Moving from Hopper (Cray XE6) to Edison (Cray XC30)

Zhengji Zhao
NERSC User Services Group

NUG 2013, February 25, 2013
• Easy to use
  – Edison programming environment is remarkably similar to that of Hopper

• Supports a rich set of production software applications, libraries, and tools needed by the entire NERSC workload
  – A robust set of programming languages, models
  – A rich set of highly optimized libraries, tools and applications
  – Community and pre-packaged applications
  – Shared-object libraries and socket communication (CCM)

• Enables effective application performance at scale, single node (high-throughput computing), and everything in between
## Edison Phase 1 Configuration

<table>
<thead>
<tr>
<th></th>
<th>Hopper</th>
<th>Edison Phase 1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute node</strong></td>
<td>Two 12 core MagnyCours processors</td>
<td>Two 8 core Intel Sandy Bridge processors</td>
<td>Hyper-Threading is available on Edison</td>
</tr>
<tr>
<td>System size</td>
<td>6384 nodes, 153216 cores</td>
<td>664 nodes, 10624 cores</td>
<td>Smaller in size</td>
</tr>
<tr>
<td>Cores per node</td>
<td>24</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Clock speed</td>
<td>2.1 GHz</td>
<td>2.6GHz</td>
<td>Faster processor</td>
</tr>
<tr>
<td>Memory</td>
<td>32 GB DDR3 1300 MHz 1.3GB/core</td>
<td>64GB DDR3 1600 MHz 4GB/core</td>
<td>Faster, larger per core memory</td>
</tr>
<tr>
<td>NUMA Domain</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
## Edison Phase 1 configuration

<table>
<thead>
<tr>
<th>Interconnect</th>
<th>Hopper</th>
<th>Edison Phase 1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gemini interconnect with 3D-Torus</td>
<td>Aries with dragonfly network topology</td>
<td>Next generation interconnect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Login nodes</th>
<th>Hopper</th>
<th>Edison Phase 1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 quad-core 2.4 GHz AMD Opteron 8378</td>
<td>quad-core, quad-socket (16 total cores) 2.0 GHz Intel &quot;Sandy Bridge&quot; processors with 512 GB memory.</td>
<td>“Fatter” login nodes</td>
</tr>
<tr>
<td></td>
<td>4 8-core AMD 2.0 GHz Opteron 6128 processors (32 cores)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Edison Phase 1 configuration

<table>
<thead>
<tr>
<th>Lustre file system</th>
<th>Hopper</th>
<th>Edison Phase 1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Disk</td>
<td>2.2 PB with 70 GB/s bandwidth</td>
<td>1.6 PB with 36 GB/s bandwidth</td>
<td></td>
</tr>
<tr>
<td>IO Servers (OSSs)</td>
<td>52</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>OSTs</td>
<td>312</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>Storage system</td>
<td>LSI 7900</td>
<td>Cray Sonexion 1600</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Batch system</th>
<th>Hopper</th>
<th>Edison Phase 1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torque/Moab</td>
<td>Internal batch server</td>
<td>External batch server</td>
<td>There are some advantages</td>
</tr>
</tbody>
</table>
New features and technologies available on Edison Compared to Hopper

- **Cray Aries interconnect with Dragonfly topology**
  - Scalable global bandwidth with lower network cost,
  - Adaptive routing
  - Designed to support both MPI and PGAS programing models.

- **Hyper-Threading Technology**
  - Some of the codes may get benefit from using HT
  - Benefit: time to solution vs charging

- **Cray Sonexion Storage System**
  - Cray’s large-scale parallel storage solutions for HPC

- **External Batch Server**
  - Single batch server
  - Torque commands (eg., pbsnodes) and environment variables (eg., PBS_NODEFILE) work as they do on Carver.
### Supported Programming Languages, Programming models, and compilers

<table>
<thead>
<tr>
<th>Supported Programming Languages</th>
<th>Supported Programming Models</th>
<th>Supported Compilers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>MPI</td>
<td>Intel*</td>
</tr>
<tr>
<td>C, C++</td>
<td>OpenMP</td>
<td>Cray</td>
</tr>
<tr>
<td>UPC</td>
<td>Cray Shmem</td>
<td>GNU</td>
</tr>
<tr>
<td>Python, Perl, Shells</td>
<td>POSIX Threads</td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>POSIX Shared Memory</td>
<td></td>
</tr>
<tr>
<td>Chapel</td>
<td>UPC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coarray Fortran</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chapel</td>
<td></td>
</tr>
</tbody>
</table>

*) Intel compiler is the default compiler on Edison.
PGI compiler is not supported on Edison.
## Cray Scientific and Math Libraries

<table>
<thead>
<tr>
<th>Libraries</th>
<th>Intel</th>
<th>Cray</th>
<th>GNU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LibSci:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAPACK, ScaLAPACK, BLACS, PBLAS</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Third Party Scientific Libraries:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUMPS, SuperLU, ParMETIS, HYPRE, Scotch</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Trilinos</strong></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>FFTW</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>PETSc</strong></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>DMAPP API for Aries</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>MPI-IO Library</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>IO Libraries:</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>HDF5, NetCDF, Parallel-netcdf</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Intel MKL</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
## More libraries, Tools, and Applications

<table>
<thead>
<tr>
<th>Scalable Debuggers</th>
<th>Performance Tools</th>
<th>Libraries and Tools</th>
<th>Applications</th>
<th>Visualization tools</th>
<th>ACTS Libraries and tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDT</td>
<td>Craypat/Apprentice2</td>
<td>Python</td>
<td>VASP</td>
<td>VisIt</td>
<td></td>
</tr>
<tr>
<td>TotalView</td>
<td>PAPI</td>
<td>GSL</td>
<td>QUANTUM ESPRESSO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATP</td>
<td>IPM</td>
<td>NCAR</td>
<td>LAMMPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADIOS</td>
<td>NAMD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BerkelyGW</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOLPRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>G09</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Our plan is to support all commonly used software and tools available on Hopper and Edison – please request if you find anything is missing
How to compile-
Use compiler wrappers

• Use module avail to see the available software
• For Cray and GNU programming environment, all Cray scientific and math libraries are available.
  – Compile codes as you do on Hopper
• Under Intel programming environment, some math and Cray scientific libraries are not available, noticeably libsci, use MKL for math routines
  – ftn test_scalapack.f90 –mkl=cluster
• There is a separate talk for compilation (Mike Stewart) to help you to make the transition from PGI to Intel.
How to run-
Use the similar job script on Hopper

• **To run with 2 nodes:**
  
  ```
  #PBS -N test  
  #PBS -q debug  
  #PBS -l mppwidth=32  
  #PBS -l walltime=0:30:00  
  #PBS -j oe  
  
  cd $PBS_O_WORKDIR  
  aprun -n 32 ./a.out
  ```
Sample job script to run MPI+OpenMP jobs

• To run an Intel binary a.out with 2 nodes:

```bash
#!/bin/bash
-­‐l
#PBS -q debug
#PBS -l mppwidth=32
#PBS -l walltime=0:30:00
#PBS -j oe

cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=8
export KMP_AFFINITY=compact
aprun –n4 –N2 –S1 –d8 –cc numa_node –ss ./a.out
```
Process/memory/thread affinity on Edison

• Process/memory/thread affinity works fine under the Cray and GNU programming environment.

• For Intel programming environment, the Process/memory affinity works fine for pure MPI jobs, but the thread affinity does not work properly. The work around is to use the env KMP_AFFINITY now.

• If you need to run MPI_OpenMP jobs with any number of threads, please experiment with the aprun’s –cc and –ss options and with env KMP_AFFINITY.

• Cray code xthi.c can be used for your test
What is Hyper-Threading (HT)?

- HT is Intel's term for its simultaneous multithreading implementation. For each physical processor core, the operating system addresses two virtual or logical cores, and schedules two threads or processes simultaneously (duplicating architectural state but sharing the execution resources).

- When the processor is stalled due to cache miss, branch misprediction, or data dependency, a HT equipped processor executes another scheduled task.

- According to Intel the first implementation only used 5% more die area than the comparable non-hyperthreaded processor, but the performance was 15–30% better.
How to run with HT

• To run with 2 nodes:
  
  #PBS -q debug
  #PBS -l mppwidth=64
  #PBS -l mppnppn=32
  #PBS -l walltime=0:30:00
  #PBS -j oe

  cd $PBS_O_WORKDIR
  aprun –j2 -n 64 ./a.out
Sample job script to run MPI+OpenMP jobs with HT

• To run with 2 nodes:
  
  `#!/bin/bash -l
  #PBS -q debug
  #PBS -l mppwidth=64
  #PBS -l mppnppn=32
  #PBS -l walltime=0:30:00
  #PBS -j oe

  cd $PBS_O_WORKDIR
  export OMP_NUM_NUM_THREADS=16
  export KMP_AFFINITY=compact
  aprun -j2 -n 4 -N2 -S1 -d16 -cc numa_node -ss ./a.out`
Edison Queues

• Configuration is subject to changes
• Submit production jobs to regular queue.
  – qstat –Qf to see the queue configuration detail
  – qstat –q to see if batch queues are enabled.
• 24 hours wall limit for now
Open Issues

• Intel programming environment
  – Please report any missing items in the intel programming environment.

• Batch system issue
  – Opened bug with Adaptive working for a solution

• IO performance
  – Still don’t meet the target, Cray is working on this.

• DVS performance issue
  – Do not do shared file IO, especially file read against the global file systems, eg., /gscratch, /project, etc.

• Core affinity
  – Intel binary thread affinity, work with KMP_AFFINITY to get the best thread affinity as you can.
  – In practice working with KMP_AFFINITY=disabled, and aprun ... –cc numa_node or –cc none maybe preferred
Summary

• Edison has **faster processors with a larger memory per core** when compared to Hopper.

• The programming environment on Edison is very similar to that of Hopper, except **Intel compiler** is the default on Edison, and **MKL** is used.

• Uses same job script as on Hopper, just **Edison has 16 cores per node**.

• If you run with **HT**, Edison has **32 cores per node**. Run with `mppnppn=32` and `aprun –j2`. Experiment if HT benefits your codes.
Quick start for Hopper users

1. ssh edison@nersc.gov -l YourUserName
2. ftn mycode.f90
3. Write job script
   zz217@hopper08:~> cat run.pbs
   #PBS -q debug
   #PBS -l mppwidth=32
   #PBS -l walltime=0:05:00
   #PBS -j oe
   
   cd $PBS_O_WORKDIR
   aprun -n 32 ./a.out
4. qsub run.pbs
5. Analyze results