Building Applications at NERSC

Jack Deslippe and Zhengji Zhao

Presentation URL: http://goo.gl/zXcDz

BERKELEY LAB
Lawrence Berkeley National Laboratory

Outline

- Applications Already Available at NERSC
- Available Compilers
- Available Libraries

- Fix the problem Game

Presentation URL: http://goo.gl/zXcDz

Applications Already Available

Did you know that NERSC offers precompiled executables for more than 100 applications?

Applications Already Available

Did you know that NERSC offers precompiled executables for more than 100 applications?

Example, Materials Science:

VASP, NAMD, LAMMPS, NWCHEM, Quantum ESPRESSO, BerkeleyGW, SIESTA, Abinit, Gamess, GROMACS, GPAW MEEP, cpmd, libxc, etsf_io, atompaw, Wiek2K, Gaussian, PARATEC, cp2k, Wannier90, Amber, Yambo, XCrysden, Q-Chem

Applications Already Available

http://www.nersc.gov/users/software/all-software-list/

Show 25 entries Search:

#	Package	Platform	Category	Version	Module	Install Date	Date Made Default
0	ABINIT	carver	applications/ material sciences	6.0.3	abinit/6.0.3	2010-05-06	
1	ABINIT	carver	applications/ material sciences	6.10.3	abinit/6.10.3	2012-01-10	
2	ABINIT	carver	applications/ material sciences	6.2.2	abinit/6.2.2	2010-08-16	
3	ABINIT	carver	applications/ material sciences	6.4.1	abinit/6.4.1	2010-11-24	2012-01-12
4	ABINIT	franklin	applications/ material sciences	5.6	abinit/5.6	2008-12-03	2008-12-03
5	ABINIT	franklin	applications/ material sciences	5.8.4	abinit/5.8.4	2009-12-06	2009-12-15
6	ABINIT	franklin	applications/ material sciences	6.10.3	abinit/6.10.3	2012-01-10	
7	ABINIT	franklin	applications/ material sciences	6.4.3	abinit/6.4.3	2011-02-22	2011-03-24
8	ABINIT	hopper	applications/ material sciences	6.10.3	abinit/6.10.3	2012-01-09	

The NERSC Software Database on the web shows all of our available pre-compiled applications for Hopper, Franklin, Carver and Euclid.

Applications Already Available

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=gr
```

```
% make
```

```
% make install
```

Back to Top

Available Compilers

Compilers vs. Compiler Wrappers

pgf90, pgcc, pgCC vs ftn, cc, CC (mpif90, mpicc, mpiCC)

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

Available Compilers Across Machines:

	Hopper	Franklin	Carver	Euclid
PGI	✗	✗	✗	✗
GNU	✓	✓	✓	✓
Intel	✓		✓	
Pathscale	✓	✓		
Cray	✓	✓		

Available Compilers

Available Compilers Across Machines:

	Hopper	Franklin	Carver	Euclid
PGI	✗	✗	✗	✗
GNU	✗	✗	✗	✗
Intel	✗		✗	
Pathscale	✗	✗		
Cray	✗	✗		

Hopper/Franklin Module Access:

```
% module swap PrgEnv-pgi PrgEnv-gnu
```

Carver Module Access:

```
% module swap pgi gcc
% module swap openmpi openmpi-gcc
```

Available Compilers

When to use a particular compiler?

PGI - Default compiler. Easy to use. Good performance.

GNU - Use for best compatibility with open-source applications. Good performance on Fortran. Great C/C++ performance.

INTEL - Use for compatibility with applications + often better Fortran performance than GNU.

CRAY - Use to test performance.

PathScale - Use if you need another option to try.

Available Compilers

When to use a particular compiler?

PGI - Default compiler. Easy to use. Good performance.

GNU - Use for best compatibility with open-source applications. Good performance on Fortran. Great C/C++ performance.

INTEL - Use for compatibility with applications + often better Fortran performance than GNU.

CRAY - Use to test performance.

PathScale - Use if you need another option to try.

See Mike Stewart's presentation tomorrow for more information on optimizing your code.

Available Compilers

Useful Compiler Options:

	PGI	GNU	Intel	Cray
Optimization	-fast	-O3	-O3	-O3
OpenMP	-mp=nonuma	-fopenmp	-openmp	-Oomp
Version	-V	--version	-V	-V
Verbose	-v	-v	-v	-v
Debugging	-g -C -Wl,-ydgemm_	-g -fbounds-check	-g -warn all -CB	-g -R bs

- 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>"

Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  hopper/franklin: **ftn, cc, CC** not **gcc, gfortran, pgf90 ...**
  carver: **mpif90, mpicc, mpiCC**

Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  hopper/franklin: **ftn, cc, CC** not **gcc, gfortran, pgf90 ...**
  carver: **mpif90, mpicc, mpiCC**

- ftn links libraries (blas etc...) by default. mpif90 does not.

Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  hopper/franklin: **ftn, cc, CC** not **gcc, gfortran, pgf90 ...**
  carver: **mpif90, mpicc, mpiCC**

- ftn links libraries (blas etc...) by default. mpif90 does not.

- Hopper/Franklin statically link by default, but Carver dynamically links by default.

Available Compilers

Tips for using compilers:

- Use the compiler wrappers for parallel programs:
  hopper/franklin: **ftn, cc, CC** not **gcc, gfortran, pgf90 ...**
  carver: **mpif90, mpicc, mpiCC**

- ftn links libraries (blas etc...) by default. mpif90 does not.

- Hopper/Franklin statically link by default, but Carver dynamically links by default.

You can dynamically link on Hopper by adding -dynamic flag to compilation and "setenv CRAY_ROOTFS DSL" in batch script.

Available Compilers

Tips for using compilers:

To show libraries linked automatically with ftn

```
% ftn -v test.f90 -o test.x
...
--start-group -lscicpp_gnu -lsci_gnu_mp -lstdc++ -l gfortran -l m -lmpichf90_gnu -lmpich_gnu -
lmpichf90_gnu -lmpl -lsma -lxpmem -ldmapp -lugni -lpmi -Wl,--as-needed -lalpslli -lalpsutil -Wl,--no-as-
needed -ludreg -u pthread_mutex_destroy -u pthread_create -lpthread -Wl,--end-group -lgomp -lpthread -l
gfortran -l m
Using built-in specs.
...
```

To show which library "dgemm" is used from:

```
% ftn -Wl,-ydgemm_  test.f90 -o test.x
/scratch/scratchdirs/jdeslip/cc8ImteH.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. Carver:

```
% module avail fftw

--------------------------- /usr/common/usg/Modules/modulefiles ---------------------------
fftw/2.1.5(default) fftw/3.2.2          fftw-Intel/2.1.5    fftw-gnu/2.1.5
fftw/2.1.5-gnu      fftw/3.2.2-gnu      fftw-Intel/3.2.2    fftw-gnu/3.2.2
```

```
% module show fftw/2.1.5

setenv          FFTW_ROOT /usr/common/usg/fftw/2.1.5
setenv          FFTW_LIBDIR /usr/common/usg/fftw/2.1.5/lib
setenv          FFTW_INC /usr/common/usg/fftw/2.1.5/include
setenv          FFTW -I/usr/common/usg/fftw/2.1.5/include -L/usr/common/usg/fftw/2.1.5/lib
prepend-path    LD_LIBRARY_PATH /usr/common/usg/fftw/2.1.5/lib
prepend-path    INFOPATH /usr/common/usg/fftw/2.1.5/info
conflict        fftw/3.2.2
```

Fix the Problem Game

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.

Round 1

Quantum ESPRESSO on Hopper:

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

Round 1

Quantum ESPRESSO on Hopper:

```
% 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
     .....
```

Round 1

Quantum ESPRESSO on Hopper:

```
% 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....

Round 1

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

Round 1

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

```
% cat make.sys
.....
DFLAGS        =  -D__PGI -D__ACML
FDFLAGS       = $(DFLAGS)
MPIF90        = pgf90
#F90          = pgf90
CC            = pgcc
F77           = pgf77


.....
```

```
% 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 Hopper

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

LAPACKLIB =
FFTWLIB      =

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

Round 2

BerkeleyGW on Hopper

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

LAPACKLIB =
FFTWLIB      =

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.

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.

```
% 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
```

Round 4

etsf_io on Hopper

```
% 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!?!?!?

Round 4

Solution

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.

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

Presentation URL: http://goo.gl/zXcDz

More Information

NERSC Website:

- http://www.nersc.gov/nusers/systems/carver/programming/index.php
- http://www..nersc.gov/users/computational-systems/hopper/programming/

Man Pages:

% man pgf90

eMail Us:

consult@nersc.gov

Presentation URL: http://goo.gl/zXcDz

U.S. DEPARTMENT OF ENERGY
Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Round 2

User MPI Code

```
% cat jobscript.e3250732

Initial temperature 500.0000000000002 500.0000000000001
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 readepqr3ndonsite.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
```

nprocs=0

```
do ia=rank+1,nsites,nprocs
```

```
% 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) $<
....
```

Round 2

Solution

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

```
 % cat Makefile
...
FC = ftn #mpif90 -fpp          # Modified
FLAGSOPM =  #-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
OBJS77= $(FILESF77:.f=.o)
.f.o:
        $(FC) -c $(FFLAGS) $(DEFINES) $<
.F.o:
        $(FC) -c $(FFLAGS) $(DEFINES) $<
....
```