GPU/Computation Server Setup and Use

Alun Jones and Dave Price

12th July 2021

1 Introduction

We operate a server, currently with two powerful GPU cards, for use with deep learning and similar projects. The server also has some non-GPU computation nodes.

This document gives some basic details of how to submit jobs to it and describes the setup of the server.

2 The System

So as to closely follow the arrangement provided by SCW (Super Computing Wales) we are using the SLURM system to let people queue work for processing by the GPU or non-GPU computation nodes.

Access to the facility is currently provided via five virtual hosts.

- `slurm.dcs.aber.ac.uk`: This is the machine you log into to create and submit jobs.
- `gpu1.dcs.aber.ac.uk`: This is the current GPU compute engine. It's the only virtual server that can "see" the GPU cards. You **do not** have direct login access to this machine, but you can submit jobs to run on it using Slurm commands issued on `slurm.dcs.aber.ac.uk`.

  `gpu1.dcs.aber.ac.uk` and `slurm.dcs.aber.ac.uk` have almost identical software sets installed on them. So you can develop and test code on `slurm.dcs.aber.ac.uk` (without access to GPUs) then submit it, via Slurm, onto `gpu1.dcs.aber.ac.uk` where, all being well, it should be able to run on one or both GPUs.
- `cpu1.dcs.aber.ac.uk and cpu2.dcs.aber.ac.uk`: These are the two current non-GPU nodes. These nodes are machines that were recovered from an older cluster. They do *not* have GPUs, but they do have 128GBytes of memory and 16 Xeon cores each at 2.6GHz and are therefore suitable for running jobs which require lots of memory (but no GPU). You **do not** have direct login access to these machines, but you can submit jobs to run on them using Slurm commands issued on `slurm.dcs.aber.ac.uk`.
- `slurmsrv.dcs.aber.ac.uk`: This runs no user code at all. It's the server that's responsible for taking job requests from users and sharing them out onto compute resources.

Access to the system is restricted. In order to gain access, you need to contact our support team to request an account (email cs-support@aber.ac.uk). Once created, your account will use your usual Aberystwyth username and password.

Authorised users are provided with a "home directory" which is stored on the local disks of our GPU server.

N.B. While this filestore is on RAID disk, and should thus be relatively safe against disk failure, we perform NO backups of the filestore. You should consider it as scratch space for running jobs and keep copies of any important content elsewhere.

3 Using the System

Job submission is done via the Slurm batch management system, and you will need to be logged into `slurm.dcs.aber.ac.uk` (using SSH/Putty) to launch jobs.

Lots of information about Slurm is available at `https://slurm.schedmd.com/tutorials.html`.

3.1 Cheat sheet

Command (enter as one line)	**Description**
`sinfo`	Show basic information about the nodes known to Slurm.
`squeue`	List the current Slurm job queue.
`srun [--gres gpu:1] command`	Run `command` on the GPU node, waiting for it to finish. `--gres gpu:1` tells it that your code needs access to one GPU. If you have written your code to access more than one GPU, you can use `--gres gpu:2` to request access to both. If your code uses a GPU and you don't request access to one, your code will not work (or, depending on the framework you're using, might choose to do everything very slowly on the CPU). **N.B.** Do not request both GPUs if your code only uses one. That would lock out anyone from using the other GPU at the same time as you.
`sbatch [--gres gpu:1] scriptname`	Add `scriptname` on the job queue, exiting immediately. The job's output will be written to `slurm-NNN.out` where NNN is the job number that was assigned. `scriptname` must refer to a shellscript starting with "`#!`". See notes for `srun` above regarding GPU access. **N.B.** anaconda3-launch (described below) is one such shellscript that could be started by using `sbatch`.
`anaconda3-launch [-env envname] command`	Run a command with its environment configured to find the install of Anaconda 3. If `-env envname` is specified then use the environment named `envname`. For example: `$ anaconda3-launch -env tensorflow-gpu-1.12 python something.py`
`anaconda3-launch conda env list`	List installed anaconda environments.
`sbatch --gres gpu:1 anaconda3-launch -env tensorflow-gpu-1.12 python my-tensorflow-code.py`	Putting it all together - submit a job to the GPU server to run something within the Anaconda tensorflow-gpu-1.12 environment.
`salloc`	Please **do not** attempt to use the salloc command. Its use can potentially grab resources and thus prevent others from using our shared GPU resources even when you are not actively using them.

NOTE: To run commands on the CPU (rather than GPU) nodes, you should add `-p cpubig` to your `srun` or `sbatch` commands. You do not need to reserve resources on these nodes with the `--gres` option.

3.2 Moving data and programs to/from your home directory on our GPU server

There are several ways you can transfer your programs and your data to/from our GPU server.

- `scp`

  For those of you familiar with the `scp` command you could issue a command such as

  `scp` *my-local-file* *username*`@slurm.dcs.aber.ac.uk:`*my-file-on-gpu-server*

  and supply your password when requested.

- `map a network drive`

  We expect many users will choose this approach.

  Your home directory on the GPU server is available for access from other computers within our network as a **network drive**.

  - You could thus attach to this from a Microsoft Windows computer by using the "`Map network drive`" menu option and requesting to connect to:

    `\\slurm.dcs.aber.ac.uk\`*username*

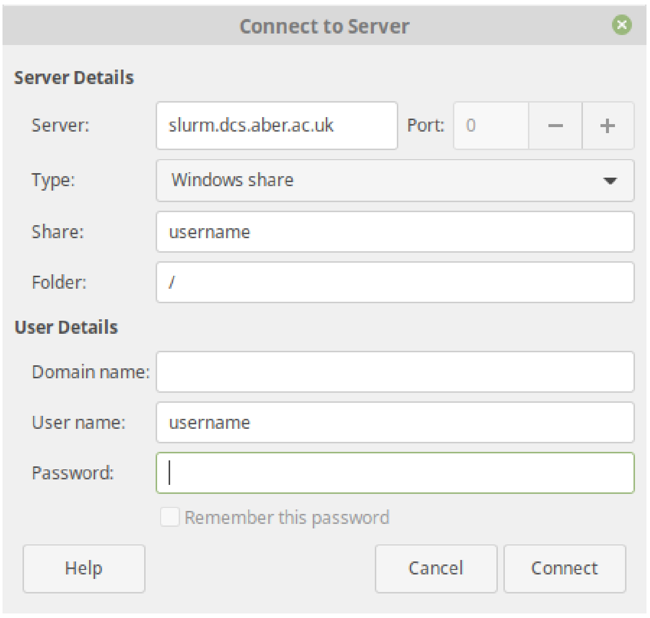
  - If you are working on Apple's MacOS then in `Finder` select the `Go` menu and then choose `Connect to Server` and specify

    `smb://slurm.dcs.aber.ac.uk/`*username*

    or

    `cifs://slurm.dcs.aber.ac.uk/`*username*

  - If you are working on a Linux computer as a non-root user then you could start the File Manager application (Files), go to `File -> Connect to Server` and then complete the pop up window as shown in the following diagram, replacing `username` by your username and by entering your password in the Password box. Filestore connected this way can only be accessed inside the file manager.



Connect to Server

Server Details

Server: slurm.dcs.aber.ac.uk Port: 0

Type: Windows share

Share: username

Folder: /

User Details

Domain name:

User name: username

Password:

  - If you are working on a Linux computer where you have root permission, you could issue the command line instruction shown below in a shell terminal window.

    `sudo mount -t cifs -o username=`*username* `//slurm.dcs.aber.ac.uk/`*username* `/mnt`

- `sshfs`

  This could prove useful for many users and allows non-root users to properly mount remote filestore.

  If you are working on a Linux computer you could issue the command line instructions shown below in a shell terminal window. This might prove useful for anyone working on our Linux Mint computers for instance. You only need the `mkdir` command the first time you do this.

  ```
  mkdir my-slurm-home
  ```

  ```
  sshfs username@slurm.dcs.aber.ac.uk:  my-slurm-home
  ```

4 Software installed

In addition to the Slurm client tools, we have installed a typical, representative, set of deep learning resources. If software is missing, we can discuss your needs and come to a decision as to whether it's sensible for us to install it centrally or for you to maintain your own install.

4.1 CUDA

This is nVidia's compiler and driver suite. If you are developing programs written in C/C++ for direct access to the GPU, this is what you need and, presumably, you'll know what to do! The software is installed under /usr/local/cuda.

4.2 Anaconda 3

We expect that most people will be using deep learning frameworks written in Python. Since there are various Python-based frameworks and we suspect that people will need to be able to use a variety of versions of those, we have taken the approach of installing a basic Anaconda 3 package onto the system, then adding Anaconda environments for a few frameworks. Subject to CUDA driver version constraints, we can look at installing extra environments centrally, or you can create your own and compile your own packages.

4.2.1 Environments

You can get a list of installed Anaconda environments using the following command:

```
$ anaconda3-launch conda env list
```

```
auj@slurm:~$ anaconda3-launch conda env list
# conda environments:
#
base                  *  /opt/anaconda3
caffe-cpu-1.0            /opt/anaconda3-envs/caffe-cpu-1.0
caffe-gpu-1.0            /opt/anaconda3-envs/caffe-gpu-1.0
tensorflow-2.3.0         /opt/anaconda3-envs/tensorflow-2.3.0
tensorflow-cpu-1.12      /opt/anaconda3-envs/tensorflow-cpu-1.12
tensorflow-cpu-2.0.0     /opt/anaconda3-envs/tensorflow-cpu-2.0.0
tensorflow-gpu-1.12      /opt/anaconda3-envs/tensorflow-gpu-1.12
tensorflow-gpu-2.0.0     /opt/anaconda3-envs/tensorflow-gpu-2.0.0
theano-1.0               /opt/anaconda3-envs/theano-1
```

5 The Server Setup

This section is largely for interest only - it describes the basic setup of the system and how we're managing software and hardware access.

The machine itself has 48GBytes of memory, 7TBytes of disk, 32 CPU cores and two nVidia GP100GL GPU cards, each with 16GBytes of onboard memory.

The server is running Ubuntu 18.04, since Ubuntu seems to be the Linux distribution which has the best support for the nVidia compiler suite (CUDA).

Since this system is going to be used by multiple people and we currently have (only) two GPUs, we needed some method for mediating access. We decided that it would be a good idea to use the same scheduling software that is used in Super Computing Wales.

So we have installed "Slurm" (https://slurm.schedmd.com/overview.html) on the system.

Slurm is suitable for managing large clusters of machines, whereas we currently only have one. The work involved in configuring a Slurm service with one compute engine is not significantly less than that of configuring it to support many compute engines. So we have tried to set up the system to be easily expanded to support more, should any future machines get added. We therefore "containerised" the service. We have three "thin" virtual servers running on the physical server we've bought:

- `gpu1.dcs.aber.ac.uk`: This is the main compute engine. It's the only virtual server that can "see" the GPU cards. You don't have direct login access to this machine, but can submit jobs onto it using Slurm.
- `cpu1.dcs.aber.ac.uk and cpu2.dcs.aber.ac.uk`: These are the two current non-GPU nodes. They do *not* have GPUs, but they do have 128GBytes of memory and 16 Xeon cores each at 2.6GHz and are therefore suitable for running jobs which require lots of memory (but no GPU). You **do not** have direct login access to these machines, but can submit jobs onto them using Slurm.
- `slurm.dcs.aber.ac.uk`: This is the machine you log into to submit jobs. `gpu1.dcs.aber.ac.uk` and `slurm.dcs.aber.ac.uk` have almost identical software sets installed onto them. So you can develop and test code on slurm.dcs.aber.ac.uk (without access to GPUs) then submit it, via Slurm, onto `gpu1.dcs.aber.ac.uk` where, all being well, it should be able to run on one or both GPUs.
- `slurmsrv.dcs.aber.ac.uk`: This runs no user code at all. It's the server that's responsible for taking job requests from users and sharing them out onto compute resources.

We've carved off 4TBytes of the 7TByte disk array and made it into /home, the location where your home directory will live on this server. The filestore is available on both `slurm.dcs.aber.ac.uk` and `gpu1.dcs.aber.ac.uk` so you can run jobs that are stored within, and refer to files within, your home directory.

N.B. While this filestore is on RAID disk, so should be relatively safe against disk failure, we perform NO backups of the filestore. You should consider it as scratch space for running jobs and keep copies of any important content elsewhere.

The idea behind this structure is that it is easy to keep `gpu1.dcs.aber.ac.uk` and `slurm.dcs.aber.ac.uk` in sync as regards their installed software, and it should be easy to replicate this install onto other GPU-enabled machines, should any be added in future. In that case, you would continue to develop and submit jobs on `slurm.dcs.aber.ac.uk` and they would be seamlessly deployed onto those new machines as needed.