his favorite applications at NERSC.

Can you help fix the problem?

Rules: No looking ahead.
Prizes: Eternal glory in the minds of those around you.
Quantum ESPRESSO on Edison:

% cat README

...  
Quick installation instructions for the impatient:
. /configure [options] 
   make all  
...
Quantum ESPRESSO on Edison:

% cat README
...
Quick installation instructions for the impatient:
./configure [options]
make all
...

That seems easy enough...

% ./configure ... (Success!)
% make all ... come back in 20 minutes ... (Success!)

% aprun -n 5 ~/PresentationDir/espresso-4.3.2/bin/pw.x -in ./in

Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
This program is part of the open-source Quantum ESPRESSO suite
This program is part of the open-source Quantum ESPRESSO suite
.....
Quantum ESPRESSO on Edison:

% cat README
...
Quick installation instructions for the impatient:
  ./configure [options]
  make all
...

That seems easy enough...

% ./configure ... (Success!)
% make all ... come back in 20 minutes ... (Success!)

% aprun -n 5 ~/PresentationDir/espresso-4.3.2/bin/pw.x -in ./in

Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44
Program PWSCF v.4.3.2 starts on 25Jan2012 at 15:19:44

This program is part of the open-source Quantum ESPRESSO suite
This program is part of the open-source Quantum ESPRESSO suite
.....

The output looks weird and repeated....
Solution

Use the compiler wrappers, ftn, cc, CC. They can often be specified in configure:
./configure FC=ftn CC=CC CXX=CC or in make.sys file
Solution

Use the compiler wrappers, ftn, cc, CC. They can often be specified in configure:

```
./configure FC=ftn CC=CC CXX=CC or in make.sys file
```

```bash
% cat make.sys
.....
DFLAGS     = -D__PGI -D__ACML
FDFLAGS    = $(DFLAGS)
MPIF90     = ifort
#F90       = ifrot
CC         = icc
F77        = ifort
.....

% cat make.sys_fixed
.....
DFLAGS     = -D__PGI -D__ACML -D__MPI
FDFLAGS    = $(DFLAGS)
MPIF90     = ftn
#F90       = ftn
CC         = cc
F77        = ftn
.....
```
Round 2

BerkeleyGW on Edison

```
% cat arch.mk
...
FCPP = /usr/bin/cpp -ansi
F90free = ftn -Mfree
LINK = ftn

LAPACKLIB =
FFTWLIB =

FOPTS = -fast
FNOOPTS = $(FOPTS)
...
```
Round 2

BerkeleyGW on Edison

% cat arch.mk
...
FCPP = /usr/bin/cpp -ansi
F90free = ftn -Mfree
LINK = ftn

FOPTS = -fast
FNOOPTS = $(FOPTS)
...

% make

/global/u2/j/jdeslip/PresentationDir/BGW_2.4.x/Xi0/../Common/fftw.f90:270: undefined reference to `fftwnd_f77_create_plan_
/global/u2/j/jdeslip/PresentationDir/BGW_2.4.x/Xi0/../Common/fftw.f90:270: undefined reference to `fftwnd_f77_create_plan_
/global/u2/j/jdeslip/PresentationDir/BGW_2.4.x/Xi0/../Common/fftw.f90:285: undefined reference to `fftwnd_f77_one_
/global/u2/j/jdeslip/PresentationDir/BGW_2.4.x/Xi0/../Common/fftw.f90:287: undefined reference to `fftwnd_f77_one_
Round 2

Solution

"undefined reference" errors usually mean you are missing a library at link time. In this case, we are missing the fftw library.

Note that ftn links lapack/blas equivalents automatically.
"undefined reference" errors usually mean you are missing a library at link time. In this case, we are missing the fftw library.

Note that ftn links lapack/blas equivalents automatically.

% cat arch.mk
...
FCPP = /usr/bin/cpp -ansi
F90free = ftn -Mfree
LINK = ftn

LAPACKLIB =
FFTWLIB =

FOPTS = -fast
FNOOPTS = $(FOPTS)

% module load fftw
% cat arch.mk -fixed
...
FCPP = /usr/bin/cpp -ansi
F90free = ftn -Mfree
LINK = ftn

LAPACKLIB =
FFTWLIB = -L$(FFTW_DIR) -ldfftw

FOPTS = -fast
FNOOPTS = $(FOPTS)
...
Round 3

BerkeleyGW on Carver

% module load fftw mkl
% cat arch.mk
...
F90free = mpif90 -Mfree
LINK   = mpif90
FOPTS  = -fast

FFTWPATH = $(FFTW_ROOT)
FFTWLIB  = -L$(FFTW_LIBDIR) -ldfftw
FFTWINCLUDE = $(FFTW_INC)

LAPACKLIB = $(MKL)
SCALAPACKLIB = -L$(MKL_LIBDIR) -lmkl_scalapack_lp64 -lmkl_blacs_openmpi_lp64
...

% make (SUCCESS!)
Round 3

BerkeleyGW on Carver

% module load fftw mkl
% cat arch.mk
...
F90free = mpif90 -Mfree
LINK   = mpif90
FOPTS  = -fast

FFTWPATH     = $(FFTW_ROOT)
FFTWLIB      = -L$(FFTW_LIBDIR) -ldfftw
FFTWINCLUDE = $(FFTW_INC)

LAPACKLIB    = $(MKL)
SCALAPACKLIB = -L$(MKL_LIBDIR) -lmkl_scalapack_lp64 -lmkl_blacs_openmpi_lp64
...

% make (SUCCESS!)

(later that day...)

% mpirun -np 2 xi0.cplx.x
xi0.cplx.x: error while loading shared libraries: libmkl_scalapack_lp64.so: cannot open shared object file: No such file
xi0.cplx.x: error while loading shared libraries: libmkl_scalapack_lp64.so: cannot open shared object file: No such file
Round 3

Solution

On carver, we link against shared object files. These need to be present at runtime.

These need to be in your $LD_LIBRARY_PATH
Round 3

Solution

On carver, we link against shared object files. These need to be present at runtime.

These need to be in your $LD_LIBRARY_PATH

% module load mkl fftw

(or)

% export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/common/usg/mkl/10.2.2.025/lib/em64t:/usr/common/usg/fftw/2.1.5/lib
% module swap PrgEnv-pgi PrgEnv-gnu
% module load netcdf
% ./configure --prefix=""pwd""/../" CC=cc CXX=cc FC=ftn F77=ftn FCFLAGS="-O3" F77FLAGS="-O3" CFLAGS="-O3" CXXFLAGS="-O3" --with-netcdf-module-path="$CRAY_NETCDF_DIR/gnu/45/include" --enable-fortran

checking for module extension for compiler 'gcc'... mod
checking for ranlib... ranlib
checking for ar... ar
checking for /opt/cray/netcdf/4.1.3/gnu/45/include/netcdf.mod... yes
checking for netcdf library... no
Action: install NetCDF and set the library link path with --with-netcdf-ldflags.
configure: error: "No 'NetCDF' library found."

But... This worked for me one month ago just fine!?!?!?
To debug ./configure errors. Look at generated config.log.

% cat config.log
....
configure:3866: ftn -o conftest -O3 -I/opt/cray/netcdf/4.1.3/gnu/45/include conftest.f90 -Lnetcdf >&5
conftest.f90:3.12:

    use netcdf
    1
Fatal Error: Wrong module version '4' (expected '6') for file 'netcdf.mod' opened at (1)
configure:3872: $? = 1
configure: failed program was:
 |
 | program main
 | use netcdf
 | integer :: s, ncid
 | s = nf90_open(path = "", mode = NF90_NOWRITE, ncid = ncid)
 | end program main
....

ftn --version shows gcc 4.6, but netcdf module was compiled with gcc 4.5.
Round 2

User MPI Code

% cat jobscript.e3250732
Initial temperature 500.00000000000002 500.00000000000001
Application 5341714 exit signals: Floating point exception

do ia=rank+1,nsites,nprocs

% cat Makefile
...
FC = ftn #mpif90 -fpp # Modified
DEFINES = -DMPI

FILESF77= mdheff.F readinput.f readepq3ndonsite.f ran1.f timer.f
OBJS77= $(FILESF77:.f=.o)
.f.o:
   $(FC) -c $(FFLAGS) $(DEFINES) $<
....
Round 2

User MPI Code

% cat jobscript.e3250732
Initial temperature 500.0000000000002 500.0000000000001
Application 5341714 exit signals: Floating point exception

% cat Makefile
... FC = ftn #mpif90 -fpp # Modified
DEFINES = -DMPI
FILESF77= mdheff.F readinput.f readepqr3ndonsite.f ran1.f timer.f
OBJS77= $(FILESF77:.f=.o)
.f.o:
   $(FC) -c $(FFLAGS) $(DEFINES) $<
...
Solution

The $(DEFINES)$ was not included in the compilation of the .F files.

% cat Makefile
...
FC = ftn #mpif90 -fpp     # Modified
FLAGSOMP = # -O2 -march=pentium3 -mtune=core2 -assume byterecl -funroll-loops -fp-model precise -pc64 # Modified
DEFINES = -DMPI

FILESF77= mdheff.F readinput.f readepqr3ndonsite.f ran1.f timer.f
OBJSF77= $(FILESF77:.f=.o)
   .f.o:
       $(FC) -c $(FFLAGS) $(DEFINES) $<
   .F.o:
       $(FC) -c $(FFLAGS) $(DEFINES) $<
....