Tips for running code on ScAi Servers

The Connection flows of servers at UCLA

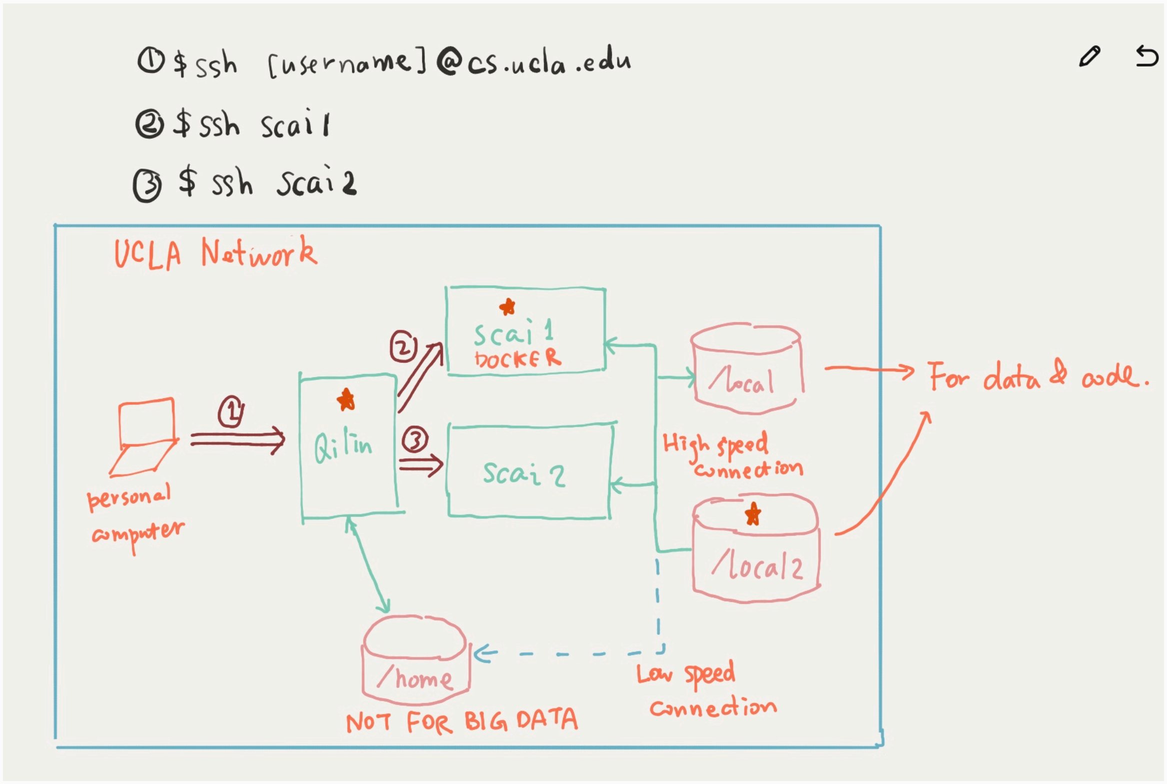
UCLA ScAi Lab currently has four servers for computation and storage. For general ML tasks, `qilin`, `Scai1` and `Scai2` would be the main workforce. `qilin` is the entrance of ScAi servers. Only through `Qilin` can users under UCLA network access `scai1` and `scai2`.

The figure below shows the linkages between the computing modules and storages modules. Please use the following commands to log in the right server to perform tasks with heavy computational cost. Do **NOT** run anything heavy on `qilin`.

1. (1) --> (2) to `scai1`
2. (1) --> (3) to `scai2`

You **MUST** have all the data/code related to your project saved in `/local` or `/local2` which are only acceessible from `scai1` and `scai2`. Do **NOT** save anything in `/home`. When you first log in Scai servers, please do the following:

1. (in scai1 or scai2) `$cd /local` or `$ cd /local2`
2. `$ mkdir [your_id]` to create a dir that only belongs to you
3. `$ cd [your_id]` and do whatever you want here



Login `qilin`.

1. Make sure you are in UCLA network (or on UCLA VPN).
2. Run

```
$ ssh [user_name]@qilin.cs.ucla.edu
```

... then type in the password.

3. Then you should see the cmd like this

```
$ [user_name]@qilin:~
```

4. Tired of typing password everytime? Check this out. Then you will be able to directly do `$ssh [alias]`. The `[alias]` is the name you set as the shortcut of that ssh connection.

Login in `scai1` or `scai2`.

`qilin` is the entry point of the rest three servers: `scai1` and `scai2`.

1. Login `scai1` and `$ssh scai1` and then put in password.
2. *[On `scai1` only]* Launch a docker container as a "virtual machine":

```
$ nvidia-docker run -it --rm -v
/local2/zyli/scdpmia:/workspace/scdpmia
nvcr.io/nvidia/tensorflow:18.04-py3

$ nvidia-docker run -it --rm -v
/local2/zyli/scdpmia:/workspace/scdpmia
nvcr.io/nvidia/pytorch:19.10-py3
```

For multiple folders to mount:

```
$ nvidia-docker run -it --rm -v [out_dir1]:
[in_dir1] -v [out_dir2]:[in_dir2] -v [out_dir3]:
[in_dir3] [nvcr.io/nvidia/tensorflow:18.04-py3
```

Inside Docker of `scai1`

1. **[Recommended]** Use `tmux` to keep the context so you can logoff anytime without worrying about the connection.

```
$ tmux  # open tmux
$ tmux list-session  # open active sessions
$ tmux a -t [session_name]  # attach an
active session
```

2. Use `nvidia-smi` to print the current usage of GPUs. The cmd will print a snapshot of the current usage, use `watch -n0.5 nvidia-smi` to see the monitoring with dynamic update every 0.5s.

   ```
   $ nvidia-smi  # for a snapshot of resource usage
   $ watch -n0.5 nvidida-smi  # constantly update nvidia0-smi every 0.5s
   ```

3. Use `htop` or `top` to monitor the CPU/Memory usage.

   ```
   $ top  # real-time resource monitor
   $ htop  # graphical top, needs to be installed first
   ```

4. Use `pip install [package name]` to install python packages.

   ```
   $ pip install [package name]  # install a single package
   $ pip install -r requirements.txt  # install packages according to file
   ```

How to use PyCharm and VS Code to edit the code on `scai1` or `scai2`

VSCode [Recommended]

1. Follow instruction on this link: Link

2. Edit `Users/your_id/.ssh/config` and add the following settings:

    ```
    Host qilin
        HostName qilin.cs.ucla.edu
        User [username]
    Host scai1
        HostName scai1.cs.ucla.edu
        User [username]
        ProxyJump qilin
        LocalForward 8888 localhost:8888

    Host scai1
        HostName scai2.cs.ucla.edu
        User [username]
        ProxyJump qilin
        LocalForward 9999 localhost:9999
    ```

PyCharm

1. Port forwarding: Link
2. Setup remote interpreter: Link
3. Keep a local copy of all code. PyCharm edit the local code and maintain a real-time auto synchronization with the files on `/local2` . VSCode directly edit files on `/local2` through ssh connection.

(Almost) Best practices

Edit code:

By VSCode with Remote-SSH extension.

In Terminal

1. Create a new `tmux` session or attach to the session specifically for the project.

   ```
   [Outside tmux]
   $ tmux # <-- create new session
   $ tmux list-sessions # <-- list all sessions
   $ tmux a -t [session_name] # <-- attach a session

   [Inside tmux]
   <ctrl-b>, $ # <-- rename session
   <ctrl-b>, %/<Double Quote>  # <-- split panel
   <ctrl-b>, <Up>/<Down>/<Left>/<Right> # <-- switch panel
   <ctrl-b>, c/1/2/3/4  # <-- create/switch window
   <ctrl-b>, d # <-- detach session
   ```

2. In one window, run docker container **ONLY** for run programs. Check out Tmux Cheat Sheet for better knowledge.
3. In other windows, without docker, run normal Linux commands.

H3

H3