Computer Lab Supplement

The following document serves as a handout for an introductory meeting to labs. It is not at all required to dive into the details, or even get to know the tools and commands. You might be perfectly fine doing computations on your personal computer with reasonably modern hardware, since the tasks typically do not require great computational powers.

However, if you'd like to take advantage of the resources available at our faculty, we encourage you to do so.

1 Hardware

We have configured the computers in Lab110 and Lab137.

	110		137
CPU	Intel i7-4930K @ 3.40GHz	Intel i5-4460 @ 3.20GHz	Intel i7-2600 @ 3.40GHz
#CPU cores	6	4	4
GPU	GTX 780 [1]	GTX 660	onboard
OS	Linux/Windows		Linux
IP	192.168.4.{70,...,88}		192.168.4.1{50,...,67,69}
Hostname	lab110-{01,...,10}	lab110-{11,...,20}	lab137-{01,...,19}

To check which computers are up, invoke

```
student@lab110-01:~$ sinfo
```

Note: `sinfo` is not configured to list machines `lab110-{11,..,19}`, though they are still configured with the rest of our software.

2 Software

Our software stack resides under `/pio/os/`. Each package has to be sourced separately with

```
student@lab110-01:~$ source /pio/os/<package-name>/set-env.sh
```

For example

```
student@lab110-01:~$ source /pio/os/anaconda/set-env.sh
```

will set Anaconda located at `/pio/os/anaconda` as the default source of Python binaries (such as `python` and `ipython`).

Please keep in mind that you need to source the appropriate `set-env.sh` scripts at each session separately. If you forget to set it up, you might still use some tools (eg. Python), but in old, local version lacking essential libraries. Before you begin your work, check Python version:

```
student@lab110-01:~$ python --version
```

It is a good practice to source commonly used software either in `~/.bashrc`, or use the `pio-shell.sh` script which we have provided on the course page. Further reading (not at all mandatory) *Bash guide for Beginners, 3.1. Shell initialization files*: `http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_03_01.html`.

Additional Python modules might be installed locally with:

```
student@lab110-01:~$ pip install --user <module_name>
```

[1] Some of the machines have older cards with no CUDA capabilities. Please check using `lspci` command.

2.1 NVIDIA CUDA

Likewise to use the GPGPU cards it is best to use the CUDA compilers from `/pio/os`:

```
student@lab110-01:~$ source /pio/os/cuda-8.0/set-env.sh
```

There are two specific versions of CUDA 8.0 libraries with different versions of the NVIDIA CUDA Deep Neural Network library (CUDNN). In order to load a specific version type either:

```
student@lab110-01:~$ source /pio/os/cuda-8.0_cudnn-5.1/set-env.sh
```

or

```
student@lab110-01:~$ source /pio/os/cuda-8.0_cudnn-6.0/set-env.sh
```

Different versions of deep learning libraries tend to use different versions of CUDNN.

3 File Storage

You can store temp files on two larger network drives, `/pio/os/scratch/1` and `/pio/os/scratch/2`. They are available in Lab110 and in Lab137. Please create a folder with you username in one of the scratch spaces. Since there is no quota, please use as little as you need and don't force us to establish a quota system. We will clean the scratch space after the semester.

4 Networking

A direct connect might be set up with a chained ssh tunnel:

```
student@personal-pc:~$ ssh -t xxx@tryglaw.ii.uni.wroc.pl \
>                      ssh -t hera ssh -t lab110-01.stud.ii
```

Local port might be forwarded to one of the lab machines:

```
student@personal-pc:~$ ssh -L 8890:localhost:8890 xxx@tryglaw.ii.uni.wroc.pl \
>                      ssh -L 8890:localhost:8890 hera \
>                      ssh -L 8890:localhost:8890 lab110-01.stud.ii
```

which allows direct access to an interactive session of `jupyter notebook`. Start the session with:

```
student@lab110-01:~$ jupyter notebook --ip="*" --port=8890
```

Further, it might be a good idea to:

a) set up passwordless communication using ssh,
`http://www.cyberciti.biz/faq/how-to-set-up-ssh-keys-on-linux-unix/`

b) set up proper config file to further ease the connection process,
`http://blog.sanctum.geek.nz/uses-for-ssh-config/`

(there are plenty of other resources all over the web how to achieve this).

5 Remote Computations

First, get to know basic commands like `top, htop, who, w, nvidia-smi`, etc. listing available resources, logged users, running processes and machine load.

By default, when you log out the processes you have started are terminated by receiving the hangup signal `https://en.wikipedia.org/wiki/SIGHUP`. Please do not attempt to run your computations by leaving your session open overnight, as the computations might be interrupted

in case of a sudden drop of the connection.

`GNU Screen` makes for a very convenient, albeit a bit more complicated tool. It allows a few processes to share one physical terminal, and has support for detaching and leaving the whole setup running even after logging out. Again, there are plenty of tutorials on the web, some of them listed at `http://aperiodic.net/screen/start`.

Note: By default, multithreaded Python processes might actually be executed on a single core `https://en.wikipedia.org/wiki/Global_interpreter_lock`.

6 Etiquette

Remember that you're sharing resources with your colleagues, which might be relevant in the face of an upcoming deadline. Try to use the machines efficiently by not allocating more CPU-bound processes on single machines, than the number of available cores. The labs will be monitored for extreme cases of resource hogging, but moderate load should be fine in most cases.