A Prototype CPU-GPU Cluster for Research in High Performance Computing and Visualization of Large Scale Applications

Fritz McCall fmccall@umiacs.umd.edu
Brad Erdman chef@umiacs.umd.edu
Cluster Goals

• GPUs are increasingly programmable and very fast for vector operations and stream processing, but they present a challenging programming model; they can’t branch or loop.

• We want to couple them with traditional CPUs as high-performance co-processors and visualization engines for large-scale computation.
Cluster Overview
Node Overview

- Dell Precision 470 Chassis with Dual 3.0 Ghz Xeon EM64 Processors with 8GB of Main Memory
- Redhat Enterprise Linux Advanced Server
- 80 GB Root Disk and 160 GB Scratch Disk
- Ethernet and Infiniband
- NVIDIA 6800 Ultra GPU (nv45 series)
- NFS Home Directories (/vnodehomes)
Development Environment

• C/C++/Fortran Compilers
  • /usr/local/stow/gcc*, /opt/NAGWare*
• Java and Perl
  • /opt/jdk* and /usr/local/stow/perl*
• Debuggers
  • /opt/insure++, /usr/local/stow/valgrind*
Writing Shaders with Cg

- We currently support the NVIDIA Cg as a high-level language for GPU programming.

http://developer.nvidia.com/page/cg_main.html

/usr/bin/cgc

/usr/share/doc/Cg-1.3.0501

/opt/Cg/Examples
NVIDIA Cg (cont.)

- See these references for more details on GPU programming:
  

  The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics
  
  GPU Gems I and II
The NVIDIA SDK

• The NVIDIA SDK is installed in /opt/nvidia-SDK; it includes sample programs, effects, and libraries to demonstrate the capabilities of the NVIDIA GPUs.

NVIDIA Cg and SDK

Usage

• Use NVIDIA Cg and SDK to develop new low-level “shader” programs to run directly on the GPUs, especially where those programs concentrate on visualization or graphics.

• It could certainly be used to develop general purpose programs as well.
BrookGPU

- BrookGPU is a compiler and runtime for the “Brook stream program language”, a new language that extends C to enable general purpose programming on modern graphics processors.

  http://graphics.stanford.edu/projects/brookgpu/

- It is installed in /opt/brookgpu-0.4.
BrookeGPU Usage

- Consider using BrookGPU as a toolkit for general purpose computing on the GPU.
- It is intended to encourage a new general computational model called streaming which encourages “data parallelism” (i.e. doing the same thing in parallel on different data) and “arithmetic intensity” (i.e. minimizing global communication and maximizing localized computation).
Chromium

- Chromium is a derivative of wiregl; it is middleware for processing OpenGL streams in a distributed environment.

  http://chromium.sourceforge.net/

Chromium (cont.)

- More specifically, Chromium can do the following things:

  1. Intercept an opengl command stream from an unmodified application.
  2. Distribute the command stream across multiple nodes in the cluster (sort-first, sort-last, or custom sorting).
  3. Process the steams using one or more stream processing units on each node.
Chromium (cont.)

• As a developer, you can do lots of things with it:
  1. Write your own SPUs for custom processing
  2. Determine how the OpenGL command stream is distributed across the cluster
  3. Customize how the nodes apply the SPUs
Chromium Usage

• Chromium is useful if you are interested in distributed rendering or would like to try parallel stream processing between the graphic processors on different nodes.

• You want flexibility in applying a number of stream processing units on a single host.
• We currently support an MPICH implementation that is optimized for Infiniband. It is installed under /opt/topspin/mpi/mpich/

http://www-unix.mcs.anl.gov/mpi/mpich/

Using MPI: Portable Parallel Programming with the Message-Passing Interface
Logging in

• Login to vnodesub00 using ssh.
• Use your existing cluster account or Openlab password.
• You may transfer files using scp or sftp.
• Your home directory is mounted uniformly on all cluster nodes as /vnodehomes
Set up your identity

- Setup keys for single sign-on across the cluster:

  cd $HOME
  ssh-keygen -t rsa1 -N "" -f $HOME/.ssh/identity
  ssh-keygen -t rsa -N "" -f $HOME/.ssh/id_rsa
  ssh-keygen -t dsa -N "" -f $HOME/.ssh/id_dsa
  cd .ssh
  touch authorized_keys authorized_keys2
  cat identity.pub >> authorized_keys
  cat id_rsa.pub id_dsa.pub >> authorized_keys2
  chmod 640 authorized_keys authorized_keys2
Setup your path

- In your .cshrc, you may want to include the following in your path:

  # Use Chromium
  set path = ( /opt/cr-1.7/bin/Linux $path )
  # Use Brook
  set path = ( /opt/brookegpu-0.4/bin $path )
  # Use MPI
  set path = ( /opt/topspin/mpi/mpich/bin $path )
Please refrain from any serious computation on vnode_sub00. Instead, access the compute nodes as follows:

- For interactive jobs: `qsub -l`
- For batch jobs: `qsub batchscript.sh`
Accessing the nodes

- The scheduler is the same as all our other schedulers. More details about submitting jobs are online:

  http://www.umiacs.umd.edu/research/parallel/classguide.htm


- Please see the sections about PBS usage.
Running on the nodes

- You will interact with the GPU through the root Xserver by setting your DISPLAY environment variable to ‘localhost:0’. In tcsh, `setenv DISPLAY localhost:0`

- Oddly enough, you won’t see the graphical output since it will display to the root display! However, you will be able to run programs on the GPU using Cg, BrookGPU, and Chromium.
Running a sample BrookGPU program

- You can run a sample BrookGPU program as follows:

```bash
# allocate a compute node
[fmccall@vnodesub00 ~]$ qsub -l
qsub: waiting for job 67.vnodescheduler.umiacs.umd.edu to start
qsub: job 67.vnodescheduler.umiacs.umd.edu ready
localhost being added to access control list

# set the display
[fmccall@vnode06 ~]$ setenv DISPLAY localhost:0

# set the BRT_RUNTIME to opengl
[fmccall@vnode06 ~]$ setenv BRT_RUNTIME ogl

# apply an fft to a ppm file
[fmccall@vnode06 ~]$ /opt/brookgpu-0.4/bin/fft /opt/brookgpu-0.4/prog/apps/fft/franklin.ppm /tmp/out2.ppm
```
Running a simple Chromium Job

• You can run a sample chromium job as follows:

```
[fmccall@vnodesub00 ~]$ qsub -l
qsub: waiting for job 68.vnodescheduler.umiacs.umd.edu to start
qsub: job 68.vnodescheduler.umiacs.umd.edu ready
localhost being added to access control list

[fmccall@vnode06 ~]$ set path = ( /opt/cr-1.7/bin/Linux $path ); rehash
[fmccall@vnode06 ~]$ cd /opt/cr-1.7/mothership/configs/
[fmccall@vnode06 configs]$ python crdemo.conf fonttest &
This is Chromium, Version 1.7
Start a crappfaker on vnode06.umiacs.umd.edu
Start a crserver on vnode06.umiacs.umd.edu

[fmccall@vnode06 configs]$ crappfaker
[fmccall@vnode06 configs]$ crserver
```
Community

Many sites are working on these problems:

http://www.gpgpu.org/

http://spire.stanford.edu/

http://www.cs.sunysb.edu/~vislab/projects/cluster/

http://www.llnl.gov/str/November04/Louis.html
Coming Soon

• A 4-8TB parallel file system for high-speed data access to large data collections.

• Remote access to the compute node “screens”. So you can see what you are doing.

• More nodes with new features, like SLI.