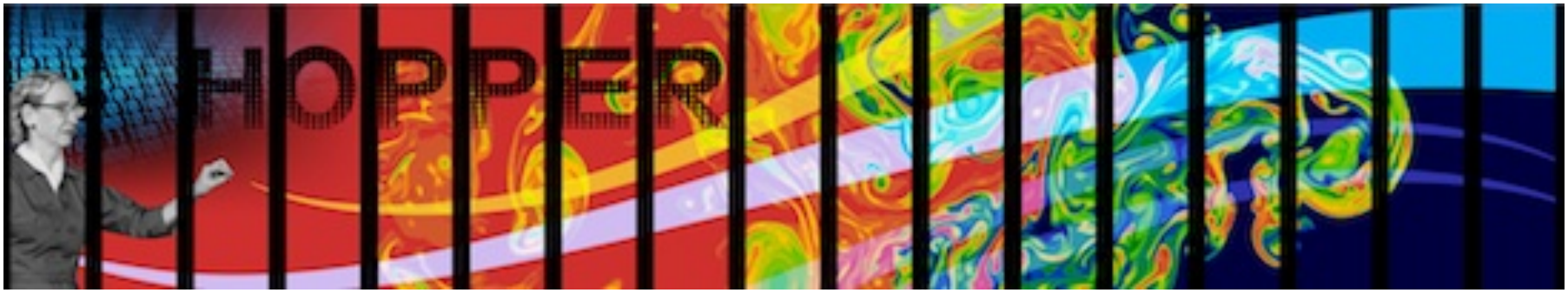
Compiling Code on the XE6



Rob Cunningham, LANL

Zhengji Zhao, NERSC

February 7, 2011

Compilation Topics

- Establish environment through modulefiles
- Compilers and their usage
- Shared objects
- A bit about Makefiles

Choose Environment: Modulefiles

- A software packaging mechanism to provide compatible sets of software tools and libraries
- Simple command-line interface: `module`
- Common sub-commands are: `avail`, `list`, `show`, `swap`, `unload`, `load`
- Default is PGI Compiler with Cray MPT and libsci
- On Hopper II, use `xtpe-mc12`

Modulefile Glimpse

- Default Modulefiles are Extensive, but can be tailored
- *Important*: use same modulefiles on both login and MOM node -- either in your dotfiles or within your script

```
hopper04 r/rtc> module list
Currently Loaded Modulefiles:
  1) modules/3.2.6.6                      10) gni-headers/2.1-1.0301.2792.5.1.gem
  2) xtpe-network-gemini                  11) xpmem/0.1-2.0301.24575.5.2.gem
  3) pgi/10.9.0                           12) xe-sysroot/3.1.49
  4) xt-libsci/10.5.0                     13) xt-asyncpe/4.7
  5) xt-mpt/5.1.4                         14) PrgEnv-pgi/3.1.49
  6) udreg/2.1-1.0301.2797.5.2.gem        15) eswrap/1.0.8
  7) ugni/2.1-1.0301.2798.5.2.gem         16) xtpe-mc12
  8) pmi/1.0-1.0000.8160.39.2.gem         17) torque/2.4.8-snap.201004261413
  9) dmapp/2.2-1.0301.2791.5.1.gem        18) moab/5.3.6-s14846
hopper04 r/rtc> module swap PrgEnv-pgi PrgEnv-cray
hopper04 r/rtc> module list >& mlist ; grep PrgEnv mlist
  3) PrgEnv-cray/3.1.49
hopper04 r/rtc>
```

Underneath Modulefiles

- No magic in Modulefiles – simple environment variables
- The software is already there, Modulefiles point to it

```
hopper04 r/rtc> module show cce
-------------------------------------------------------------------
/opt/modulefiles/cce/7.3.1:

setenv          GCC_X86_64 /opt/gcc/4.4.4/snos
          ...
prepend-path    NLSPATH /opt/cray/cce/7.3.1/CC/x86-64/nls/En/%N.cat:/opt/
cray/cce/7.3.1/craylibs/x86-64/nls/En/%N.cat:/opt/cray/cce/7.3.1/cftn/x86-64/
nls/En/%N.cat
prepend-path    INCLUDE_PATH_X86_64 /opt/cray/cce/7.3.1/craylibs/x86-64/
include
prepend-path    PATH /opt/cray/cce/7.3.1/cray-binutils/x86_64-unknown-linux-
gnu/bin:/opt/cray/cce/7.3.1/craylibs/x86-64/bin:/opt/cray/cce/7.3.1/cftn/
bin:/opt/cray/cce/7.3.1/CC/bin
append-path     MANPATH /usr/share/man
-------------------------------------------------------------------

hopper04 r/rtc>
```

Available Compilers

- Default is Portland Group: `PrgEnv-pgi`
- Cray also provides one: `PrgEnv-cray`
- gnu: `PrgEnv-gnu`
- Pathscale: `PrgEnv-pathscale`
- More information, recommendations in later session, and here:

  https://newweb.nersc.gov/for-users/computational-systems/hopper/programming/compiling-codes/

Common Command Line for Compilation

- In general, you will use only three compiler commands:

  `ftn`, `cc`, or `CC`

- Swapped compilers, same command
- See their manpages -- or
- More information from individual compiler manpages: `cray{ftn|cc|CC}`, `pg{f90|cc|CC}` , `gcc/gfortran`, etc.

Common Compiler Command Options

- Compile only: `-c` Output file: `-o <file>`
- Enable debug: `-g` use lib: `-L<path>`
- Include files: `-I<path>`
- Optimization: `-fast` (much more on this later)
- Many other options to tailor debugging, optimization, architecture

Dynamic Shared Objects and Libraries (DSL)

- Using system provided dynamic shared libraries
  1. module swap xt-mpt xt-mpich2
  2. Link codes with -dynamic
  3. Set runtime env, CRAY_ROOTFS=DSL

```
hopper01> pwd
/global/homes/z/zz217/XE6-feb-2011/compile
hopper01> module swap xt-mpt xt-mpich2
hopper01> ftn -dynamic mpi_test.f90
hopper01> qsub -I -V -l mppwidth=2 -q debug
qsub: waiting for job 141142.sdb to start
qsub: job 141142.sdb ready

nid05430> cd $PBS_O_WORKDIR
nid05430> export CRAY_ROOTFS=DSL
nid05430> aprun -n 2 a.out
  Hello World, I am process              0
  Hello World, I am process              1
Application 536003 resources: utime ~0s, stime ~0s
```

Dynamic Shared Objects and Libraries (DSL)

- Using user defined dynamic shared libraries
  1. module swap xt-mpt xt-mpich2
  2. Build shared libraries:
     a) Compile with –shared –fPIC
     b) Create dynamic shared libraries with cc –shared
  3. Set runtime env, CRAY_ROOTFS=DSL , LD_LIBRARY_PATH

Continued…

```
nid05430> ftn -shared -fPIC -c callC.f
nid05430> cc -shared -o libflib.so callC.o
nid05430> cc -dynamic callF.c -L./ -lflib
nid05430> export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:./
nid05430> aprun -n 2 a.out
 reached Fortran
        …
the Long int is 12345678901
Application 536015 exit codes: 28
Application 536015 resources: utime ~0s, stime ~0s
```

Makefie Usage

- `make` utility is designed to build executables from a set of rules and macros
- Can prevent excess typing, assist with complex builds
- Avoids redundancy when iterating through compile/run/modify loop
- Prevents mistakes from typos
- Allows simple clean-up