Connecting to the NIPG38 server

Szalontai Balázs

December 2025

Our department shares a server with the AI department, which we can use to train and evaluate deep learning-based models, such as LLMs. This server has eight A100 40GB GPUs. For reference, one of these is enough to load in a ∼16B model in bf16 preision and use for inferene; or to train a ∼4B model in full parameter fine-tuning. We cannot use all of these GPUs, just 2 of them. In this document, you'll read about how to connect to the server.

1 The config file

You have to move the attached *config* file to the *.ssh* folder, located in your home folder (on all operating systems as far as I know). You don't need to change anything in this file as you will log in with my credentials.

2 Generate an SSH key

Run the following command in your terminal:

```
ssh-keygen
```

You don't necessairly need to provide any passphrase, just feel free to hit enter until the process is done. This creates a .pub file in your *.ssh* folder. You have to send the entire content of this file to me, as I need to paste it to the *.authorized_keys* file on the server. Wait until this gets done.

3 Check if you can ssh into the server

Try connecting to the server. This is the most straightforward way to open a terminal in the serer.

```
ssh nipg38
```

This is the most straightforward way to open a terminal in the serer. The super-secret password is:

```
Asdfgh11_CodeRef
```

If you encounter any issues, let me know. If everything goes fine, you can exit for now with Ctrl+D.

4 Connect with port forwarding

Working with a single terminal is fine for some, but I usually edit files with Jupyter Notebook. If you're with me on this, you have to connect to the serer with port forwarding in the following way:

```
ssh -L abcd:localhost:abcd bszalontai@nipg38
```

Replace abcd with the port of your choice. It should be unique, so that it does not conflict with ports used by others. Yes, 8888, 1234, 6969 are taken. Be more creative!

5 Using the Apptainer image

Many packages are not available by default on the server. You also don't have sudo permissions to download anything. This is where Apptainer comes into play. It is similar to a docker image, containing the packages you need. You can use the following command to load the image:

```
apptainer run --nv --bind /ssd/bszalontai_local/ llama_YY_MM_DD.sif
```

Here, YY_MM_DD refers to the date when the image was created. I usually update images every now and then, so make sure that you're using the latest one.

You could also create your own image, but this has not often been necessary thus far, as everyone on the server does similar things and uses the same packages. Let me know, if you want to create your own image, and I'll help you with that.

6 Launch a Jupyter server

You can launch the Jupyter server with the following command:

```
CUDA_VISIBLE_DEVICES=X jupyter notebook --port=abcd
```

Replace abcd with the port you used for port forwarding. Replace X with the GPU ID you want to use throughout the session. For more info on GPU IDs, see Section 7.1.

Open the link in your browser. If a password is required, use the same password as above.

You should now see the home diretory of the server in your browser. Here, create your own working directory. This is where you'll do everything. Good luck!

7 Further notes

You're almost all set, but here's a bunch of other things...

7.1 Which GPU should you use?

There's no built-in mechanism to prevent you to use any of the GPUs on the server. That's why you need to be extra careful not to use a GPU that is not ours. Whenever you use the serer, I suggest opening a separate terminal (ssh nipg38), and logging GPU usage with the *nvidia-smi -l* command. Keep this terminal open to see if you are indeed using the GPU you want to.

You can specify the used GPU by setting the environment variable *CUDA_VISIBLE_DEVICES* to the chosen GPU ID. My suggestion is that whenever you launch a Jupyter server or run a Python script, do it with the *CUDA_VISIBLE_DEVICES=X* prefix. If you don't want to use a GPU, just use the *CUDA_VISIBLE_DEVICES=""* prefix.

Our group can use GPUs 2 and 3 in general. But we are a somewhat large group, so we keep track of our intentions and requests. The table is accessible here. Here, you can reserve a specific GPU. If you don't have acces to this sheet, let me know.

Be mindful about your reservations, as others are also using the same 2 GPUs. Only reserve a GPU if you are really using it. If you're not using it all day, include the specific interval of the reseravtion. Only reserve for multiple days if you are really planning on running something for that long.

7.2 Use the SSD for large files

The home directory is accessed through network, which is a huge bottleneck. Don't even think about uploading larger files here. Our quota is also limited to 500GB, which seems a lot, but we can run out of it fairly quickly.

Therefore, you should use the SSD for large files (/ssd/bszalontai_local/). Feel free to create your own folder, or to use the *models_hf* folder, where you can find a bunch of pretrained models from Huggingface.

7.3 Notebooks vs. Python files

Notebooks are great to develop and mess around with your models, whenever training takes more time, you probably don't want to use them anymore.

Once you are ready to run your long process (model training or evaluation), I suggest converting your notebook to a Python file. This can be done with Jupytext:

```
jupytext --to py your_notebook.ipynb
```

You can now run your notebook as a script:

```
CUDA_VISIBLE_DEVICES=X python3 your_notebook.py
```

7.4 Running a process in the background

Building on the previous point, you can run a process in the background by opening a screen session. This is useful, as otherwise losing the connection to the server will kill your valuable training process. I've been there, it feels bad.

Once you logged in to the server, don't just call apptainer as usual, but run it in a screen session:

```
screen -S screen_name apptainer run --nv --bind /ssd/bszalontai_local/
llama_YY_MM_DD.sif
```

Here, screen_name can be anything you want. Once you opened the session, you can let it do its thing by detaching from the screen with Ctrl+A+D. If you want to reattach to it, use the following command:

```
screen -r screen_name
```

You can list the currently running screen sessions with the following command:

```
screen -ls
```

If you lose the connection or exit an attached screen in an unorthodox way, you might not be able to reattach to it as it is shown as attached. In such a case, you can detach using:

```
screen -d screen_name
```

If you don't need the session anyore, close it with Ctrl+D.