Notes on CUDA practicals on Arcus-B cluster

Prof. Mike Giles

1 The Arcus-B cluster

The layout of the K80 nodes in the Arcus-B cluster is shown in Figure 1.

The head node which sits on the external network is `arcus-b.arc.ox.ac.uk`. This can be accessed from machines on the university network through an SSH command:

```
ssh -X username@arcus-b.arc.ox.ac.uk
```

Note that your username on Arcus-B will be different to your username on the home system in the Thom building. The `-X` option provides X-window forwarding which is convenient when using windows-based editors such as `emacs`, or the Nsight IDE.

The 5 K80 GPU nodes on Arcus-B each have 2 K80 GPU cards, but each card has 2 GPUs, so there are a total of 4 GPUs on each node, numbered 0 – 3. It's also possible we may use some other compute nodes with K40 GPUs.

2 CUDA 9.0.176

Your account should be pre-configured to use CUDA version 9.0.176, the NVIDIA compiler `nvcc` and the various CUDA libraries. The SDK (software development kit) is located at

```
/system/software/arcus-b/gpu/cuda/9.0.176/
```

Its `samples` subdirectory has lots of useful example codes, the `include` has various header files, and `lib` has the libraries.

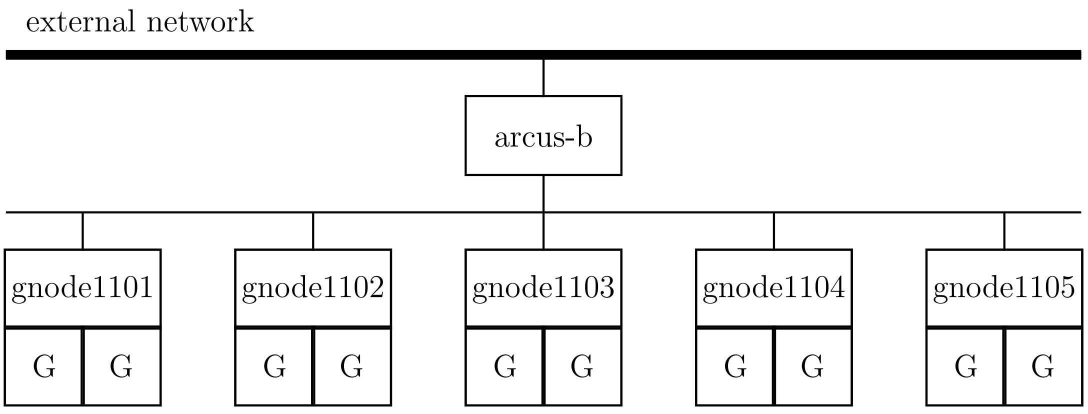

Figure 1: Arcus-B cluster with head node and K80 compute nodes

3 Editing, file transfer and printing

`emacs`, `vi` and `gedit` are all available on `arcus-b`. Windows users who are unused to Linux editors may prefer `gedit` which can be launched in its own window using the command:

```
gedit &
```

Using `gedit` you can open and save files by clicking on icons, so it's very easy to use, though it doesn't have the advanced features of `emacs`.

Files should be transferred using `scp` to your home system in the Thom building for local printing.

4 Interactive sessions on the compute nodes

When logged into the Arcus-B head node, you can create an interactive session on one of the K80 compute nodes by issuing the following command:

```
salloc -pgpu --ntasks-per-node=1 srun --pty --x11 --preserve-env /bin/bash -l
```

and once you are then put onto one of the K80 nodes, issue the command

```
export CUDA_VISIBLE_DEVICES=0,1,2,3
```

You can then edit and compile your code as usual, and run it on one of the GPUs attached to the compute node. Using an interactive session like this can be appropriate when developing codes using test runs which last no more than a few seconds, but for anyone doing "production" runs you should always use the batch scheduler.

5 Makefiles or Nsight

You have a choice in how you work on the practicals.

One option is to use a standard editor to edit the files, and the supplied Makefiles which specify which files are compiled and linked to create an executable which you can then run at the command line. A tar file for all of the practicals can be copied across from `~mgiles/practicals.tar.gz`.

The other option is to use Nsight, an NVIDIA IDE (Integrated Development Environment) based on Eclipse; this combines an editor, debugger, profiler, etc, into one big package, a bit like Visual Studio for Windows. If you prefer this option, then again things are pre-packaged for you to import into Nsight.

To launch Nsight you use the command:

```
nsight &
```

and then to load in all of the practical "projects", select File/Import. This gives a pop-up window, where under the General category, you select Existing projects into Workspace, and in the next step select "from archive file" and browse for the `practicals_nsight.tar.gz` tar file which can be copied across from `~mgiles/practicals_nsight.tar.gz`.

For documentation on Nsight, see https://developer.nvidia.com/nsight-eclipse-edition