GPUClusterUsage

back

Contents

GPU-Cluster:
Latest Info,
Problems,
Change Log
README

Link to slides of user training "Introduction to the the GPU-Cluster" (a bit outdated)

Overview

The Center for Computing and Communication is one of the supporters of the proposal for **innovative commodity computer architectures** for scientific applications. In this context, the GPU-cluster has been installed in July 2011. Because of its innovative character, this cluster does not run in real production stage, nevertheless, it will be tried to keep it as stable and reliable as possible.

Hardware

Our GPU-cluster comprises 28 nodes with each two GPUs, and one head node with one GPU. In detail, there are **57 NVIDIA Quadro 6000** GPUs, i.e. NVIDIA's Fermi architecture. For more details about the hardware refer to the Installation page.

- 4 dialogue nodes: linuxgpud[1-4]
- 24 render nodes
- 1 head node

All GPUs are in the compute mode "**exclusive process**" which means that the user who runs a GPU program gets the whole GPU and do not have to compete with other users for resources (e.g. GPU memory). Furthermore, it enables many threads in one process to use both GPUs in one node (cf. e.g. cudaSetDevice()), instead of being restricted to one thread per device.

Operation modes

The 24 of render nodes (+ head node) are used on weekdays (in the daytime) for interactive visualizations by the Virtual Reality Group of the Center for Computing and Communication and are NOT available for GPGPU computations in this period. However, during the nighttime and on weekends, they can be used for GPU compute batch jobs. Furthermore, one of the dialogue nodes runs in batch mode the whole day with the purpose to execute small batch jobs (e.g. for testing). The other three dialogue nodes enable an interactive access to GPU hardware. Here, the GPU compute batch jobs can be prepared and GPU applications can be tested and debugged. One of the dialogue nodes stays in interactive mode the whole day. The

others switch to batch mode in the evening. Between every switch of mode, a system reboot is done.

- **Interactive Mode**
  dialogue nodes
  linuxgpud1: whole day
  linuxgpud[2-3]: working days: 8 am - 8 pm
- **Batch Mode**
  all nodes (excluding linuxgpud1 and damaged hardware, cf. "latest info" page)
  linuxgpud4: whole day (short test runs only!)
  linuxgpud[2-3], linuxgpus[01-24], linuxgpum1: working days: 8 pm - 8 am; weekends: whole day

Annotation: There might be some changes in the distribution of interactive and batch mode availability and times in the future.

Access

You can get access to the GPU-Cluster by using your TIM user account (you must have enabled "HPC" in TIM previously). Moreover, you must be a member of the "gpu" group. Be aware that the distribution of group rights may take one day. If it takes longer or you have any other problems, write an email to our service desk (servicedesk@rz.rwth-aachen.de).

Friendly usage

Since our GPUs can only be used from one person at the same time (and in batch mode, even the 2 GPUs of one node can be used from one person at the same time), please use them reasonably. That means, run big jobs only in batch mode and close the debugger after usage. We also appritiate compute jobs that allow other users to run their jobs once in a while. Thank you!

Interactive

You can use our GPUs interactively by using SSH. If wanted, you can first log into one of our X-frontends (*cluster-x* oder *cluster-x2*) and then use SSH to log into one of the GPU dialgue nodes. Compare section "Operation Modes" for the time period for interactive access and "Latest Info" for details about broken machines.

As we set our GPU to "exclusive" mode (see Overview-Hardware), you get an error message if you want to run your program on the same GPU device as another user already does. If possible, just choose another device number for executing your program.
For CUDA applications, don't set a certain device explicitly in your program (cudaSetDevice) if not needed. Then, your program should be automatically scheduled on a free GPU device (if there is one). If you set the device number, you have to wait until this certain device is available again (and try it again).
Be aware, that debug sessions alwalys run on device 0 (default) and that you might see the same probelm there.

If you want to test your **GPU + MPI** program interactively, you can also do that on the dialogue nodes (cf. "Latest Info" for damaged GPUs) using our mpiexec-wrapper `$MPIEXEC`. To get your MPI program run on the GPU machines, you have to specify them by hostname (otherwise you will get scheduled on our MPI backend which does not have any GPUs). You can do this by option:

```
-H host1,host2,host3
```

However, you should *not* use this option on our normal MPI machines! You can also denote how many processes shall run on GIVEN hosts (see example below or `$MPIEXEC -help`).
Furthermore, it can be useful to specify how many processes shall run on EACH machine (e.g. if each of your processes uses one GPU). For our interactive mpiexec-wrapper use the `-m` option, where ppn defines the processes per node:

```
-m ppn
```

Examples for the usage:

```
$MPIEXEC -np 4 -m 2 -H linuxgpud1,linuxgpud2 foobar.exe // 2 processes on each machine
$MPIEXEC -np 4 -H linuxgpud1:1,linuxgpud2:3 foobar.exe // 1 process on first machine, 3 processes on second
```

Batch

You can submit batch jobs for the GPU-Cluster the whole time. However, dependent on the batch mode availability, scheduling and occupancy, it might take a while until your compute job has finished (see batch times in section "Operation Modes").
We use LSF as batch scheduler (just as the new RWTH Compute Cluster). Information about the general usage of LSF, can be found in the corresponding Wiki (Manuals => Usage of the Cluster => Workload Management System LSF). As LSF is only used for our new hardware at the moment, you have to submit your batch jobs from one of the GPU dialogue nodes or from nodes of the new Compute Cluster.

In the following, we have to differ between short test runs on the one hand and real program runs on the other hand.

Short test runs can be done on the single machine `linuxgpud4` (comprising 2 GPUs) which is in batch mode also during daytime. If you want to test your batch script, add:

```
-a gpu
```

If you also want to use MPI (be aware that there is only one system with 12 cores and 2 GPUs), you can combine the request for GPU- and MPI-usage (please see below for more details on the usage of MPI):

```
-a "gpu openmpi"
```

Real jobs (i.e. longer runs or performance tests) have to be scheduled on the GPU-cluster (nighttime and weekends). Therefore, you have to select the appropiate **queue**:

```
-q gpu
```

You can add it either to your batch script file or to your bsub command. You will get all requested machines exclusive.
BTW: If you add accidently "`-a gpu`" AND "`-q gpu`", the "`-q gpu`" option will be taken and your job will run at nighttime on the GPU-cluster.

An small **example** batch script file can be found at the end of this page (for download). For submitting this example use:

```
bsub < gpuDeviceQueryLsf.txt
```

There, you can also download an example LSF file for an **GPU-MPI application** (using OpenMPI). If you want to use another MPI or a job over several nodes but without MPI communication, please compare the LSF-Wiki (link above). Since we do not have a requestable GPU slot in LSF at the moment, you have to specify how many processes you want to have per node (see `ptile` in script file). This is usually one or two, depending how you want to use the GPUs and how many GPUs you want to use per node.

- 1 process per node (ppn)
  - If you want to use only one GPU per node.
  - If your process uses both GPUs in one node, e.g. via cudaSetDevice.
- 2 processes per node
  - If each process communicates to one GPU of the node.
- More
  - If you have processes which do only computation on the CPU. Be aware that our GPUs are set to "exclusive process" mode which is the reason that not more than one process can use one GPU.

Please note that in batch mode all (correct working) GPU machines are available, even one single machine with only ONE GPU attached (linuxgpum1). If you have to exclude this particular machine from your host-list (e.g. due to the fact that you want to use 2 ppn and each uses one GPU), please denote additionally the following (so that you will get only machines with two GPUs):

```
-m bull-gpu-om
```

Since we have trouble with a couple of machines (see GPU-Cluster's Latest Info), we excluded these machines from the available machines for batch jobs.

Configuration details

X-Configuration or *Where to use a CUDA debugger*?

Due to the different needs of VR and HPC, we have different X-Configurations on different nodes. These may change in future.

- linuxgpud[1-2]: X session runs on display of GPU 1. On GPU 0 is the display mode disabled.
  => Here you can use cuda-gdb or TotalView to debug your GPU application. However, you might still have porblems with CUDA debugging. We are working on it.
- linuxgpud3: X sessions run on both GPUs.
- linuxgpud4: Since this GPU runs in batch mode, the display mode is disabled on both GPUs.

Usage of Programming Models

CUDA

NVIDIA developed CUDA C for programming their GPUs. PGI added a CUDA Fortran version, also for NVIDIA GPUs.
On the GPU-Cluster, we recommend to use the **CUDA toolkit Version 5.0**. We also provide the toolkit version 4.1, 4.0 and 3.2.

Usage information can be found here.

OpenCL

OpenCL is a open standard to program GPUs/CPUs (NVIDIA, AMD, Intel,...) and other device using a C-like API.
Usage information can be found here.

PGI Accelerator

The Portland Group developed the PGI Accelerator Programming Model which is a directives-based approach (similar to OpenMP) and runs on NVIDIA GPUs (so far).
Usage information can be found here.

NVIDIA GPU Computing SDK

NVIDIA provides numerous examples in CUDA C and OpenCL C.
How to get them and to use them, see here.

Syntax Highlighting

Here you can find more information about how to enable syntax highlighting for CUDA and OpenCL files. There is an overview about:

- Emacs

Files/ Links

- Example batch script files
  - Simple: Runs deviceQuery (NVIDIA SDK) on one device: gpuDeviceQueryLsf.txt
  - MPI: Runs deviceQuery (no real MPI application!) on 4 nodes (each one device): gpuMPIExampleLsf.txt

Last modified at 5/29/2013 12:50 PM by Wienke, Sandra