CS3210 – 2016/17 Semester I

Lab 2
Setting up a Cluster

Objectives:

1. Learn basic hardware and software setup for a simple cluster.
2. Learn basic message passing programming environment.

In this lab, you will learn how to set up a parallel computer cluster with two Dell nodes using the Ubuntu operating system. Ubuntu is a popular Linux distribution mostly used for personal computer / laptop, but has steadily gain user base in the server sector.

As a practice, you will add a Nvidia Jetson TK1 node to the cluster in the later part of the exercise.

As there are only 7 sets of hardware components (i.e. enough for 7 clusters), you have to form a team of **3 members** for this lab.

A. Setting up a 2-node cluster

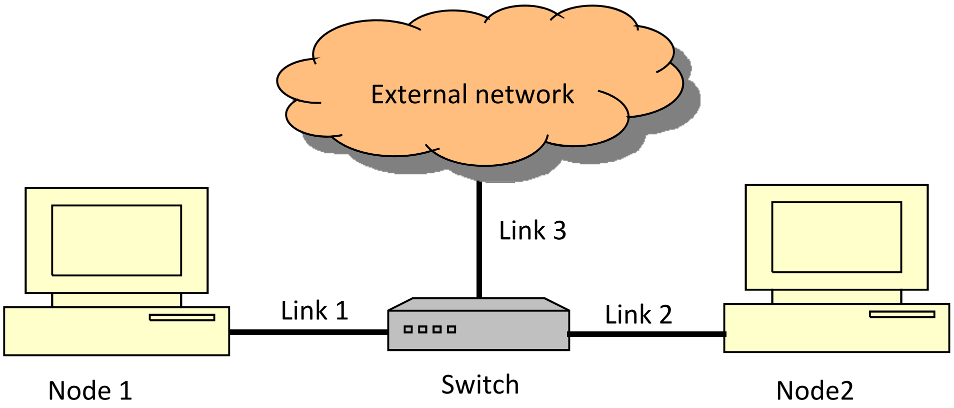
There are a total of 5 steps to setup the 2-node cluster:

1. Set up the hardware
2. Install Ubuntu
3. Install essential software
4. Configure SSH access
5. Test cluster by running an MPI[1] parallel program

Step 1: Set up Hardware

To setup a parallel cluster consisting of two nodes, connect the hardware in your workbench as follows:

[1] MPI (Message-passing library interface) is a parallel programming library for writing message-passing parallel program on distributed memory system. We will cover MPI later in the lecture. In this lab you will only run a simple hello world program to check the communication among processes running on different processors (cores) in the parallel cluster system.



Node 1 (8 cores) is a **Dell Optiplex 990** with tower form factor and node 2 (4 cores) is a **Dell Inspiron** all-in-one desktop (see Appendix for photos).

Step 2: Install Ubuntu

Install Ubuntu 14.04 Server LTS[2] on all nodes using the given Live CD. Detailed screenshots can be found at http://ubuntuserverguide.com/2014/04/how-to-install-ubuntu-server-14-04-trusty-tahr.html

The main steps to install Ubuntu Server 14.04 LTS are (trivial steps are omitted!):

a. "Select language", "Select Install Ubuntu Server", "Select installation language", "Select your location", "Configure the keyboard" …

b. Next, "Configure the network", choose the Ethernet interface, then fill in the hostname of your node (the hostname is on the yellow label on the monitors, e.g., soctf-pdc-**002** and soctf-pdc-**009**).

c. "Set up users and passwords" – set up with the following user on each node:

	Node 1 **(Dell Optiplex)**	**Node 2** **(Dell Inspiron)**
Full name	**node1**	**node2**
Username	**node1**	**node2**
Password	**node1**	**node2**

When prompted to "encrypt your home directory": ***No***

[2] The latest stable built is 16.04 as of August, 2016. However, to ensure compatibility with all different platforms, we have opted to use a slightly older version.

d. Next, “Partition disks”. When prompted to "unmount partitions that are in use", select "***Yes***".

   Choose “**Manual**” partitioning as the partitioning method. From the list of partitions, select the partition with the name ***FREE SPACE***.
   Note: The list of partitions on your machine may be different. Please consult the tutor.

   Next, select “*Automatically partition the free space*”, select “*Finish partitioning and write changes to disk”.*

   > **The tutor will inspect your setup before you proceed to the next step.** Changes to the harddisk partition is hard to reverse, so please be patient and verify before proceed.

   After your setup has been inspected, choose *“Write changes to disks: Yes”.*

e. Configure HTTP Proxy: Leave blank and select ***Continue***

f. Configure Taskel: select “*No automatic updates*”

g. Software selection.

   > **Do not press <Enter>. Use <Space Bar> to select the necessary software.**

   Select “*OpenSSH server*” then ***Continue***

h. Install the GRUB boot loader on a hard disk: ***Yes***

If the installation is successful, “Installation complete” is displayed. Select “Continue” to reboot the system. Next, login to each node using the username and password as defined earlier.

Step 3: Install Essential Software

Install the package “build-essential” (include make, gcc, etc. software development tools) on all nodes:

```
$ sudo apt-get install build-essential
```

If you are prompted for password, just enter the password of the user (e.g. User: "Node1" ➔ Password: "Node1").

Install “OpenMPI”, an implemention of MPI, on all nodes.

```
$ sudo apt-get install libopenmpi-dev openmpi-bin openmpi-doc
```

Install the package “vim” editor on all nodes:

```
$ sudo apt-get install vim
```

Installing “OpenSSH”client and server on all nodes.

```
$ sudo apt-get install openssh-server openssh-client
```

Step 4: Configure SSH Access

The nodes will communicate each other using their full hostname. For example, the full hostname of a node with hostname **soctf-pdc-002** is **soctf-pdc-002.comp.nus.edu.sg**.

Assume we have the following two nodes:

Node	Full Hostname
node1	**soctf-pdc-002.comp.nus.edu.sg**
node2	**soctf-pdc-009.comp.nus.edu.sg**

First, ensure the node can reach each other via network.

On node ***node1***:

```
$ ping soctf-pdc-009.comp.nus.edu.sg
```

On node ***node2***:

```
$ ping soctf-pdc-002.comp.nus.edu.sg
```

On both nodes, create a new user called ***cs3210***:

```
$ sudo useradd –s /bin/bash –d /home/cs3210 –m cs3210
$ sudo passwd cs3210
```

Enter “`pc2016`” as the password.

Exit from the current user.

```
$ exit
```

To enable SSH authentication without password, we need to generate and exchange public/private key pairs on both nodes:

i. login with username **cs3210** and password **pc2016**.

ii. Generate public/private key pair.

```
$ssh-keygen -tdsa
```

[Press enter a few times to ignore the prompts]

This command generates two files: `id_dsa` and `id_dsa.pub` in the `~/.ssh` folder. `id_dsa` is the private key and `id_dsa.pub` is the public key.

We need to keep the private key and copy the public key to the other node.

To exchange the public keys, use the **`ssh-copy-id`** command to copy `id_dsa.pub`. Also, to facilitate MPI, we will also add **the node's own public key** to the authorized list.

On node ***node1*** (copy its public key, `id_dsa.pub,` to node2):

```
$cd .ssh
$ssh-copy-id –i id_dsa.pub soctf-pdc-009.comp.nus.edu.sg
$cat id_dsa.pub >> authorized_keys
```

On node ***node2*** (copy its `id_dsa.pub` to node1):

```
$cd .ssh
$ ssh-copy-id –i id_dsa.pub soctf-pdc-002.comp.nus.edu.sg
$cat id_dsa.pub >> authorized_keys
```

Now, node1 and node2 should be able to SSH each other without password. Do the following check on each node (change the hostname accordingly):

```
$ ssh cs3210@soctf-pdc-009.comp.nus.edu.sg
```

[Check that you can login without password]

```
$ exit
```

> Caution: As we have the same user id on both nodes, it may be easily confused (for example, SSH into ***node2*** and mistakenly thought that you are still working with ***node1*** etc). So, refrain from SSH into another node after this check.

Step 5: Test cluster by running an MPI parallel program

In this step, we are going to run an MPI program on the cluster. On ***node1***, download `hello.c` program:

```
$ wget www.comp.nus.edu.sg/~sooyj/cs3210/Lab2/hello.c
```

You can find the full source code in the appendix.

First, compile the program on ***node1***:

```
$ mpicc hello.c –o hello
```

Next, copy the binary to node ***node2***.

```
$ scp hello cs3210@soctf-pdc-009.comp.nus.edu.sg:~
```

In order to run the MPI program on the cluster, create a file called `hostfile` using the editor `vim` (or any text editor you are familiar with) as follows:

```
$vim hostfile
```

(to start editing in `vim`, press `<Esc>:i`)

Add the list of hostnames of nodes in the cluster as follows:

```
soctf-pdc-002.comp.nus.edu.sg
soctf-pdc-009.comp.nus.edu.sg
```

(to save and exit from `vim` press `<Esc>:wq`)

Lastly, execute the program:

```
$mpirun –hostfile hostfile –np 2 hello
Hello world from process 0 of 2 running on soctf-pdc-002
Hello world from process 1 of 2 running on soctf-pdc-009
```

Well done, you have successfully setup a working parallel system!

You can try to increase the number of independent process from 2 to 4:

```
$mpirun –hostfile hostfile –np 4 hello
```

to observe the output.

B. Adding a Nvidia Jetson TK1 node to the cluster

Ubuntu 14.04 LTS is already installed in the embedded-class low-power Jetson TK1[3] system and a user account, ***cs3210***, has been created. Your tasks are:

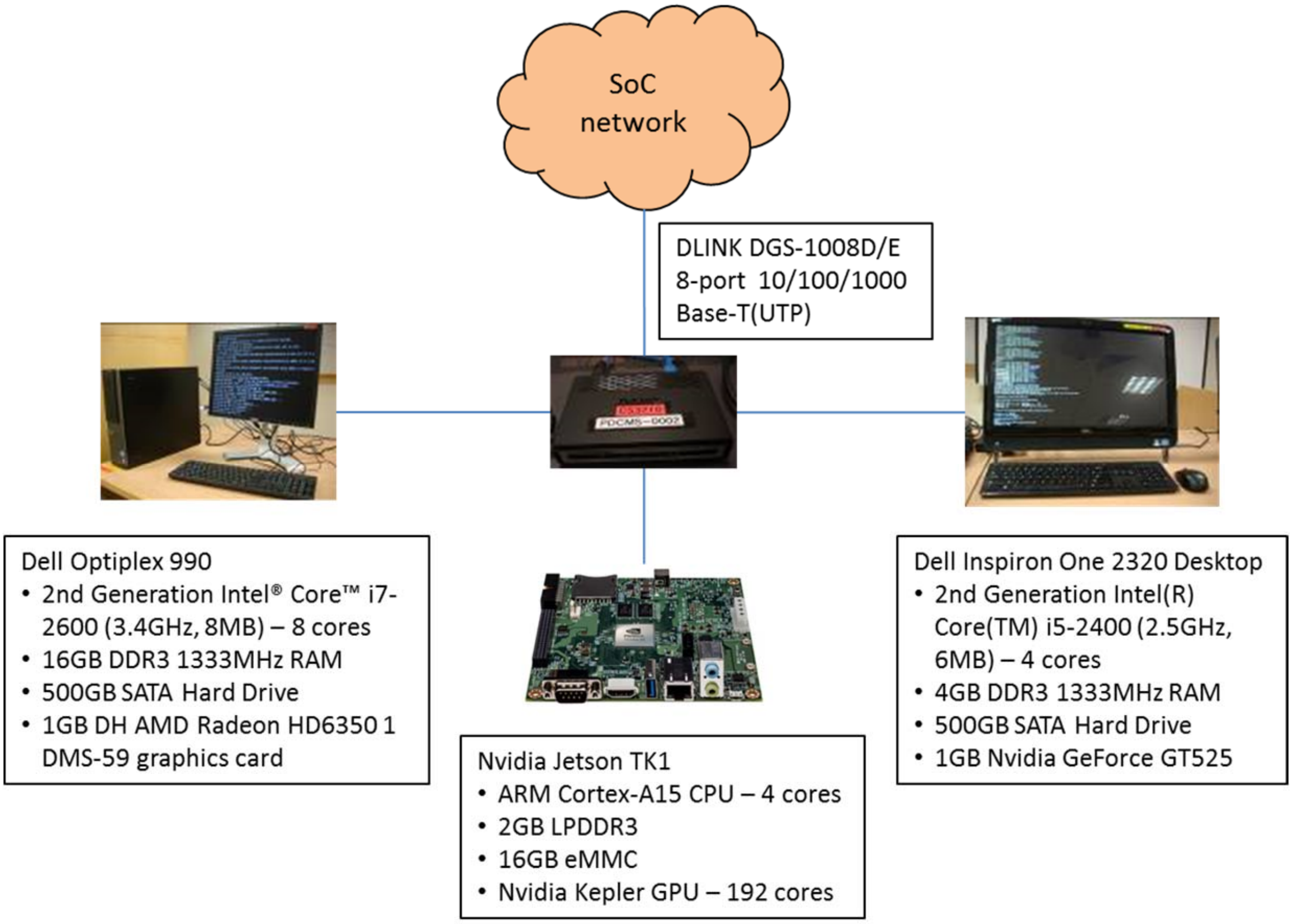
1. Add this system to the 2-node cluster
2. Configure the ssh access (same step 4 in section A)
3. Run the MPI "Hello World" program on the 3-node cluster (similar to step 5 in section A). (Hint: node1 and node2 are both x86 machines, while jetson is on ARM processor).

Once your 3-node cluster is working, please ask the TA to verify.

[3] More about Jetson TK1 at http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

Appendix

Cluster Configuration



DLINK DGS-1008D/E
8-port 10/100/1000
Base-T(UTP)

Dell Optiplex 990

- 2nd Generation Intel® Core™ i7-2600 (3.4GHz, 8MB) – 8 cores
- 16GB DDR3 1333MHz RAM
- 500GB SATA Hard Drive
- 1GB DH AMD Radeon HD6350 1 DMS-59 graphics card

Nvidia Jetson TK1

- ARM Cortex-A15 CPU – 4 cores
- 2GB LPDDR3
- 16GB eMMC
- Nvidia Kepler GPU – 192 cores

Dell Inspiron One 2320 Desktop

- 2nd Generation Intel(R) Core(TM) i5-2400 (2.5GHz, 6MB) – 4 cores
- 4GB DDR3 1333MHz RAM
- 500GB SATA Hard Drive
- 1GB Nvidia GeForce GT525

Installing Network Hardware

Plug one end of the cable into a node and the other end into a switch

Source Code for the MPI Program `hello.c`

```
/* C Example */
#include <stdio.h>
#include <mpi.h>

int main (int argc, char *argv[])
{
        int rank, size;
        char msg[80];
        int length;

        /* starts MPI */
        MPI_Init (&argc, &argv);

        /* get current process id */
        MPI_Comm_rank (MPI_COMM_WORLD, &rank);

        /* get number of processes */
        MPI_Comm_size (MPI_COMM_WORLD, &size);

        /* Gets the name of the processor (computer name) */
        MPI_Get_processor_name(msg, &length);

        printf( "Hello world from process %d of %d running on %s
\n", rank, size, msg);

        /* exiting */
        MPI_Finalize();
        return 0;
}
```