Workshop 3 Remote computing

Introduction

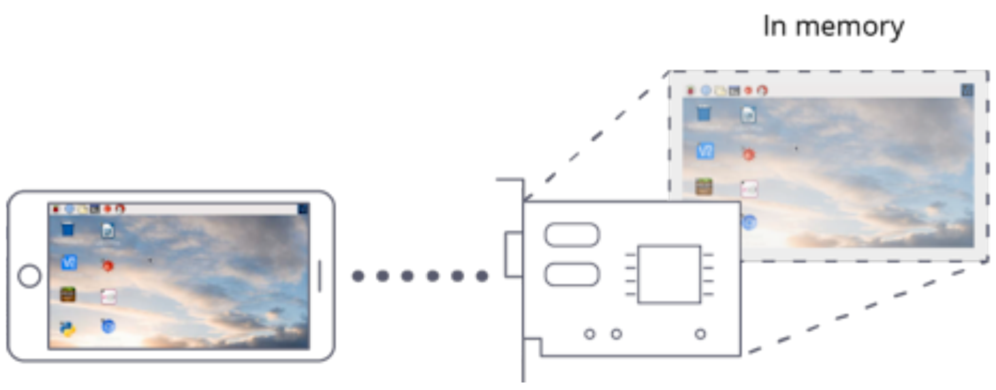
- Run on local shell

```
{ echo -n "scale=50;"; seq 1 2 200 | xargs -n1 -I{} echo '(16*(1/5)^{}/{}-4*(1/239)^
```

- Your local processor did the calculation. If calculation was more complex, it would have run for longer, possible for days.
    - Cannot stop your laptop
    - Possibly block any other tasks...
- Idea: **delegate** (have someone else to do it for you)
    - **Colleague computer** (*not recommended)
    - **Cluster computing** - conceptually it is essentially smashing up many machines to make a really big & powerful one. Those remote computers share physical resources: CPU, Storage, Memory etc.
        - ideal for task that can be break in smaller independent/parallel actions: e.g. BLAST (each CPU align a subset of the sequences or database)
    - **Cloud computing** is simply an aggregate of computing power. Remote (virtual) computers share some resources: Each computer has independent CPU and memory.
        - not just raw computing, also entire services, or storage...
    - **Grid computing**. A grid is simply many computers which together might solve a given problem/crunch data. The fundamental difference between a grid and a cluster is that in a grid each node is relatively independent of others; problems are solved in a divide and conquer fashion.
        - ideal for long tasks with variable requirement (CPU, memory etc.): not massivement parallel

How does that work

Using a client software to connect to the remote server/service and works as if you were on your local computer: e.g. using **Gmail**. Your internet browser relay all your action/command to a remote server that store, send or receive email. You only consult the results.



Remote access login

One essential tool to master is SSH.

SSH, or Secure Shell, is a protocol used to securely log onto remote systems. It is the most common way to access remote Linux and Unix-like servers.

The tool for connecting to a remote system using SSH is called, unsurprisingly, `ssh`.

The most basic form of the command is:

```
ssh remote_username@remote_host
```

The `remote_host` in this example is the IP address or domain name that you are trying to connect to.

Once you have connected to the server, you will probably be asked to verify your identity by providing a password.

To exit back into your local session, simply type:

```
exit
```

```
$ ssh guest@XXX.XXX.XXX.XXX

> { echo -n "scale=50;"; seq 1 2 200 | xargs -n1 -I{} echo '(16*(1/5)^{}/{}-4*(1/239
> exit
```

Similarly, out can upload and download file using FTP (File Transport Protocol) and SFTP (Secure FTP).

Basic SSH Usage

1. On the command prompt, type `ssh remote_username@remote_host`.
2. Enter your password.
3. Run the desired program.
4. Type `exit` to disconnect and logout.

Advanced SSH Usage

Modern processing power combined with automated scripts make brute forcing a password-protected account very possible. SSH keys prove to be a reliable and secure alternative.

SSH key pairs are two cryptographically secure keys that can be used to authenticate a client to an SSH server. Each key pair consists of a public key and a private key.

The private key is retained by the client and should be kept absolutely secret. Any compromise of the private key will allow the attacker to log into servers that are configured with the associated public key without additional authentication. As an additional precaution, the key can be encrypted on disk with a passphrase.

The associated public key can be shared freely without any negative consequences. The public key can be used to encrypt messages that only the private key can decrypt. This property is employed as a way of authenticating using the key pair.

When a client attempts to authenticate using SSH keys, the server can test the client on whether they are in possession of the private key. If the client can prove that it owns the private key, a shell session is spawned or the requested command is executed.

How To Create SSH Keys

The first step to configure SSH key authentication to your server is to generate an SSH key pair on your local computer by typing:

```
ssh-keygen
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/username/.ssh/id_rsa):

Created directory '/home/username/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Next, you will be prompted to enter a passphrase for the key. This is an optional passphrase that can be used to encrypt the private key file on disk.

```
Your identification has been saved in /home/username/.ssh/id_rsa.
Your public key has been saved in /home/username/.ssh/id_rsa.pub.
```

Copying your Public Key Manually

The content of your `id_rsa.pub` file will have to be added to a file at `~/.ssh/authorized_keys` on your remote machine.

To display the content of your `id_rsa.pub` key, type this into your local computer:

```
cat ~/.ssh/id_rsa.pub
```

You will see the key's content, which may look something like this:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCqql6MzstZYh1TmWWv11q5O3pISj2ZFl9HgH1JLknLLx44
```

Access your remote host using password method. Once you have access to your account on the remote server, you should make sure the `~/.ssh` directory is created. This command will create the directory if necessary, or do nothing if it already exists:

```
mkdir -p ~/.ssh
```

Now, you can create or modify the authorized_keys file within this directory. You can add the contents of your `id_rsa.pub` file to the end of the `authorized_keys` file, creating it if necessary, using this:

```
echo "public_key_string" >> ~/.ssh/authorized_keys
```

In the above command, substitute the `public_key_string` with the output from the cat `~/.ssh/id_rsa.pub` command that you executed on your local system. It should start with `ssh-rsa AAAA....`

If this works, you can move on to try to authenticate without a password.

```
ssh remote_username@remote_host
```

Grid or Cloud computing - how to run a basic process: `screen`

If you perform a long-running task on a remote machine and suddenly your connection drops, the SSH session is terminated and your work is lost. `screen` allows to resume the sessions.

Screen or GNU Screen is a terminal multiplexer. In other words, it means that you can start a screen session and then open any number of windows (virtual terminals) inside that session. Processes running in Screen will continue to run when their window is not visible even if you get **disconnected**.

Starting Linux Screen

To start a screen session, simply type screen in your console:

```
screen
```

This will open a screen session, create a new window and start a shell in that window.

Now that you have opened a screen session you can get a list of commands by typing:

`Ctrl+a` `?`

Starting Named Session

Named sessions are useful when you run multiple screen sessions. To create a named session, run the screen command with the following arguments:

```
screen -S session_name
```

It's always a good idea to choose a descriptive session name.

Working with Linux Screen Windows

When you start a new screen session by default it creates a single window with a shell in it.

You can have multiple windows inside a Screen session.

To create a new window with shell type `Ctrl+a` `c`, the first available number from the range `0...9` will be assigned to it.

Below are some most common commands for managing Linux Screen Windows:

- `Ctrl+a` `c` Create a new window (with shell)
- `Ctrl+a` `"` List all window
- `Ctrl+a` `0` Switch to window 0 (by number)
- `Ctrl+a` `S` Split current region horizontally into two regions
- `Ctrl+a` `tab` Switch the input focus to the next region
- `Ctrl+a` `Ctrl+a` Toggle between the current and previous region
- `Ctrl+a` `Q` Close all regions but the current one
- `Ctrl+a` `X` Close the current region

Detach from Linux Screen Session

You can detach from the screen session at any time by typing:

`Ctrl+a` `d`

The program running in the screen session will continue to run after you detach from the session.

Reattach to a Linux Screen

To resume your screen session, use the following command:

```
screen -r
```

In case you have multiple screen sessions running on your machine you will need to append the screen session ID after the r switch.

To find the session ID list the current running screen sessions with:

```
screen -ls
```

```
There are screens on:
    10835.pts-0   (Detached)
    10366.pts-0   (Detached)
```

If you want to restore screen `10835.pts-0` , then type the following command:

```
screen -r 10835
```

Basic Linux Screen Usage

1. On the command prompt, type `screen` .

2. Run the desired program.
3. Use the key sequence `Ctrl-a` + `d` to detach from the screen session.
4. Reattach to the screen session by typing `screen -r`.