Cray system specifications

- Overview
  - Hardware
  - Compute resource
  - Beagle2 Nodes
    - Accelerator/GPU nodes with Nvidia Tesla K20X, Kepler GK110
    - Traditional Compute Nodes (XE6) with AMD Opteron 6380, "Abu Dhabi" processors
    - MOM Nodes
    - Login Nodes
  - The Gemini Interconnect
  - Storage Resource
  - Network
    - High speed network (HSN)
  - Software
  - Operating system
  - Other XE systems

Overview

Beagle is a Cray XE6 system, whose design diverges substantially from a standard cluster architecture. The Cray XE6 can scale to up to one million processor cores. It allows for global memory access and a high rate of small message (MPI message rates 20 times faster than a XT5 supercomputer). The XE6 interconnect – based on the ASIC Gemini – utilizes error correcting code (ECC) and adaptive routing hardware (which spreads data packets over the four available lanes which comprise each of the torus links) provide improved system and applications resiliency. Moreover, the system contains redundant power supplies and voltage conversion modules to increase reliability at scale. Cray provides an integrated compilation environment including Fortran/C/UPC/CAF/C++. Compilers from GNU and PGI are also available. In addition, the XE6 has Industry leading “sustained performance” energy efficiency.

Extreme Scalability Mode (ESM)

To execute an application on Beagle2's compute nodes in the ESM execution environment, you must invoke the aprun application launch command in your batch job script. Submitting your batch job to TORQUE (using the qsub command) places your job on one of the aprun service nodes. These nodes have limited resources shared between all users on the system and are not intended for computational use. You must invoke the aprun command in your job script or from the aprun command line to launch your application on one or more compute nodes in the ESM execution environment.

Cluster Compatibility Mode (CCM)

The CCM environment provides the Linux services needed to run applications that run on most standard x86_64 cluster-based systems. The CCM execution environment emulates a Linux-based cluster, allowing you to launch standard applications that won't run in the ESM environment.

To run a batch job that executes an application on Beagle2's compute nodes in the CCM execution environment, you must add the ccm module to your user environment with this module load command:

```
module load ccm
```

You can add this line to your TORQUE batch job script (after your TORQUE directives and before your executable lines).

For batch jobs, include the -l gres=ccm flag as a TORQUE directive in your job script:

```
#PBS -l gres=ccm
```

For more about the Cray Linux Environment (CLE), see Workload Management and Application Placement for the Cray Linux Environment (in PDF format).

Hardware

Beagle2 has currently 10 XE6 cabinets:

- 182 blades are compute-node blades (728 compute nodes / 4 nodes per blade = 182 blades).
- There are also Service Nodes:
  - 3 job management nodes (MOM nodes, Torque batch system)
  - 5 login nodes – 10Gbps NICs
  - 16 InfiniBand (IB) nodes, object store server (Lustre OSS nodes)
  - 1 boot node (ALPS resource manager, shared filesystem server)
  - 1 sdb node (system database, ALPS resource manager, Moab/Torque batch system, NFS file server)
  - 6 DSL service nodes (Provide dynamically-linked libraries for CCM mode)
  - 1 monitor service node (Nagios, Ganglia)
  - 4 fiber channel nodes (Lustre MDS nodes)
  - 1 admin service node (Gold allocation system, NAT service for Lustre MDS nodes.)

All nodes do not have local storage, but access a shared file system. Lustre File System is direct-attached and it is **not backed up.**
Compute resource

- CPU Cores/processor: 16-core, 2.5-GHz, 16 MB L2, Cache 16MB L3, **AMD Opteron 6380** series processors “Abu Dhabi”.
- **L1 cache**: 8 x 64 KB shared instruction caches, 16 x 16 KB data caches; **L2 cache**: 8 x 2 MB shared exclusive caches, **L3 cache**: 2 x 8 MB shared caches
- Cores per node: 32 (2 sockets per node)
- **CPU Nodes**: 724
- **GPU** Cores/processor: 2496, 2.6-GHz, memory size: 5 Gb, **Nvidia K20 GPU**
- **GPU Nodes**: 4
- Total CPU compute cores: ~23000
- Peak performance: 212 TFlops/s
- Total memory (TB): 64GB per node * 724 compute nodes = 46336GB, 32GB per node * 4 GPU compute nodes = 128GB
- Max memory bandwidth for the Opteron 6380 is 102.4 GB/s

Beagle2 Nodes

Beagle2 is equipped with three node types: traditional compute nodes (XE6), accelerated compute nodes (GPU enabled compute nodes), and job management nodes (MOM nodes). Each node type is introduced below.

**Accelerator/GPU nodes with Nvidia Tesla K20X, Kepler GK110**

Accelerator/GPU nodes are equipped with one AMD Opteron 6380 CPU processor and one **NVIDIA K20X accelerator**. The CPU acts as a host processor to the accelerator. The NVIDIA accelerator does not directly interact with the Gemini interconnect. Each GPU node has 32 GB of system memory while the accelerator has 6GB of memory. To read more about GPU accelerated computing please follow this [link](#).

<table>
<thead>
<tr>
<th></th>
<th><strong>Tesla K20X</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stream Processors</strong></td>
<td>2688</td>
</tr>
<tr>
<td><strong>Core Clock</strong></td>
<td>732MHz</td>
</tr>
<tr>
<td><strong>Shader Clock</strong></td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Memory Clock</strong></td>
<td>5.2GHz GDDR5</td>
</tr>
<tr>
<td><strong>Memory Bus Width</strong></td>
<td>384-bit</td>
</tr>
<tr>
<td><strong>VRAM</strong></td>
<td>6GB</td>
</tr>
<tr>
<td><strong>Single Precision</strong></td>
<td>3.95 TFLOPS</td>
</tr>
<tr>
<td><strong>Double Precision</strong></td>
<td>1.31 TFLOPS (1/3)</td>
</tr>
<tr>
<td><strong>Transistor Count</strong></td>
<td>7.1B</td>
</tr>
<tr>
<td><strong>TDP</strong></td>
<td>235W</td>
</tr>
<tr>
<td><strong>Manufacturing Process</strong></td>
<td>TSMC 28nm</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Kepler</td>
</tr>
</tbody>
</table>

**Traditional Compute Nodes (XE6) with AMD Opteron 6380, "Abu Dhabi" processors**
The XE6 dual-socket nodes are populated with 2 AMD Opteron 6380 processors.

<table>
<thead>
<tr>
<th>CPU Name:</th>
<th>AMD Opteron 6380</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Characteristics:</td>
<td>3.4GHz, Turbo CORE off</td>
</tr>
<tr>
<td>CPU MHz:</td>
<td>2500</td>
</tr>
<tr>
<td>CPU MHz Maximum:</td>
<td>3400</td>
</tr>
<tr>
<td>FPU:</td>
<td>Integrated</td>
</tr>
<tr>
<td>CPU(s) enabled:</td>
<td>32 cores, 2 chips, 16 cores/chip</td>
</tr>
<tr>
<td>CPU(s) orderable:</td>
<td>1-2 chips</td>
</tr>
<tr>
<td>Primary Cache:</td>
<td>512 KB I on chip per chip, 64 KB I shared / 2 cores; 16 KB D on chip per core</td>
</tr>
<tr>
<td>Secondary Cache:</td>
<td>16 MB I+D on chip per chip, 2 MB shared / 2 cores</td>
</tr>
<tr>
<td>L3 Cache:</td>
<td>16 MB I+D on chip per chip, 8 MB shared / 8 cores</td>
</tr>
<tr>
<td>Other Cache:</td>
<td>None</td>
</tr>
<tr>
<td>Memory:</td>
<td>64 GB (8 x 8 GB 2Rx4 PC3L-12800R-11, ECC)</td>
</tr>
<tr>
<td>Disk Subsystem:</td>
<td>None</td>
</tr>
<tr>
<td>Other Hardware:</td>
<td>None</td>
</tr>
<tr>
<td>Base Threads Run:</td>
<td>32</td>
</tr>
<tr>
<td>Minimum Peak Threads:</td>
<td>--</td>
</tr>
<tr>
<td>Maximum Peak Threads:</td>
<td>--</td>
</tr>
</tbody>
</table>
MOM Nodes

MOM nodes are where PBS scripts are executed and the aprun command is launched. All scripts and executables run outside of aprun command are executed on the processor of the MOM node (and usually it is a bad idea). They do not participate in MPI applications and they are service nodes not compute nodes. These nodes are in the Gemini high speed network (HSN). Avoid as much as possible to run any computations on MOM nodes as it might seriously affect other user’s computations and it is therefore against the usage policy. Such jobs usually overload MOM nodes and might even kill all the jobs run on that MOM node. Usage is monitored, and violations will not be tolerated.

Login Nodes

The login nodes have no physical disks, so the /tmp directory is actually RAM. Any data you try to stage from there will have to be copied via NFS through the mom node to the compute, so you’re not going to see good throughput. It could also have a very adverse affect on the system, putting that much load on the three mom nodes that serve the 700+ compute nodes.

The Gemini Interconnect
• Interconnect: Each pair of nodes (containing a total of 4 sockets or 64 cores) is connected to 1 Gemini Interconnect Application-Specific Integrated Circuit, ASIC.
• “Gemini” network is connected in a 3D torus.
• Latency between cores is $< 1 \mu s$ latency for two cores connected to the same Gemini chip (traffic between the two nodes connected to a single Gemini is routed internally), and a little over $1 \times 10^3 \mu s$ for two cores connected to different Gemini chips. Each Gemini chip has 168 GB/s bandwidth of switching capacity and provides about 20 GB/s injection bandwidth per node. The Gemini chips are arranged in a 3D torus (8 x 12 x 8, looped in the sense that last in each direction is connected with the first) (from “The Gemini Network” paper by Cray inc. described below)
• End-to-end latency in a Gemini network is determined by the end-point latency and the number of hops. On a quiet network, the end-point latency is $1.0 \times 10^3 \mu s$ (2100 cycles of the processor) or less for a remote put, $1.5 \times 10^3 \mu s$ (3050 cycles) or less for a small MPI message. The per hop latency is 105ns (220 cycles) on a quiet network.
• MPI: $\sim 1-2 \mu s$ of end-point latency, up to 32 MPI task simultaneously. This means a minimum latency of ~2000 to 4000 cycles.
• Communication (between cores that are not associated with the same Gemini chip) goes to the source core to the Gemini chip associated with that core, then over the torus network to another Gemini chip, then to the destination core, and it will add 220 cycles of latency for each node that it needs to hop. In a call the First 64-bit word will arrive in $1 \times 10^3 \mu s$, while the second will arrive 3.2 nanoseconds (or 6.7 cycles) later (9.5 bits per cycle per node or .4 bits per cycle per core).
• In addition to supporting MPI over the standard programming languages of C, C++ and Fortran, the Gemini interconnect has direct hardware support for partitioned global address space (PGAS) programming models including Unified Parallel C (UPC), Co-array Fortran and Chapel. Gemini allows remote references to be pipelined in these programming models that can result in orders-of-magnitude performance improvement over library-based message passing models. This feature brings highly scalable performance to communication-intensive, irregular algorithms that until now have been limited by the MPI programming paradigm.
• Memory: because of the fast interconnect, and how the Gemini operates, the XE6 can be considered a logically shared memory machine even though its memory is technically distributed across processors.

For more information on the Gemini ASIC, a good source of information is the paper by Cray inc. The Gemini Network.

Some details about the design of the Gemini ASIC are displayed in the image from “The Gemini Network” paper by Cray inc. described above.

### Storage Resource

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>31@gni/lustrefs</td>
<td>450T</td>
<td>/lustre/beagle</td>
</tr>
<tr>
<td>573@gni/beagle2</td>
<td>1.6P</td>
<td>/lustre/beagle2</td>
</tr>
</tbody>
</table>

Infiniband-connected

• The Lustre filesystem data is stored on two DDN 10000 and two DDN 12000 storage arrays, connected via Infiniband to sixteen dedicated Cray XIO service nodes.
• These storage arrays provide 600TB raw (450TB usable) and 2.0PB (1.6PB usable) of storage for the Lustre fast scratch filesystems.
• The Lustre filesystems metadata is stored on two Fiber Channel storage arrays connected to a dedicated XIO service nodes.

### Network

#### High speed network (HSN)

Beagle has a High Speed Network (HSN) with a 10-Gb connection to the Argonne Mathematics and Computer Science (MCS) Division’s HPC switch, which has 10-Gb connectivity to MREN and ESNet. Overall is 10-Gb to the University of Chicago campus.

Each login node (5), storage node (5) have 2 1-Gb bonded connections to the Beagle switch. The Beagle2 switch is a Juniper EX4200.

### Software

• Programming Languages: Each Cray XE6 system includes a fully integrated Cray programming environment with C, C++ and Fortran, plus supported parallel programming models including MPI, OpenMP, Cray SHMEM, UPC and Co-Array Fortran.
• Compilers: include those from GNU and Cray.
• Libraries: High performance-optimized math libraries of BLAS, FFTs, LAPACK, ScaLAPACK, SuperLU and Cray Scientific Libraries.
• MPI: The MPI implementation is compliant with the MPI 2.0 standard and is optimized to take advantage of the Gemini interconnect in the Cray XE6 system.
• Profiling: performance analysis tools CrayPat; with Cray Apprentice2; allow users to analyze resource utilization throughout their code at scale and eliminate bottleneck and load imbalance issues.
• Debugging: TotalView Technologies and Alinea and many open source programming tools.

Operating system

The Cray XE6 system ships with Cray Linux Environment v3 (CLE3), a suite of high performance software including a SUSE Linux-based operating system designed to run large, complex applications and scale to more than 1 million processor cores. The Linux® environment features Compute Node Linux (CNL) as the default compute kernel. When running highly scalable applications, CNL runs in Extreme Scalability Mode (ESM), which ensures operating system services do not interfere with application scalability. Real world applications have proven this optimized design scales to more than 200,000 cores and is capable of scaling to more than 1 million cores on the Cray XE6 supercomputer. Users can also run industry-standard Independent Software Vendor (ISV) computations. CLE3 accomplishes this through the new Cluster Compatibility Mode (CCM). CCM allows out-of-the-box compatibility with Linux/ x86 versions of ISV software – without recompilation or relinking – and allows for the use of various versions of MPI (e.g., MPICH, Platform MPITM). At job submission, the user can request the CNL compute nodes be configured with CCM, complete with the necessary services to ensure Linux /x86 compatibility. The service is dynamic and available on an individual job basis.


References: CRAYDOC

Other XE systems

NERSC Hopper Hopper2
ERDC Garnet
ARSC Chugach
CSCS Rosa
CUNY Salk