D. Pogosov

SurfSARA

they have RedHat linux

1. Connection: Files transfer via WinSCP or Filezilla
   * With a whitelisted IP: Windows: In Putty fill in under “Host Name (or IP address) **cartesius.surfsara.nl** From home, establish a VPN: http://www.ru.nl/isc/studenten/externe-toegang/vpn/opzetten-vpn/
   * Without a whitelisted IP: Windows: In Putty fill in under “Host Name (or IP address) “doornode.surfsara.nl” after that select “Cartesius”. Quick answer is, you can't copy files through the doornode.

2. Forwarding: If you want to forward ports from the remote (compute node) locally to your machine (will forward port 5000 of gcnX to local port 80; X - node number):
   ssh -L 80:gcnX:5000 gdemo006@cartesius.surfsara.nl
   ssh gcnX
   one node has 2 Tesla K40, more info:
   nvidia-smi

3. Software: have CUDA loaded (and potentially cudnn if you need it):
   module av cuda
   Then you’ll get a list of options and you can load one of the drivers by (for example):
   module load cuda/8.0.44
   Similarly to CUDA, you can do a:
   module av cudnn
   Then you’ll get a list of options and you can load one of the drivers by (for example):
   module load cudnn/8.0-v5.1
   I assume you might want to use also one of our python versions that has more of the ML packages installed. Therefore I recommend using:
   module load python/2.7.11
   In order to get the right information, after you load all your modules, do a:
   module show cuda
   This will show you all the paths and variables set by the currently loaded cuda module. Loading a specific desired CUDA module first is important as otherwise, you will be given the paths of the default CUDA which might not be what you want.

4. Preparing Yolo2: Compile Darknet (make). Makefile needs changing: delete a key which sets compilation optimization; change hard path to cuda to **$(CUDA_HOME)**

5. Hardware: have a GPU available. So, you can use one of the gcn* nodes. You can see the queue for them with:
   squeue -p gpu
   Nodes gcn1 and gcn2 are available for users interactively for precisely this reason: having a GPU enabled environment to prototype. Bear in mind that these (free nodes) are shared amongst all users, therefore they might be quite busy (lots of users running at the same time). Nodes gcn1 and gcn2 are busy because they are free of accounting, therefore everyone wants to run/compile things on them (using

them for actual calculations - long processes is actually frowned upon and might lead to warnings/sanctions). Alternatively, if they are really busy and you can spare the budget (make sure you can still do the workload from your course), you can also use compute nodes for compiling. In the doc I sent to you there are examples of building a batch job https://userinfo.surfsara.nl/systems/cartesius/usage/batch-usage.
If a batch job seems like too much hassle, you can simply do:
salloc -N 1 -p gpu -t 01:00:00
This command will allocate one GPU node for 1 hour (48 SBUs out of your allocated 2k). Then to see what node was allocated you can do a:
squeue -u gdemo006
then in the dump you'll see something like

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST
294256	gpu	bash	gdemo006	R	0:04	1	gcn8

so this says: job '294256' is running on the 'gpu' partition, the script file 'bash', for user 'gdemo006' and has the state "R" (running) and started "0:04" seconds ago on node "gcn8". so if this was your case, you should now ssh to that machine - in this case gcn8 - which is all yours (not shared with other users anymore) and you can do your work.
Oh, if you go for the salloc option, don't forget to kill your job when you're done, like:
scancel 294256
where 294256 is the job id that was granted with salloc.
Warning: What I meant by the budget thing was that we are billing by used time on compute nodes, so if you minimize the time you spend on compute nodes than you are being efficient. So, you would want to use the batch system (using sbatch), not necessarily the interactive one(salloc). So if we take a simple example: I have a piece of sw that runs for 2 minutes and then gives an error. What happens in the two systems? Batch: you get billed for 2 minutes. Interactive: you get billed for the 2 minutes + the time that it takes you to actually close the job yourself (so if you imagine you started a compute run and then stepped away from the keyboard, you might end up spending a lot of your budget pointlessly).

6. How to watch progress: If you submitted a slurm job (with sbatch) then probably you have some files that contain the word "**slurm**" and a job id from the folder where you launched the jobs.
   You can always redirect your outputs to a file with something like:
   python test.py > debug.log 2>&1

Speedy
(only for Classifying Nemo members)

I made an account on my gpu server for you:

Server: **speedy.cs.ru.nl**
Username: **mlip2017**

I use (from the wired network in the Mercator building):

ssh mlip2017@speedy.cs.ru.nl
and from other (science vpn, not for ubuntu 16.04) networks:
ssh -L 9022:speedy.cs.ru.nl:22 [your name]@lilo.science.ru.nl -Nf
ssh mlip2017@localhost -p9022
I use the linux command line ssh client (openssh). On windows the mingw ssh client also works. For transferring files, I use rsync or sometimes scp.

How to login:

1. In this case, you can connect to the server **lilo.science.ru.nl** directly, even without the VPN connection. You also have to use your Science credentials for this.
2. Then, from **lilo**, you should be able to connect to **speedy.cs.ru.nl** .
3. **tmux**
   nvidia-smi
4. Use only the second GPU (1)

Issues:

1. for Make
   in Makefile: $(CUDA_HOME) -> **/usr/local/cuda-8.0/**
2. **export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-8.0/lib64**
   export PATH=$PATH:/usr/local/cuda-8.0/bin
3. Out of memory: increase subdivisions, e.g. from 8 to 16
4. use **nvidia-smi -l 1** will continually give you the gpu usage info, with in refresh interval of 1 second.
5. Disk space, free **df -h .** , occupancy by current folder **du -hs**
6. to mask GPU0:
   export CUDA_VISIBLE_DEVICES="1"
7. **unset CUDA_VISIBLE_DEVICES**