Description of the Euler GPU Cluster

Abstract

This report describes the hardware and software configuration used in the Euler GPU cluster of the Wisconsin Applied Computing Center (WACC) at the University of Wisconsin - Madison. The cluster is an interdepartmental effort to promote the use of graphics processing units (GPUs) in the field of high-performance scientific computing. We would like to thank Nvidia for their generous donation of the GPUs for this project.

History of versions

Version	Date	Author(s)	Changes
0.1	2011-05-01	aas	Initial checkin.
0.2	2011-06-01	aas	Change to focus more on hardware.
0.3	2011-07-02	aas	Added software description and reference manual sections.
0.4	2011-08-18	aas	Added job submission walkthrough.
0.5	2012-01-24	aas	Updated available hardware. Rename to WACC.

Contents

1 Hardware Description

The Euler cluster consists of fourteen GPU compute nodes and a head node. The SuperMicro SuperServer 7046GT-TRF[1] was selected as the base model for the compute nodes due to its support for four double-width PCIe 2.0 x16 and PCIe 2.0 x4 slots[2], redundant 1400W power supplies and proven reliability in the Newton GPU cluster. The head node is the SuperMicro SuperServer 6016T-URF, similar to that used for Newton.

1.1 Specifications

Each node has two quad-core Intel Xeon E5520 processors, 48GB ECC registered DDR3-1333 RAM, a 2TB enterprise-class Western Digital RE4 hard drive (WD2003FYYS), and a 40 Gbps Mellanox ConnectX-2 Infiniband VPI (MT26428). Additionally, each node has four Fermi-architecture Nvidia GPUs. The nodes are divided into two groups: large (GPU) memory and small (GPU) memory. The large memory nodes are called as such because they have four Tesla-class GPUs (C2050 and/or C2070) which have either 3GB or 6GB of memory. The small memory nodes have four GeForce-class (GTX 480) GPUs, each with 1.5GB of memory.

1.1.1 GPU Cards

Table 2: GPU Properties

	Tesla C2070	**Tesla C2050**	**GeForce GTX480**
CUDA Cores	448	448	448
Memory	6 GB	3 GB	1.5 GB
Memory Bandwidth	144 GB/s	144 GB/s	177.4 GB/s
ECC	Yes	Yes	No
Frequency	1.15 GHz	1.15 GHz	1.401 GHz
SP [peak]	1.03 Tflops	1.03 Tflops	1.344 Tflops
DP [peak]	515 Gflops	515 Gflops	128 Gflops

1.1.2 Head and Compute Nodes

The head node and each of the 14 compute nodes consists of the following:

- 48 GB RAM [6x8GB 1333MHz DDR3, ECC]
- 2x Intel Xeon E5520 [Quad-core Nehalem, 2.26 GHz]

[1] `http://www.supermicro.com/products/system/4U/7046/SYS-7046GT-TRF.cfm`

[2] At least five PCIe 2.0 slots are required for the four GPUs and single Infiniband HCA.

- 2TB Western Digital RE4 [WD2003FYYS]
- 40 Gbps Mellanox ConnectX 2 Infiniband VPI [MT26428]

It should be noted that the Infiniband VPIs for the compute nodes are in PCIe 2.0 x4 slots, thereby limiting their available bandwidth to only 20 Gbps.

2 Software Description

2.1 Operating System

The cluster runs Scientific Linux 6.2, a free derivative of RedHat Enterprise Linux (RHEL). An RHEL-based distribution was chosen due to its direct support by hardware manufacturers (e.g. for the Infiniband HCAs), its native support for technologies such as GPUDirect, and its proven reliability in the field of HPC. OpenSUSE was also considered, but ultimately not chosen.

2.2 Provisioning

The eXtreme Cloud Administration Toolkit (xCAT) from IBM is used to manage the installation, configuration, and overall administration activities of the compute nodes. Other packages such as ROCKS and Perceus were trialed, with xCAT being chosen due to its ease of configuration (everything is done in a text editor instead of via cryptic commands) and overall flexibility.

Compute nodes use a 'stateless' configuration - they are booted off of a pre-configured image that is copied from the file server to the node's RAM. Volatile mountpoints such as /home and /usr/local will be mounted during boot.

2.3 User Management

While it would be desirable to use a University-wide credentials system (such as NetIDs) for user management, security restrictions would not allow for this. Instead, Euler uses its own LDAP server for user management, with the ability to fallback on departmental LDAP servers (via SSSD) if desired.

2.4 Job Submission and Management

The TORQUE resource manager is used for both resource management and job scheduling. Maui will eventually assume the job scheduling tasks once better GPU support is available.

3 Reference Manual

3.1 Accounts

Users associated with one of the groups that sponsored the cluster may directly request an account from Andrew Seidl [aaseidl@wisc.edu]. Those not associated with one of these groups must first contact one of the group leaders with a project description.

Once your account is created, you will receive an email with your username, temporary password, associated group, and access instructions. Please change your password via the `passwd` utility as soon as possible. Also note that by default all users within your group will have access to your files. If you require more security, you can either manually `chmod` your directories to remove group permissions, or request that a personal group be created for you.

3.2 Remote Access and File Sharing

Euler is accessible from anywhere within the University network, however individual departments may have firewall restrictions in place that could prevent access. Please contact your departmental support person if you are unable to connect.

To access the cluster, SSH to the head node at euler.msvc.wisc.edu. Linux and MacOSX machines come with SSH by default; Windows users must first download a client such as PuTTY. After entering your username and password, you should now have a shell on the head node. Please note that the head node does not have any GPUs and should not be used for any computational work. To test programs interactively, please see the section on `qsub` below.

3.3 Available Software

The following packages are available on Euler:

- CUDA 3.2
- CUDA 4.0
- CUDA 4.1
- MATLAB 2011a
- mvapich 1.2.0
- mvapich2 (may not work with the Infiniband fabric)
- OpenMPI 1.4.3
- Povray 3.7RC3

3.3.1 CUDA

CUDA 3.2, 4.0, and 4.1 are available on the cluster and may be selected on a per-user basis, with 4.1 as the default. The active CUDA version may be chosen using Environment Modules. The command `module avail` will show all available options. For example, `module load cuda/3.2` will set the user's active CUDA version to 3.2. Please note that this command will need to be repeated for each login, unless the `module initswitch cuda/4.1 cuda/3.2` command is used.

3.3.2 MPI

As noted above, multiple versions of MPI are available on Euler. OpenMPI is the current default for all users. To use a different one, it is best to use the associated environment modules manage the environment variables for you. Please note that while you're free to use your own version of MPI, you should make sure the appropriate configure options are set to make use of our high-speed, low-latency Infiniband fabric.

3.3.3 Software Requests

Software requests may be sent to Andrew Seidl `[aaseidl@wisc.edu]`. These requests should be limited to widely-used software packages (those most likely to be provided by the Linux distribution) or certain commercial packages (such as MATLAB). Due to limited resources, we will only attempt to compile smaller software packages. Larger, more complex packages such as `OpenMM` or `Gromacs` will be your responsibility (though we may help, should time allow).

3.4 Job Submission with TORQUE

Jobs submitted to TORQUE are configured by a creating a shell script with special TORQUE/PBS[3]-specific configuration comments. An example of these scripts is provided in your home directory, under the sym-linked directory 'Example Jobs'. Please see the TORQUE documentation for full coverage.

The below is not meant to be a comprehensive guide to using TORQUE. For that, please see the actual documentation at `http://www.adaptivecomputing.com/resources/docs/torque/2.1jobsubmission.php`.

Each job is given an ID of the form n.euler where n is an incrementally-increasing integer. This ID is used to monitor job status, delete jobs, and to arrange output and error logs.

[3]TORQUE is a fork of PBS. Most commands retain their pbs prefixes.

3.4.1 qsub: Job Submission

All configuration options may be set in either the job submission script or on the command line. The most basic job can be submitted by `qsub script.sh`, where script.sh is the name of your job submission script. One useful argument is: -l nodes=n,ppn=x,gpus=y to reserve n nodes and y GPUs per node, each with x processors (ppn: processors per node). Another useful argument is for tasks: -t x-y,z which will run the job y-x+1 times, each with a different number from the set of the range x-y, and z. One place where this would be useful would be for selectively rendering frames of an animation.

By default, stdout and stderr are redirected to two files in your home directory, with the name jobname.oid and jobname.eid, respectively, where jobname is the job name you set and id is the job ID number.

If you need to run a program interactively, qsub -I will give you a shell on one of the compute nodes.

3.4.2 qstat: Job Status

Job status may be monitored via qstat. Without any arguments, this will list all recent jobs that have submitted. Next to each job ID will be a single character for the status. 'Q' for queued, 'R' for running, 'E' for error, and 'C' for completed. Qstat will also show how long the job has run and how much time allotted remains. Depending on your submission settings, you will also receive an email saying whether your job completed or failed, and why. By default this is sent to your account on the cluster itself, which is accessible by the command 'mail'.

3.4.3 qdel: Job Removal

Jobs can be removed or cancelled via the command `qdel id` where id is the job's ID number.

3.4.4 pbsnodes and pbstop: Node Monitoring

The command `pbsnodes -a` will give you details about all nodes in the cluster (available/used memory, number of jobs running on the node, various stats on the node's GPUs). The command pbstop will give you a nicer view of this data.

4 Job Submission Walkthrough

4.1 Logging In

To log in to Euler, you must use an SSH client to connect to `euler.msvc.wisc.edu`. One of the most popular clients on Windows is PuTTY, available from `http://www.`

chiark.greenend.org.uk/~sgtatham/putty/; Linux and Mac OS X come with SSH clients preinstalled, accessible from the terminal.[4] In PuTTY, enter `euler.msvc.wisc.edu` as the hostname and hit 'Connect'. You will be asked to enter the username and password that was emailed to you.[5] In a terminal on Linux or Mac OS X, type in `ssh` *`username`*`@euler.msvc.wisc.edu` and press Enter, where *username* is the username that was emailed to you. Once you have successfully logged in, you should be able to do an `ls` and see some files that have been automatically added to your home directory.

Listing 1: Logging in to Euler from Linux

```
andrew@mahasher ~ $ ssh aseidl@euler.msvc.wisc.edu
aseidl@euler.msvc.wisc.edu's password:
[aseidl@euler ~]$ ls
Example Jobs
[aseidl@euler ~]$
```

4.2 A Basic Job

As seen in Listing 1, your home directory contains a symlink to the folder `Example Jobs`. This folder contains some example TORQUE job submission scripts. Copy the `gpu-scan.sh` script to your home directory and submit it to the job manager using `qsub`.

Listing 2: gpu-scan.sh

```
#!/bin/sh

#PBS -N gpu-scan
#PBS -l nodes=1:gpus=1,walltime=00:01:00

$NVSDKCOMPUTE_ROOT/C/bin/linux/release/scan
```

The script in Listing 2 has two main sections: job settings and the script itself. Lines 3 and 4 set the job's name to 'gpu-scan' and requests one node with one GPU. Additionally, it tells the scheduler that the job will take at most 1 minute to execute. The last line executes the `scan` code example from the CUDA SDK.

There are three main things to note in this example. First, job settings are specified by `#PBS` followed by command line options for qsub. Since the settings are defined in comments, you can directly run this script from the command line without the TORQUE-specific options causing any issues. Next, all settings defined in the script can also be passed directly to qsub from the command line. See the `qsub` man

[4] In Gnome: `gnome-terminal`. In KDE: `konsole`. In Mac OS X: `Terminal.app`.

[5] Note that nothing will be displayed when you enter your password. Simply hit Enter after typing in your password.

page for more options. Lastly, note the use of the `$NVSDKCOMPUTE_ROOT` environment variable. This allows the same script to be used with differing versions of the CUDA SDKs, depending on which version you have selected via `module load cuda`.

Listing 3: Submitting a Basic Job to Euler

```
[aseidl@euler ~]$ cp Example\ Jobs/gpu-scan.sh .
[aseidl@euler ~]$ qsub gpu-scan.sh
8117.euler
[aseidl@euler ~]$ qstat 8117
Job id          Name             User            Time Use S Queue
--------------- ---------------- --------------- -------- - -----
8117.euler      gpu-scan         aseidl          00:00:01 C batch
```

Listing 3 shows how to copy the example job script to your home directory, submit it to the TORQUE job scheduler, and check on the status of the submitted job. When submitting the job, `qsub` will output the job ID assigned to it. This ID can then be use to check on the job's status with `qstat`. You can also enter `qstat` with no arguments to see the status of all recently submitted jobs.

The output of `qstat` shows the job ID assigned, the name given via the `-N` option, the user who submitted the job, walltime used, job status, and queue which the job was submitted to. Walltime is defined as the actual time that the job ran[6]. Status is a single letter, the most common of which are: Q (queued), R (running), and C (completed). The queue allows you to submit your job to different sections of the cluster which may have different resources available or different restrictions on parameters such as walltime, number of GPUs requested, etc. Euler currently has a single queue, called 'batch'.

4.3 Interactive Jobs

To run a job manually on a compute node, you can submit what is called an 'interactive job' to TORQUE. This is done via the command `qsub -I`. Once the appropriate resources are available, you will be presented with a shell on one of the compute nodes, as shown in Listing 4.

Listing 4: Starting an Interactive Job on Node 1

```
[aseidl@euler ~]$ qsub -I
qsub: waiting for job 8123.euler to start
qsub: job 8123.euler ready

[aseidl@euler01 ~]$
```

[6]As opposed to CPU time which is the wall time multiplied by the number of CPUs used.

4.3.1 Available Compilers

As of January 2012, `gcc 4.4.6` is the only main compiler available on Euler. The `mpicc` and `nvcc` wrappers are available for compiling MPI and CUDA applications, respectively.

Listing 5: Compiling a Simple CUDA Application on Euler

```
[aseidl@euler01 ~]$ nvcc simpleExample.cu -o simpleExample
[aseidl@euler01 ~]$ ./simpleExample
Values stored in hostArray: 0, 1, 2, 3
```