NATURAL ENVIRONMENT RESEARCH COUNCIL

EPSRC

COMPILING FOR THE ARCHER HARDWARE

Slides contributed by Cray and EPCC

CRAY THE SUPERCOMPUTER COMPANY

epcc

Modules

- The Cray Programming Environment uses the GNU “modules” framework to support multiple software versions and to create integrated software packages
  - As new versions of the supported software and associated man pages become available, they are installed and added to the Programming Environment as a new version, while earlier versions are retained to support legacy applications
  - System administrators will set the default version of an application, or you can choose another version by using modules system commands
  - Users can create their own modules, or administrators can install site specific modules available to many users.

Viewing the current module state

- Each login session has its own module state which can be modified by loading, swapping or unloading the available modules.
- This state affects the functioning of the compiler wrappers and in some cases runtime of applications.
- A standard, default set of modules is always loaded at login for all users.
- Current state can be viewed by running:

```
$> module list
```

Default modules example

```
adrianj@eslogin001:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.6.7
  2) nodestat/2.2-1.0500.41375.1.85.ari
  3) sdb/1.0-1.0500.43793.6.11.ari
  4) alps/5.0.3-2.0500.8095.1.1.ari
  5) MySQL/5.0.64-1.0000.7096.23.1
  6) lustre-cray_ari_s/2.3_3.0.58_0.6.6.1_1.0500.7272.12.1-1.0500.44935.7.1
  7) udreg/2.3.2-1.0500.6756.2.10.ari
  8) ugni/5.0-1.0500.0.3.306.ari
  9) gni-headers/3.0-1.0500.7161.11.4.ari
 10) dmapp/6.0.1-1.0500.7263.9.31.ari
 11) xpmem/0.1-2.0500.41356.1.11.ari
 12) hss-llm/7.0.0
 13) Base-opts/1.0.2-1.0500.41324.1.5.ari
 14) craype-network-aries
 15) craype/1.06.05
 16) cce/8.2.0.181
...
```

Viewing available modules

- There may be many hundreds of possible modules available to users.
  - Beyond the pre-loaded defaults there are many additional packages provided by Cray
  - Sites may choose to install their own versions.
- Users can see all the modules that can be loaded using the command:
  - `module avail`
- Searches can be narrowed by passing the first few characters of the desired module, e.g.

```
adrianj@eslogin001 :~> module avail gc

------------------------------ /opt/modulefiles ---------------------------
gcc/4.6.1          gcc/4.7.2          gcc/4.8.0
gcc/4.6.3          gcc/4.7.3          gcc/4.8.1(default)
```

Modifying the default environment

- Loading, swapping or unloading modules:
  - The default version of any inidividual modules can be loaded by name
    - e.g.: `module load perftools`
  - A specific version can be specified after the forward slash.
    - e.g.: `module load perftools/6.1.0`
  - Modules can be swapped out in place
    - e.g.: `module swap intel intel/13.1.1.163`
  - Or removed entirely
    - e.g.: `module unload perftools`
- Modules will automatically change values of variables like `PATH`, `MANPATH`, `LM_LICENSE_FILE`... etc
  - Modules also provide a simple mechanism for updating certain environment variables, such as `PATH`, `MANPATH`, and `LD_LIBRARY_PATH`
  - In general, you should make use of the modules system rather than embedding specific directory paths into your startup files, makefiles, and scripts

```
adrianj@eslogin008:~> module show fftw
-------------------------------------------------------------------
/opt/cray/modulefiles/fftw/3.3.0.4:

setenv           FFTW_VERSION 3.3.0.4
setenv           CRAY_FFTW_VERSION 3.3.0.4
setenv           FFTW_DIR /opt/fftw/3.3.0.4/sandybridge/lib
setenv           FFTW_INC /opt/fftw/3.3.0.4/sandybridge/include
prepend-path     PATH /opt/fftw/3.3.0.4/sandybridge/bin
prepend-path     MANPATH /opt/fftw/3.3.0.4/share/man
prepend-path     CRAY_LD_LIBRARY_PATH /opt/fftw/3.3.0.4/sandybridge/lib
setenv           PE_FFTW_REQUIRED_PRODUCTS PE_MPICH
prepend-path     PE_PKGCONFIG_PRODUCTS PE_FFTW
setenv           PE_FFTW_TARGET_interlagos interlagos
setenv           PE_FFTW_TARGET_sandybridge sandybridge
setenv           PE_FFTW_TARGET_x86_64 x86_64
setenv           PE_FFTW_VOLATILE_PKGCONFIG_PATH /opt/fftw/3.3.0.4/
@PE_FFTW_TARGET@/lib/pkgconfig
prepend-path     PE_PKGCONFIG_LIBS
fftw3f_mpi:fftw3f_threads:fftw3f:fftw3_mpi:fftw3_threads:fftw3
module-whatis    FFTW 3.3.0.4 - Fastest Fourier Transform in the West
-------------------------------------------------------------------
```

Summary of Useful module commands

- Which modules are available?
  - `module avail`, `module avail cce`
- Which modules are currently loaded?
  - `module list`
- Load software
  - `module load perftools`
- Change programming environment
  - `module swap PrgEnv-cray PrgEnv-gnu`
- Change software version
  - `module swap cce/8.0.2  cce/7.4.4`
- Unload module
  - `module unload cce`
- Display module release notes
  - `module help cce`
- Show summary of module environment changes
  - `module show cce`

NATURAL ENVIRONMENT RESEARCH COUNCIL

EPSRC

COMPILING APPLICATIONS FOR THE CRAY XC

CRAY
THE SUPERCOMPUTER COMPANY

epcc

THE UNIVERSITY OF EDINBURGH

Compiler Driver Wrappers (1)

- All applications that will run in parallel on the Cray XC should be compiled with the standard language wrappers.

  The compiler drivers for each language are:
  - `cc` – wrapper around the C compiler
  - `CC` – wrapper around the C++ compiler
  - `ftn` – wrapper around the Fortran compiler

- These scripts will choose the required compiler version, target architecture options, scientific libraries and their include files automatically from the module environment.

- Use them exactly like you would the original compiler, e.g. To compile `prog1.f90` run

```
ftn -c prog1.f90
```

Compiler Driver Wrappers (2)

- The scripts choose which compiler to use from the `PrgEnv` module loaded

PrgEnv	Description	Real Compilers
`PrgEnv-cray`	Cray Compilation Environment	`crayftn, craycc, crayCC`
`PrgEnv-intel`	Intel Composer Suite	`ifort, icc, icpc`
`PrgEnv-gnu`	GNU Compiler Collection	`gfortran, gcc, g++`

- Use module swap to change `PrgEnv`, e.g.
  - `module swap PrgEnv-cray PrgEnv-intel`
- `PrgEnv-cray` is loaded by default at login. This may differ on other Cray systems.
  - use `module list` to check what is currently loaded
- The Cray MPI module is loaded by default (`cray-mpich`).
  - To support SHMEM load the `cray-shmem` module.
- Check that the craype-ivybridge module is loaded
- The drivers automatically support an MPI build
  - No need to use specific wrappers such as mpiifort, mpicc

PLEASE NOTE : Cross Compiling Environment

- You are compiling on a Linux login node
  but generating an executable for a CLE compute node

- Do not use crayftn, craycc, ifort, icc, gcc, g++…
  unless you want a Linux executable for the service node
  - ALWAYS Use `ftn`, `cc`, or `CC` instead
  - Use the direct compiler commands if the executable is supposed to run on the service nodes (utilities, setup, …)

Compiler Versions

- There are usually multiple versions of each compiler available to users.
  - The most recent version is usually the default and will be loaded when swapping `PrgEnvs`.
  - To change the version of the compiler in use, swap the Compiler Module. e.g. `module swap cce cce/8.1.6`

PrgEnv	Compiler Module
`PrgEnv-cray`	`cce`
`PrgEnv-intel`	`Intel`
`PrgEnv-gnu`	`gcc`
~~`PrgEnv-pgi`~~	~~`pgi`~~

About the `-I`, `-L` and `-l` flags

- For libraries and include files covered by module files, you should NOT add anything to your Makefile
  - No additional MPI flags are needed (included by wrappers)
  - You do not need to add any `-I`, `-l` or `-L` flags for the Cray provided libraries

- If your Makefile needs an input for `-L` to work correctly, try using '.'

- If you really, really need a specific path, try checking 'module show X' for some environment variables

```
adrianj@eslogin008:~> module show fftw
-------------------------------------------------------------------
/opt/cray/modulefiles/fftw/3.3.0.4:

setenv           FFTW_VERSION 3.3.0.4
setenv           CRAY_FFTW_VERSION 3.3.0.4
setenv           FFTW_DIR /opt/fftw/3.3.0.4/sandybridge/lib
setenv           FFTW_INC /opt/fftw/3.3.0.4/sandybridge/include
prepend-path     PATH /opt/fftw/3.3.0.4/sandybridge/bin
prepend-path     MANPATH /opt/fftw/3.3.0.4/share/man
prepend-path     CRAY_LD_LIBRARY_PATH /opt/fftw/3.3.0.4/sandybridge/lib
setenv           PE_FFTW_REQUIRED_PRODUCTS PE_MPICH
prepend-path     PE_PKGCONFIG_PRODUCTS PE_FFTW
setenv           PE_FFTW_TARGET_interlagos interlagos
setenv           PE_FFTW_TARGET_sandybridge sandybridge
setenv           PE_FFTW_TARGET_x86_64 x86_64
setenv           PE_FFTW_VOLATILE_PKGCONFIG_PATH /opt/fftw/3.3.0.4/
@PE_FFTW_TARGET@/lib/pkgconfig
prepend-path     PE_PKGCONFIG_LIBS
fftw3f_mpi:fftw3f_threads:fftw3f:fftw3_mpi:fftw3_threads:fftw3
module-whatis    FFTW 3.3.0.4 - Fastest Fourier Transform in the West
-------------------------------------------------------------------
```

OpenMP

- OpenMP is support by all of the `PrgEnvs`.
  - CCE (`PrgEnv-cray`) recognizes and interprets OpenMP directives by default. If you have OpenMP directives in your application but do not wish to use them, disable OpenMP recognition with `–hnoomp`.

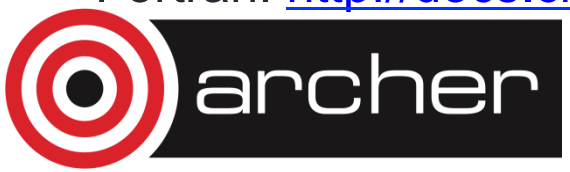
PrgEnv	Enable OpenMP	Disable OpenMP
PrgEnv-cray	-homp	-hnoomp
PrgEnv-intel	-openmp	
PrgEnv-gnu	-fopenmp	

Compiler man pages and documentation

- For more information on individual compilers

PrgEnv	C	C++	Fortran
PrgEnv-cray	man craycc	man crayCC	man crayftn
PrgEnv-intel	man icc	man icpc	man ifort
PrgEnv-gnu	man gcc	man g++	man gfortran
Wrappers	man cc	man CC	man ftn

- To verify that you are using the correct version of a compiler, use:
  - **-V** option on a cc, CC, or ftn command with CCE and Intel
  - **--version** option on a cc, CC, or ftn command with GNU
- Cray Reference Manuals:
  - C and C++: http://docs.cray.com/books/S-2179-81/
  - Fortran: http://docs.cray.com/books/S-3901-81/



Dynamic compilation

- Default behaviour is to perform static linking
- Dynamic linking possible:
  - Use the -dynamic flag when invoking the compiler for linking.
  - Set the environment variable CRAYPE_LINK_TYPE=dynamic without any extra compilation/linking options.
- Will need to have libraries available on /work filesystem

General Cray Compiler Flags

- Optimisation Options
  - **-O2** — optimal flags [enabled by default]
  - **-O3** — aggressive optimization
  - **-O ipaN (ftn)** or **-hipaN (cc/CC)** — inlining, N=0-5 [default N=3]
- Create listing files with optimization info
  - **-ra (ftn)** or **-hlist=a (cc/CC)** — creates a listing file with all optimization info
  - **-rm (ftn)** or **-hlist=m (cc/CC)** — produces a source listing with loopmark information
- Parallelization Options
  - **-O omp (ftn)** or **-h omp (cc/CC)** — Recognize OpenMP directives [default]
  - **-O threadN (ftn)** or **h threadN (cc/CC)** — control the compilation and optimization of OpenMP directives, N=0-3 [default N=2]

➔ More info: man crayftn, man craycc, man crayCC

Recommended CCE Compilation Options

- Use default optimization levels
  - It's the equivalent of most other compilers -O3 or -fast
  - It is also our most thoroughly tested configuration
- Use -O3,fp3 (or -O3 -hfp3, or some variation) if the application runs cleanly with these options
  - **-O3** only gives you slightly more than the default **-O2**
  - Cray also test this thoroughly
  - **-hfp3** gives you a lot more floating point optimization (default is **-hfp2**)
- If an application is intolerant of floating point reordering, try a lower -hfp number
  - Try **-hfp1** first, only **-hfp0** if absolutely necessary (**-hfp4** is the maximum)
  - Might be needed for tests that require strict IEEE conformance
  - Or applications that have 'validated' results from a different compiler
- Do not use too aggressive optimizations , e.g. -hfp4
  - Higher numbers are not always correlated with better performance

OpenMP

- OpenMP is ON by default
  - This is the opposite default behavior that you get from GNU and Intel compilers
  - Optimizations controlled by **-OthreadN (ftn)** or **-hthreadN (cc/CC)**, N=0-3 [default N=2]
  - To shut off use **-O/-h thread0** or **-xomp (ftn)** or **-hnoomp**

- Autothreading is NOT on by default
  - **-hautothread** to turn on
  - Interacts with OpenMP directives

- If you do not want to use OpenMP and have OMP directives in the code, make sure to shut off OpenMP at compile time

CCE – GNU – Intel compilers

- More or less all optimizations and features provided by CCE are available in Intel and GNU compilers
  - GNU compiler serves a wide range of users & needs
    - Default compiler with Linux, some people only test with GNU
    - **GNU defaults are conservative** (e.g. -O1)
      - -O3 includes vectorization and most inlining
    - Performance users set additional options
  - Intel compiler is typically more aggressive in the optimizations
    - **Intel defaults are more aggressive** (e.g -O2), to give better performance "out-of-the-box"
      - Includes vectorization; some loop transformations such as unrolling; inlining within source file
    - Options to scale back optimizations for better floating-point reproducibility, easier debugging, etc.
    - Additional options for optimizations less sure to benefit all applications
  - **CCE is even more aggressive** in the optimizations by default
    - Better inlining and vectorization
    - Aggressive floating-point optimizations
    - OpenMP enabled by default
- GNU users probably have to specify higher optimisation levels

Cray, Intel and GNU compiler flags

Feature	Cray	Intel	GNU
Listing	-ra (fnt) -hlist=a (cc/CC)	-opt-report3	-fdump-tree-all
Free format (ftn)	-f free	-free	-ffree-form
Vectorization	By default at -O1 and above	By default at -O2 and above	By default at -O3 or using -ftree-vectorize
Inter-Procedural Optimization	-hwp	-ipo	-flto (note: link-time optimization)
Floating-point optimizations	-hfpN, N=0...4	-fp-model [fast|fast=2| precise| except|strict]	-f[no-]fast-math or -funsafe-math-optimizations
Suggested Optimization	(default)	-O2 -xAVX	-O2 -mavx -ftree-vectorize -ffast-math -funroll-loops
Aggressive Optimization	-O3 -hfp3	-fast	-Ofast -mavx -funroll-loops
OpenMP recognition	(default)	-fopenmp	-fopenmp
Variables size (ftn)	-s real64 -s integer64	-real-size 64 -integer-size 64	-freal-4-real-8 -finteger-4-integer-8

NATURAL ENVIRONMENT RESEARCH COUNCIL

EPSRC

ARCHER PBS BATCH SYSTEM

CRAY
THE SUPERCOMPUTER COMPANY

epcc

THE UNIVERSITY OF EDINBURGH

Requesting resources from PBS

Jobs provide a list of requirements as #PBS comments in the headers of the submission script, e.g.

```
#PBS -l walltime=12:00:00
```

These can be overriden or supplemented as submission by adding to the qsub command line, e.g.

```
> qsub -l walltime=11:59:59 run.pbs
```

Common options include:

Option	Description
`-N <name>`	A name for job,
`-q <queue>`	Submit job to a specific queues.
`-o <output file>`	A file to write the job's stdout stream in to.
`--error <error file>`	A file to write the job's stderr stream in to.
`-j oe`	Join stderr stream in to stdout stream as a single file
`-l walltime=<HH:MM:SS>`	Maximum wall time job will occupy
`-A <code>`	Account to run job under (for controlling budgets)

Requesting parallel resources

Jobs must also request “chunks” of nodes:

This is done using the select option, e.g.

```
-l select=<numnodes>
```

Option	Description
`select=<numnodes>`	Requests `<numnodes>` nodes from the system.
`select=bigmem=true`	High memory nodes

```
qsub -l select=<numnodes> ./myjob.pbs
qsub -l select=<numnodes>:bigmem=true ./mybigjob.pbs
```

Launching Parallel applications

- ALPS : Application Level Placement Scheduler
- aprun is the ALPS application launcher
  - aprun launches sets of PEs on the compute nodes.
  - aprun man page contains several useful examples
  - The most important parameters to set is -n:

Description	Option
Total Number of PEs used by the application	-n
Number of PEs per compute node	-N

```
aprun –n 24 ./mympiprog.exe          # default –N 24
aprun –n 24 –N 12 ./mympiprog.exe    # uses 2 nodes
```

Example batch script

```
#!/bin/bash --login

# PBS job options (name, compute nodes, job time)
#PBS -N Example_MPI_Job
#PBS -l select=64
#PBS -l walltime=00:20:00

# Replace [project code] below with your project code (e.g. t01)
#PBS -A [project code]

# Make sure any symbolic links are resolved to absolute path
export PBS_O_WORKDIR=$(readlink -f $PBS_O_WORKDIR)

# Change to the directory that the job was submitted from
# (remember this should be on the /work filesystem)
cd $PBS_O_WORKDIR

# Launch the parallel job
#   Using 1536 MPI processes and 24 MPI processes per node
aprun -n 1536 ./my_mpi_executable.x arg1 arg2
```

A Note on the mppwidth, mppnppn and mppdepth

Casual examination of older Cray documentation or internet searches may suggest the alternatives, mppwidth, mppnppn and mppdepth for requesting resources.

These methods, while still functional, are Cray specific extensions to PBS Pro which have been deprecated by Altair.

Therefore we recommend all new scripts be structure using “select” notation describe previously

User Tools

- module load epcc-tools
- budgets command

```
adrianj@eslogin003:~> budgets
========================================
              Budget       Remaining kAUs
----------------------------------------
                 z01               559.812
========================================
```

- checkScript
  - Test your pbs script to see if it's correct
- serialJobs
  - Display the jobs running on the post-processing/serial nodes
- bolt
  - Create job submission script for you

NATURAL ENVIRONMENT RESEARCH COUNCIL

EPSRC

QUESTIONS?

CRAY
THE SUPERCOMPUTER COMPANY

epcc

THE UNIVERSITY OF EDINBURGH