Running applications on the Cray XC30

Running on compute nodes

By default, users do not log in and run applications on the compute nodes directly.

Instead they launch jobs on compute nodes in one of three available modes:

1. **Extreme Scaling Mode (ESM Mode)**
   The default high performance MPP mode
2. **Pre/Post Processing Mode**
   A throughput mode designed to allows efficient Multiple Applications Multiple User (MAMU) access for smaller applications.
3. **Cluster Compatibility Mode (CCM)**
   Emulation mode designed primarily to support 3rd ISV applications which cannot be recompiled.

Extreme Scaling Mode (ESM)

ESM is a high performance mode designed to run larger applications at scale. Important features are:

- **Dedicated compute nodes for each user job**
  - No interference from other users sharing the node
  - Minimum quanta of compute is 1 node.
  - Nodes running with low-noise, low-overhead CNL OS
- **Inter-node communication is via native Aries API.**
  - The best latency and bandwidth comms are available using ESM.
  - Applications need to be linked with a Cray comms libraries (MPI, Shmem etc) or compiled with Cray language support (UPC, Coarrays)
- **The appropriate parallel runtime environment is automatically set up between nodes**

ESM is expected to be the default mode for the majority of applications that require at least one node to run.

Requesting resources from a batch system

- **As resources are dedicated to users in ESM mode, most supercomputers share nodes via batch systems.**
- **ECMWF Cray XC30s use PBS Pro to schedule resources**
  - Users submit batch job scripts to a scheduler from a login node (e.g. PBS) for execution at some point in the future.
    Each job requests resources and predicts how long it will run.
  - The scheduler (running on an external server) then chooses which jobs to run and when, allocating appropriate resources at the start.
  - The batch system will then execute a copy of the user's job script on an a one of the "MOM" nodes.
  - The scheduler monitors the job throughout it lifetime. Reclaiming the resources when job scripts finish or are killed for overrunning.
- **Each user job scripts will contain two types of command**
  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 launches to run on the allocated compute nodes.
     1. Launched using the `aprun` command.

Example ECMWF batch script

```
#!/bin/ksh
#PBS -N xthi
#PBS -l EC_total_tasks=256
#PBS -l EC_threads_per_task=6
#PBS -j oe
#PBS -o ./output
#PBS -l walltime=00:05:00

export OMP_NUM_THREADS=$EC_threads_per_task

aprun -n $EC_total_tasks \
      -N $EC_tasks_per_node \
      -d $EC_threads_per_task ./a.out
```

Request resources from the batch system

Launch the executable on the compute nodes in parallel

Lifecycle of an ECMWF batch script

esLogin

qsub run.sh

Example Batch Job Script – run.sh

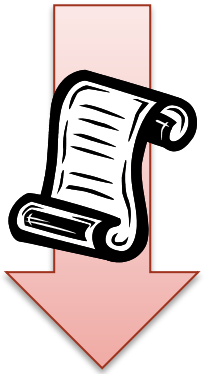
```
#!/bin/bash
#PBS -l EC_total_tasks=256,EC_threads_per_task=6
#PBS -l walltime=1:00:00

cd $WORKDIR
aprun -n 256 -d 6 simulation.exe
rm -r $WORKDIR/tmp
```

Requested Resources

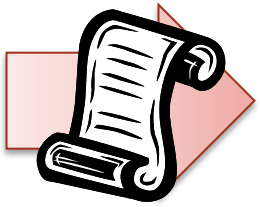
Serial

Parallel

Serial

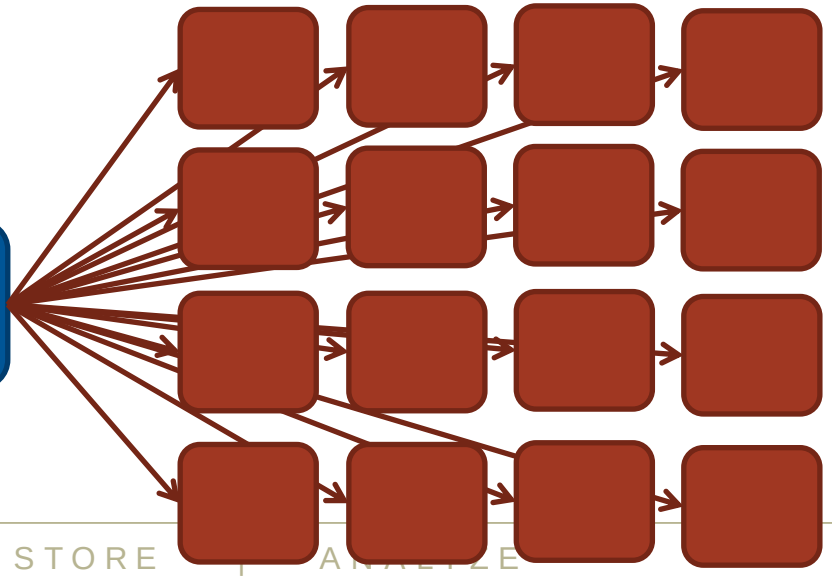


PBS Queue Manager



PBS MOM Node

Cray XE Compute Nodes



Submitting jobs to the batch system

- Job scripts are submitted to the batch system with qsub:
  - `qsub script.pbs`
- Once a job is submitted is assigned a PBS Job ID, e.g.
  - `1842232.sdb`
- To view the state of all the currently queued and running jobs run:
  - `qstat`
- To limit to just jobs owned by a specific user
  - `qstat –u <username>`
- To remove a job from the queue, or cancel a running job
  - `qdel <job id>`

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 suplemented at submission by adding to the qsub command line, e.g.

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

Common options are:

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.
`-e <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

Glossary of terms

To understand how many compute nodes a job needs, we need to understand how parallel jobs are described by Cray.

PE/Processing Element

- A discrete software process with an individual address space. One PE is equivalent to:

1 MPI Rank, 1 Coarray Image, 1 UPC Thread, or 1 SHMEM PE

Threads

- A logically separate stream of execution inside a parent PE that shares the same address space

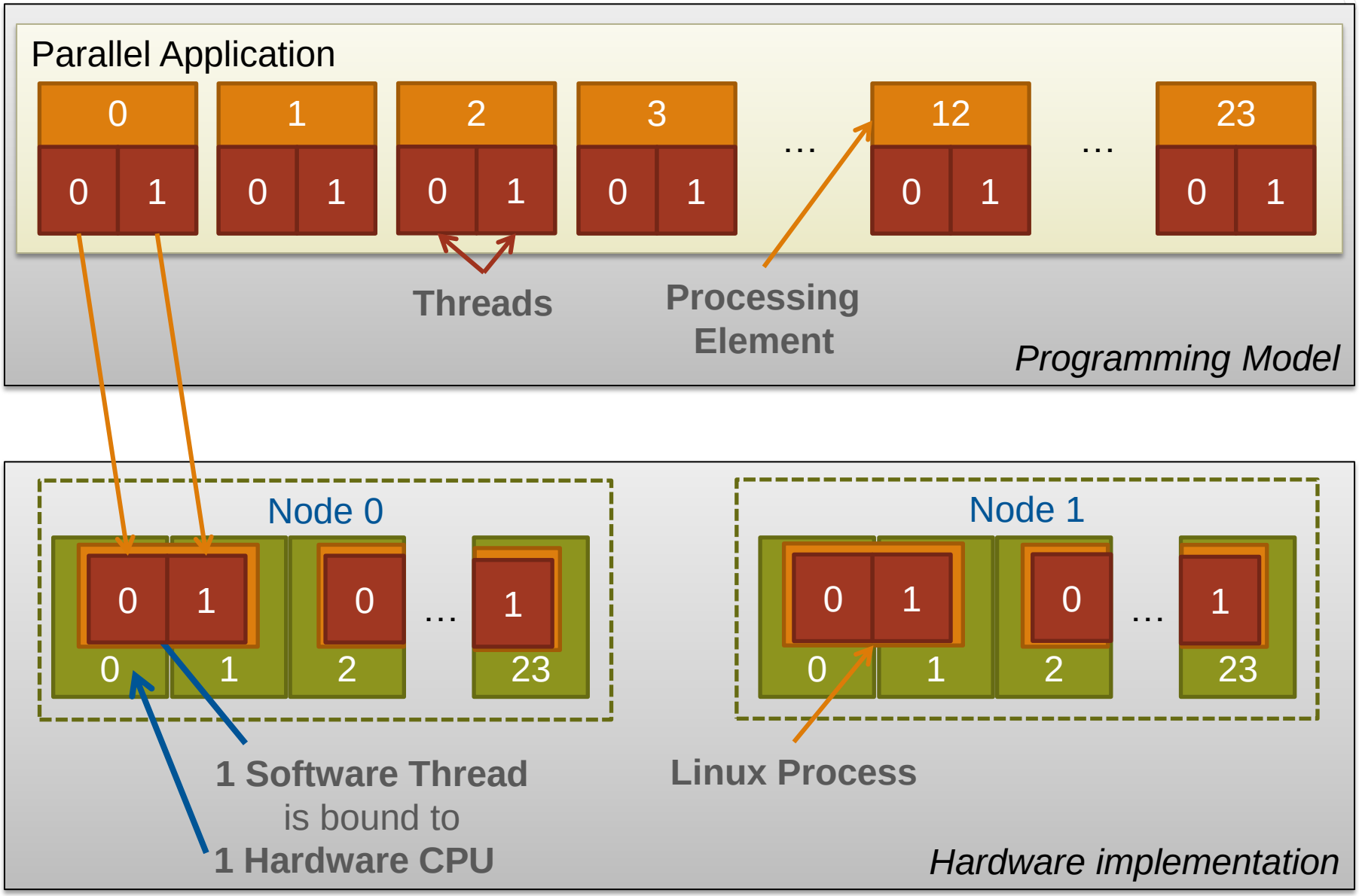
CPU

- The minimum piece of hardware capable of running a PE. It may share some or all of its hardware resources with other CPUs
  Equivalent to a single “Intel Hyperthread”

Compute Unit

- The individual unit of hardware for processing, may be seen described as a “core”. May provide one or more CPUs.

Implementing the Parallel Programming Model on hardware



Launching ESM Parallel applications

- **ALPS : Application Level Placement Scheduler**
- **aprun is the ALPS application launcher**
  - It **must** be used to run application on the XC compute nodes in ESM mode, (either interactively or as a batch job)
  - If aprun is not used, the application will be run on the MOM node (and will most likely fail).
  - aprun launches sets of PEs on the compute nodes.
  - aprun man page contains several useful examples
  - The 4 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
Number of to CPUs to use per Compute Unit	-j

Running applications on the Cray XC30: Some basic examples

Assuming XC30 nodes with 2x12 core Ivybridge processors

- Each node has: 48 CPUs/Hyperthreads and 24 Compute Units/cores

- **Launching an MPI application on all CPUs of 64 nodes:**
  - Using 1 CPU per Compute Unit means a maximum of 24 PEs per node.
  - 64 nodes x 24 ranks/node = 1536 ranks

```
$ aprun -n 1536 -N 24 -j1 ./model-mpi.exe
```

- **Launch the same MPI application on 64 nodes but with half as many ranks per node**
  - Still using 1 CPU per Compute Unit, but limiting to 12 Compute Units.

```
$ aprun -n 768 -N 12 -j1 ./model-mpi.exe
```

  - Doubles the available memory for each PE on the node

- **To use all availble CPUs on 64 nodes.**
  - Using 2 CPUs per Compute unit, so 48 PEs per node)

```
$ aprun -n 3072 -N 48 -j2 ./model-mpi.exe
```

Some examples of hybrid invocation

- **To launch a Hybrid MPI/OpenMP application on 64 nodes**
  - 256 total ranks, using 1 CPU per Compute Unit (Max 24 Threads)
  - Use 4 PEs per node and 6 Threads per PE
  - Threads set by exporting `OMP_NUM_THREADS`

```
$ export OMP_NUM_THREADS=6
$ aprun -n 256 -N 4 -d $OMP_NUM_THREADS -j1 ./model-hybrid.exe
```

- **Launch the same hybrid application with 2 CPUs per CU**
  - Still 256 total ranks, using 2 CPU per Compute Unit (Max 48 Threads)
  - Use 4 PEs per node and 12 Threads per PE

```
$ export OMP_NUM_THREADS=12
$ aprun -n 256 -N 4 -d $OMP_NUM_THREADS -j2 ./model-hybrid.exe
```

Much more detail in later session on advance placement and binding.

ECMWF PBS Job Directives

- ECMWF have created a bespoke set of job directives for PBS that can be used to define the job.
- The ECMWF job directives provide a direct map between aprun options and PBS jobs.

Description	Aprun Option	EC Job Directive
Total PEs	`-n <n>`	`#PBS -l EC_total_tasks=<n>`
PEs per node	`-N <N>`	`#PBS -l EC_tasks_per_node=<N>`
Threads per PE	`-d <d>`	`#PBS -l EC_threads per_task=<d>`
CPUs per CU	`-j <j>`	`#PBS -l EC_hyperthreads=<j>`

A Note on the mppwidth, mppnppn, mppdepth and select notation

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.

More recent versions of PBS Pro support “select” syntax which is in use at other Cray sites.

Support for select has been discontinued at ECMWF in favour of a customised schema.

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

Example qstat output

```
Job id            Name             User              Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
123557.sdb        g1ma:t_wconstA   rdx                      0 Q of
123610.sdb        test.sh          syi                      0 Q of
123654.sdb        getini_1         emos              00:00:09 R os
123656.sdb        getini_1         emos              00:00:00 R os
123675.sdb        getini_1         emos              00:00:00 R os
123677.sdb        getini_1         emos              00:00:00 R os
123680.sdb        getini_1         emos              00:00:09 R os
123682.sdb        getini_1         emos              00:00:02 R os
123686.sdb        job_nf           co5                      0 Q of
123694.sdb        job_nf           co5                      0 Q of
123739.sdb        job_nf           co5                      0 Q nf
```

Example xtnodestat output (cct)

```
Current Allocation Status at Thu Feb 06 15:16:14 2014

     C0-0
  n3    --S---------;
  n2 SSS--S----------
  n1 SSS--S----------
c0n0    SSS----------
    s0123456789abcdef

Legend:
   nonexistent node                          S  service node
;  free interactive compute node             -  free batch compute node
A  allocated (idle) compute or ccm node      ?  suspect compute node
W  waiting or non-running job                X  down compute node
Y  down or admindown service node            Z  admindown compute node

Available compute nodes:          1 interactive,          45 batch
```

Pre/Post Processing Mode

This is a new mode for the Cray XC30.

- **Designed to allow multi-user jobs to share compute nodes**
  - More efficient for apps running on less than one node
  - Possible interference from other users on the node
- **Uses the same fully featured OS as service nodes**
- **Multiple use cases, applications can be:**
  - entirely serial
  - embarrassingly parallel e.g. fork/exec, spawn + barrier.
  - shared memory using OpenMP or other threading model.
  - MPI (limited to intra-node MPI only*)
- **Scheduling and launching handled by PBS**
  - Similar to any normal PBS cluster deployment.

Understanding Module Targets

- **The wrappers, cc, CC and ftn are cross compilation environments that by default target the compute nodes.**
  - This means compilers will build binaries explicitly targeting the CPU architecture in the compute nodes
  - It will also link distributed memory libraries by default.
  - Binaries built using the default settings will probably not work on serial nodes or pre/post processing nodes.
- **Users many need to switch the CPU target and/or opt to remove network dependencies.**
  - For example when compiling serial applications for use on esLogin or pre/post processing nodes.
- **Targets are changed by adding/removing appropriate modules**

Cray XC30 Target modules

CPU Architecture Targets	Network Targets
• `craype-ivybridge` (default) Tell compiler and system libraries to target Intel Ivybridge processors • ~~`craype-sandybridge`~~ ~~Tell compiler and system libraries to build binaries targeting Intel Sandybridge processors~~	• `craype-network-aries` (default) Link network libraries into binary (ESM mode). • `craype-target-local_host` Do not link network libraries into the binary (serial apps non-ESM mode).

- **For MAMU nodes use "`mpiexec`" from module cray-smplauncher instead of "`aprun`" if you want support for MPI**

Cluster Compatibility Mode (CCM)

- CCM creates small TCP/IP connected clusters of compute nodes.
- It was designed to support legacy ISV applications which could not be recompiled to use ESM mode.
- There is a small cost in performance due to the overhead of using TCP/IP rather than native Aries calls.
- Users also are also responsible for initialising the application on all the participating nodes, including any daemon processes.
- Still exclusive use of each node by a single users, so only recommended when recompilation unavailable.