Cray CX1 at IUCAA

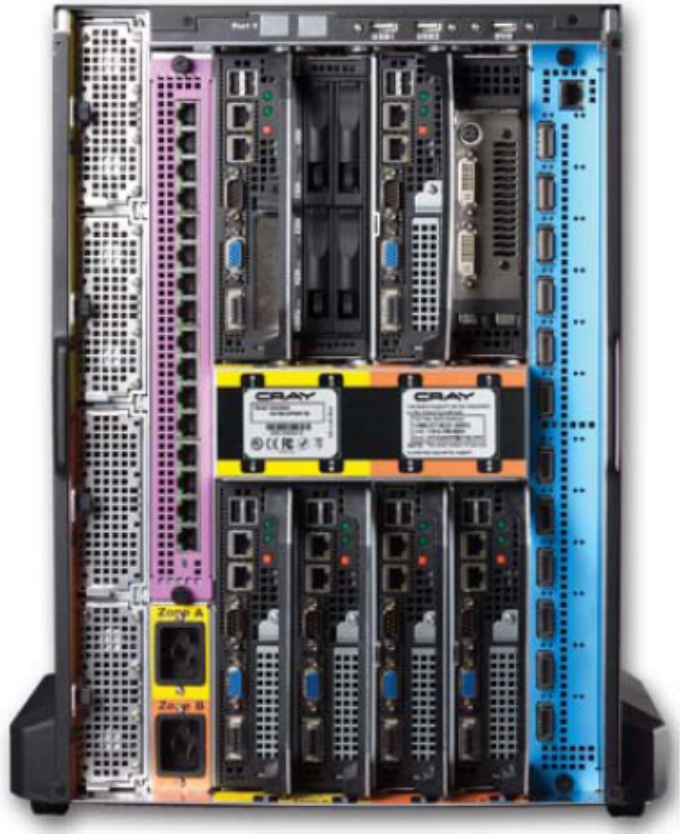
An Overview

Jayanti Prasad

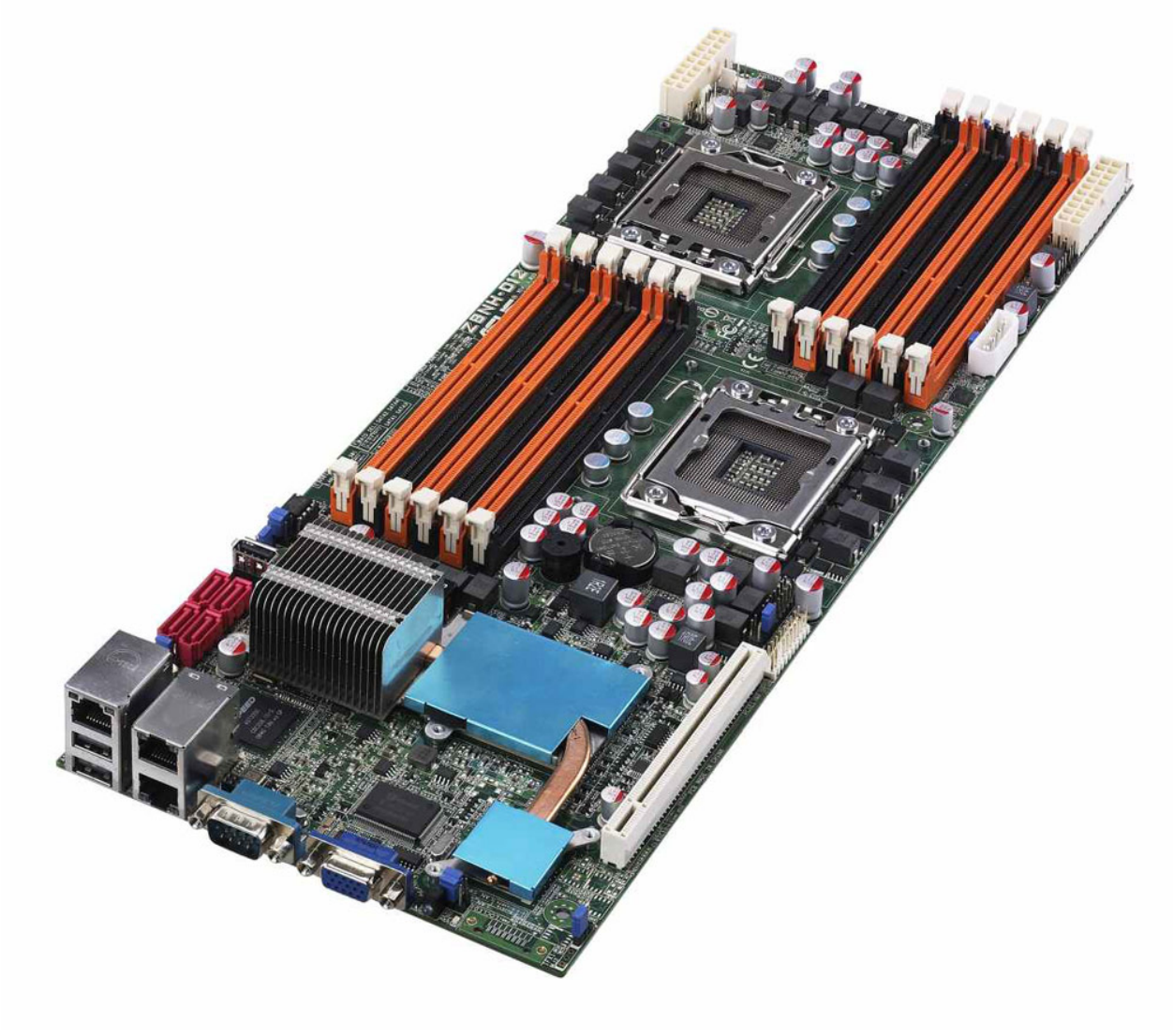
Inter-University Centre for Astronomy & Astrophysics
Pune, India (411007)

Sept 07, 2011

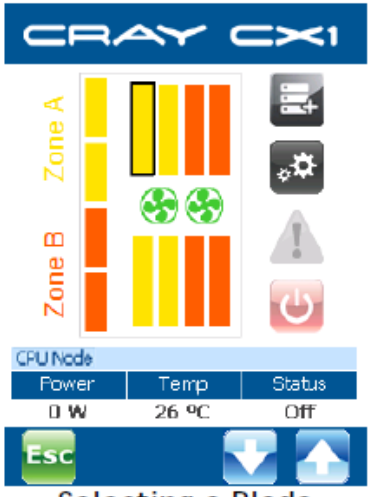
CRAY CX1 : Front and back view



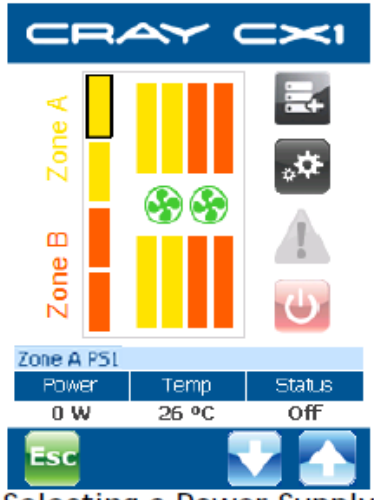
CRAY CX1 : motherboard



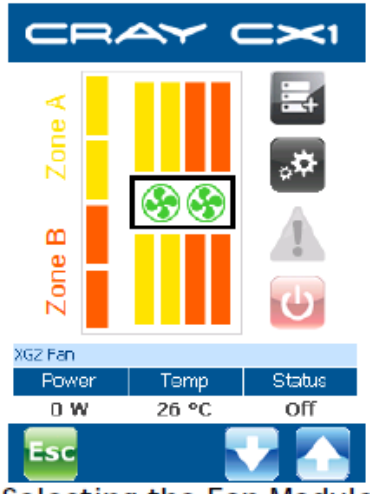
CRAY CX1 : Control panel



Selecting a Blade



Selecting a Power Supply



Selecting the Fan Module

Overview

- Cray CX1 is a desktop/desk-side supercomputer with 1 head node, 1 GPU node and 4 compute nodes.
- Each node has two 2.67 Ghz Intel Xeon(5650) processors, with every processor having six cores.
- In total there are 72 cores on the systems. Since there are twelve cores on every node, one can use twelve OpenMP hardware threads.
- Every node has 24 GB RAM and in total the system has 144 GB RAM.
- At present there is around 5 TB disk space available for users on three different partitions, named /home, /data1/ and /data2.
- Theoretical computing power of the system on CPU is 768.96 GF and on GPU is 515 GF (with 3GB Memory).

Compilers

- Names of the nodes are as **hpccmb** (head node), **compute-00-00** (GPU node) and **compute-00-01, compute-00-02** and , **compute-00-04**.
- There are many software, packages and libraries already installed in the system and more (open source) can be installed. At present the system has the following compilers:
  1. gcc version 4.1.2 (**gfortran** and **gcc**).
  2. Compilers supplied with the *platform* system (**mpicc**, **mpiCC**, **mpif77**, **mpif90** etc.,) are installed in "/opt/platform_mpi/".
  3. Open MPI compilers (**mpic++ ,mpicc, mpiCC, mpicxx, mpiexec, mpif77** etc.,) installed in " /data1/software/openmpi".
  4. Intel compilers (**icc** and **ifort**) are installed in "/opt/intel/". In this area Intel Mathematical Kernel Library (MKL) and Intel Threading Building Blocks (ITBB) are also installed.

Packages

- Platform LSF is successfully running on the system and by default all MPI jobs will be assigned in a "batch queue" mode.
- In general, users are not needed to specify the nodes to which they want to assign the job, however, it may be useful in some cases.
- The system already has many scientific packages/libraries pre-installed or have been installed, some of them are as follows:
  1. **blacs, linpack, scalapack** and **hdf5** supplied with the system are installed in "/opt".
  2. **gnuplot, gv, acoread** etc., are installed.
  3. A good number of packages including **fftw2, fftw3, pgplot, cfitsio, lapack** are installed in "/data1/software".
  4. Some of the packages related to CMBR work (**cmbfast, camb, cosmomc, healpix etc.,**) are already been installed in "/data1/soft_cmbr/" and more will be installed.

Running the Jobs

- Job on the system can be submitted using two different modes. In the first case job can be submitted using a *platform* GUI supplied with the system which can be accessed at the following URL from the IUCAA network.

  `http://192.168.11.243:8080/platform/,`

- Jobs can be submitted from the terminal also and a typical MPI submission script may look like:

```
mpirun  -np 32 -hostlist "compute-00-00,compute-00-01,compute-00-02,compute-00-04"  \
./cosmomc params.ini
```

  apart from this the direct submission also works:

```
mpirun -np 4 ./a.out
```

  I think one of the most useful ways to submit a MPI job, particularly when it uses OpenMP also is by using the *appfile*. Create an appfile with the following content:

```
cat appfile
-h compute-00-01 -np 2 ./cosmomc params.ini
-h compute-00-02 -np 2 ./cosmomc params.ini
-h compute-00-03 -np 2 ./cosmomc params.ini
-h compute-00-04 -np 2 ./cosmomc params.ini
```

  Once appfile is created it can be run using the following command.

```
mpirun -f appfile
```

GPU Computing

- The system has one Nvidia Tesla C2050 GPU which is hosted on the node **compute-00-00**. Detail status of the GPU can be printed on the standard output using the the **nvidia-smi** command one of its example is as follows:

  nvidia-smi -q

  This will print a lot of useful information about the GPU. You can use "–help" option also to find all the option available for **nvidia-smi**.
- Compute Unified Device Architecture (CUDA) library as well as the compiler **nvcc** is installed in the are "/usr/local/cuda". In order to run a cuda program, follow the following three steps:
  - Write a source/program file with suffix ".cu" following the cuda syntax.
  - Compile the program with **nvcc**:
    nvcc prog.cu
  - Run the executable: ./a.out

Specs of the GPU

```
 --- General Information for device 0 ---
Name: Tesla C2050
Compute capability: 2.0
Clock rate: 1147000
Device copy overlap: Enabled
Kernel execition timeout : Disabled
 --- Memory Information for device 0 ---
Total global mem: 2817982464
Total constant Mem: 65536
Max mem pitch: 2147483647
Texture Alignment: 512
 --- MP Information for device 0 ---
Multiprocessor count: 14
Shared mem per mp: 49152
Registers per mp: 32768
Threads in warp: 32
Max threads per block: 1024
Max thread dimensions: (1024, 1024, 64)
Max grid dimensions: (65535, 65535, 65535)
```

Thank You !