Preparing applications for the Cray XE

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++
PrgEnv-pgi	Portland Group Compilers	pgf90, pgcc, pgCC

- 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-mpich2).
  - To support SHMEM load the `cray-shmem` module.

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

EXCEPTION: Cross Compiling Environment

- **The wrapper scripts, `ftn`, `cc` and `CC`, will create a highly optimised executable tuned for the Cray XE's compute nodes.**

- **This executable may not run on the login nodes**
  1. Login nodes do not support running distributed memory applications
  2. Some Cray architectures may have different processors in the login and compute nodes. E.g. cross-compilation.

- **If you are compiling for the login node you should use the original direct compiler commands**
  - e.g. ifort, ipcp, crayftn, gcc, g++ or gfortran.
  - The PATH variable will change with the modules.

About the –I, –L and –l flags

- **For libraries and include files being triggered 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**

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	
PrgEnv-pgi	-mp	

Compiler man Pages

- **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
PrgEnv-pgi	man pgcc	man pgCC	man pgf90
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 PGI, Intel and Cray
  - --version option on a cc, CC, or ftn command with GNU

Running applications on the Cray XE

How applications run on a Cray XE

- **Most Cray XEs are batch systems.**
  - Users submit batch job scripts to a scheduler from a login node (e.g. PBS, MOAB, SLURM) for execution at some point in the future. Each job requires resources and a predicts how long it will run.
  - The scheduler (running on an external server) chooses which jobs to run and allocates appropriate resources
  - The batch system will then execute the user's job script on an a different login or batch "MOM" node.
  - The scheduler monitors the job and kills any that overrun their runtime prediction.
- **User job scripts typically contain two types of statements.**
  1. Serial commands that are executed by the MOM node, e.g.
     - quick setup and post processing commands
     - e.g. (`rm`, `cd`, `mkdir` etc)
  2. Parallel executables that run on compute nodes.
     1. Launched using the `aprun` command.

The Two types of Cray XE Nodes

Login or service nodes

- This is the node you access when you first log in to the system.
- It runs a full version of the CLE operating system (all libraries and tools available)
- They are used for editing files, compiling code, submitting jobs to the batch queue and other interactive tasks.
- They are shared resources that may be used concurrently by multiple users.
- There may be many login nodes in any Cray XE6 and can be used for various system services (IO routers, daemon servers).
- They can be either connected to the Cray Gemini network (internal login nodes) or proxies (external or esLogin nodes).

Compute nodes

- These are the nodes on which production jobs are executed
- It runs Compute Node Linux, a version of the OS optimised for running batch workloads
- They can only be accessed by submitting jobs through a batch management system (e.g. PBS Pro, Moab, SLURM)
- They are exclusive resources that may only be used by a single user.
- There are many more compute nodes in any Cray XE6 than login or service nodes.
- They are always directly connected to the Cray Aries.

Primary File Systems on HECToR

- **Home space (`$HOME`, NFS)**
  - Not visible to compute nodes
  - Has quotas and is backed up
- **Work Space (1200TB, Lustre, `/esfs{1,2}`)**
  - Parallel filesystem optimized for large files, high bandwidth
  - Use for scientific output, restart files etc.
  - Visible to login nodes and compute nodes
  - Not backed up
- **There is no local storage on compute nodes**

Lifecycle of a batch script

esLogin

`sbatch run.sh`

Example Batch Job Script – `run.sh`

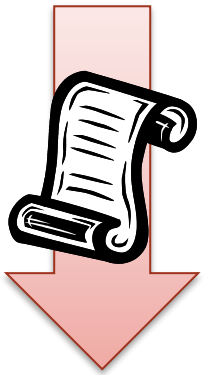
Scheduler Resources

Serial

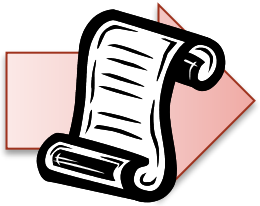
Parallel

```
#!/bin/bash
#PBS -l mppwidth=64
#PBS -l mppnppn=4

cd $WORKDIR
aprun -n 64 -N 4 simulation.exe
rm -r $WORKDIR/tmp
```

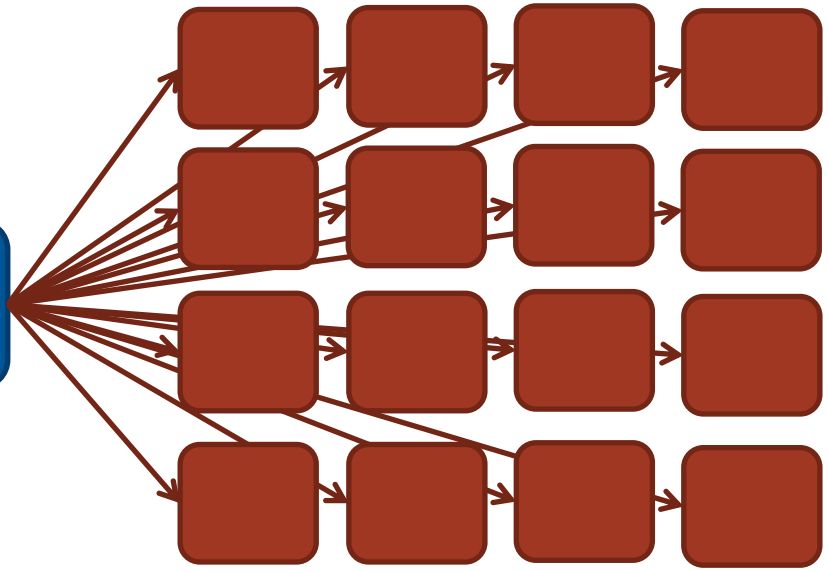


PBS Queue Manager



PBS MOM Node

Cray XE Compute Nodes



Running an application on the Cray XE ALPS + aprun

- **ALPS : Application Level Placement Scheduler**
- **aprun is the ALPS application launcher**
  - It **must** be used to run application on the XE compute nodes: interactively or in a batch job
  - If `aprun` is not used, the application is launched on the MOM node (and will most likely fail).
  - `aprun` launches groups of Processing Elements (PEs) on the compute nodes
    (PE == MPI RANK == Coarray Image == UPC Thread)
  - `aprun` man page contains several useful examples
  - The 3 most important parameters to set are:

Description	Option
Total Number of PEs used by the application	-n
Number of PEs per compute node	-N
Number of threads per PE (More precise, the “stride” between 2 PEs on a node)	-d

Running applications on the Cray XE6: Some basic examples

- Assuming an XC30 with Sandybridge nodes (32 cores per node with Hyperthreading)
- Pure MPI application, using all the available cores in a node

```
$ aprun -n 32 -N 32 ./a.out
```

- Pure MPI application, using only 1 rank per node
  - 32 MPI tasks, 32 nodes with 32*32 core allocated
  - Can be done to increase the available memory for the MPI tasks

```
$ aprun -N 1 -n 32 -d 32 ./a.out
```

- Hybrid MPI/OpenMP application, 4 MPI ranks (PE) per node
  - 32 MPI tasks, 8 OpenMP threads each
  - need to set `OMP_NUM_THREADS`

```
$ export OMP_NUM_THREADS=8
$ aprun -n 32 -N 4 -d $OMP_NUM_THREADS ./a.out
```


15

Scheduling a batch application with PBS

- The number of required nodes can be specified in the job header
- The job is submitted by the **`qsub`** command
- At the end of the exection, output and error files are returned to submission directory
- You can check the status of jobs with:
  `qstat ?<jobid>?`
- You can delete a job with
  qdel <jobid>

Hybrid MPI + OpenMP

```
#!/bin/bash
#PBS -N hybrid
#PBS -l mppwidth=64
#PBS -l mppnppn=8
#PBS -l mppdepth=4
#PBS -l walltime=00:20:00

export OMP_NUM_THREADS=4
aprun -n64 -d4 -N8 a.out
```

Other PBS options

- `#PBS -l mppwidth=1024`
  Number of PEs to use in the job
- `#PBS -l mppnppn=8`
  Number of PEs to use per node
- `#PBS -l mppdepth=4`
  Number of threads per PE
- `#PBS -o std.out`
  `#PBS --error=std.err`
  File names of stdout and stderr
- `#PBS -j oe`
  Join stdout and stderr into a single output stream (stdout name)
- `#PBS -A d26`
  HECToR's account code for the project (d26 is training).

PBS and aprun

PBS	aprun	Descropt
`-l mppwidth=$PE`	`-n $PE`	Number of PE to start
`-l mppdepth=$threads`	`-d $threads`	# threads/PE
`-l mppnppn=$N`	`-N $N`	# (PEs per node)

- Shortcut: `aprun`'s `-B` option will automatically use the appropriate PBS settings for `-n`,`-N`,`-d` and `-m`, e.g.

```
aprun -B ./a.out
```

Watching a launched job on the Cray XE

- **`xtnodestat`**
  - Shows how XE nodes are allocated and corresponding `aprun` commands
- **`apstat`**
  - Shows `aprun` processes status
  - `apstat` overview
  - `apstat -a[ apid ]` info about all the applications or a specific one
  - `apstat -n` info about the status of the nodes
- **Batch `qstat` command**
  - shows batch jobs