Lab 0 - Getting Started

Objectives

At the end of this lab you should be able to login to the MIC server from a laptop, and query the status of the MIC cards on the Host system.

Terminology

In this lab, we use the terms:

- **Client** or **Laptop**: The laptop, used to connect to the ***Host*** server
- **Host** or **Server**: system hosting the Intel® Xeon Phi™ coprocessor card(s)
- **Target** or **MIC Card**: Intel® Xeon Phi™ coprocessor card(s)

Activity 1: Getting Connected

1. Write down you client user ID and password here – this is the one you use to logon to your laptop (or client terminal).

ID	
Password	

Table 1- Client login details

2. If you are connecting via wifi, record the details here:

SSID	
Key	

Table 2- Wifi login details

3. There will be teams working on the same coprocessor. You'll get assigned the coprocessor name to use (e.g. **`mic0`**, **`mic1, etc`**) and user name for both host & coprocessor (e.g. **`user1`**).

- Record your details below.
- Also record if any other users are sharing the same MIC card as you.

User Name	
Password	
Host IP address	
SSH port number	
MIC assigned to me	
Am I sharing this MIC with anyone (if, so list the users)	

Table 3 server login details

Important Note: The Lab material will use **`mic0`** as default coprocessor name. You'll may be assigned a different MIC number, please make sure you use ***your*** mic number in the labs!

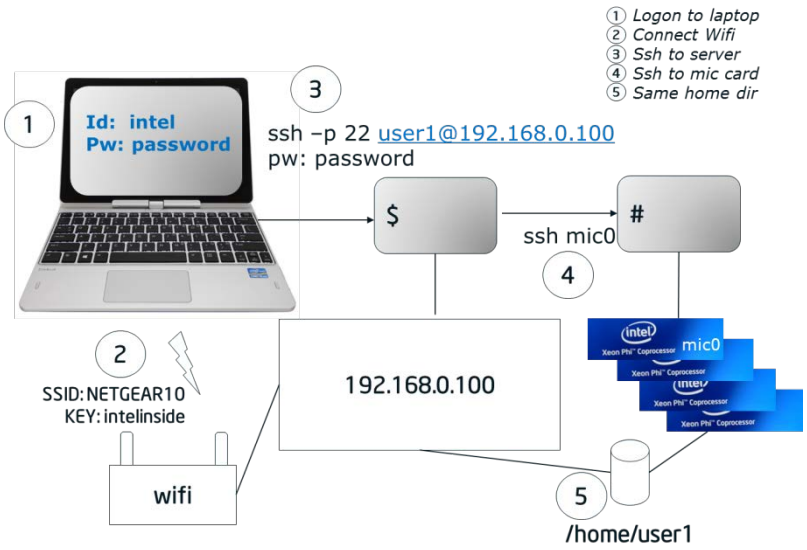

Figure 1 – A typical lab setup

4. Now log onto you client computer, and then SSH to the server computer. Once you are connected to the server computer, try to ssh to your MIC card (remember to use the MIC number you recorded in *Table 3*. Do this in the following order (example in figure 1)

- Logon to laptop (client)
- If needed, Connect wifi
- Ssh to the server

```
ssh user1@192.168.0.100
user1@192.168.0.100's password:
```

```
Last login: Sun May 25 20:02:55 2014 from 192.168.0.3
[user1@knc ~]$ login as: user1
Password:
```

- Check the path of your home directory

```
[user1@knc ~]$ pwd
/home/user1
```

- Create an empty file in your home director – give the file a unique name (so for example, you name)

```
[user1@knc ~]$ echo "" > JoeBloggs
[user1@knc ~]$ ls
JoeBloggs
```

- Now ssh to the mic card, and look at the contents of the home directory

```
[user1@knc ~]$ ssh mic0
user1@knc-mic0:~$ pwd
/home/user1
user1@knc-mic0:~$ ls
JoeBloggs
```

If you can see the file Joe Bloggs,, then that confirms that home directory on the server is the same as on the mic card

Read this if your $HOME directory on MIC and HOST is not the same . . .

If your $HOME directory is not shared between host and target, then first look to see if you can find any other folder that is shared (for example /micfs or /tmp).

If there are no shared folders, then you will have to use scp to copy any files across:

e.g.

```
scp ~/JoeBloggs  mic0:
```

- Now log of the mic card by typing ‘exit’

```
user1@knc-mic0:~$ exit
logout
Connection to mic0 closed.
```

Activity 2: Checking the Status of the MIC card

What's our status?

Before we start working with the coprocessor we query the status to see whether it's up and running.

- Use the **micctrl** tool to check the status by issuing the following command (notice there are TWO hyphens before the option 'status':

```
[user1@knc ~]$ micctrl –status
mic0: online (mode: linux image: /usr/.../bzImage-
knightscorner)
```

If the coprocessor is running and its image has been booted you'll see **mic0: online**.

- If a different status is reported (e.g. **mic0: ready**), check to see if the **mpss** service is running:

```
$ service mpss status
mpss is stopped
```

*If the **mpss** service is not running, please consult one of the trainers.*

Finding out more details using micinfo

- Use the utility micinfo to find out more about the host system and the MIC cards that it contains.

```
[user1@knc ~]$ micinfo
MicInfo Utility Log
Copyright 2011-2013 Intel Corporation All Rights Reserved.
Created Mon May 26 15:43:23 2014
```

- Record some of the details in the table below

Host OS	
OS Version	
MPSS version	
Number of MIC cards in server	
Number of Active Cores	

GDDR Size	
GDDR Speed	

Table 4- Some details about the host and target

Using micsmc – the Platform Status Panel

- if you enabled X11 forwarding when you first connected to to the server (the –X option used with ssh) then look at Intel® Xeon Phi™ Coprocessor Platform Status Panel

```
[user1@knc ~]$ micsmc &
```

Initially, there's only the summary pane visible. A separate pane with more details can be added for each coprocessor.

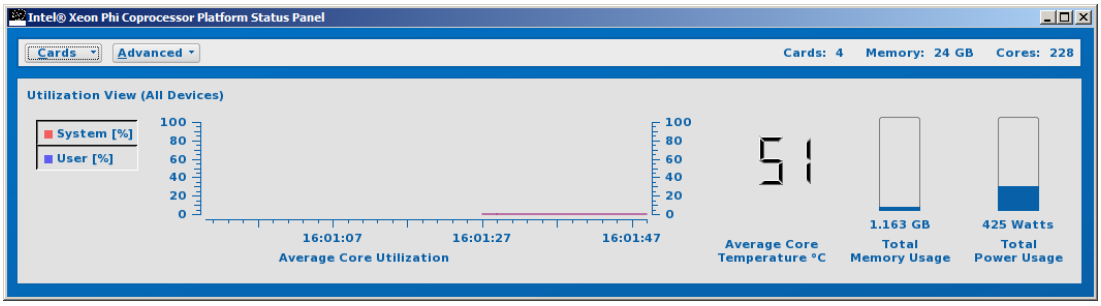

Figure 2 –The Platform Status Panel.

- Use the menu ***Cards*** and select the coprocessor that has been assigned to you for the lab exercises. If you have exclusive use of the host system, then we recommend to leave it open while you work with the coprocessor and switch to the *Core Histogram View* as shown in the screenshot above.

- Explore all the options in the ***Advanced*** button, and then answer the following questions.

Is it possible to reset the card from this utility?
Can you find the details you recorded in Table 4 using micsmc?

Activity 3: Create and run a Coprocessor Application

Now, you'll create an application for the coprocessor. More precisely, the first example will use the offload extensions. You'll learn more about it later. Please, build the application and execute it:

```
$ icc hello_offload.c –o hello_offload
$ ./hello_offload

Hello, World from host! # threads = 48
Number of Xeon Phi cards = 2
Hello, World from Phi 0 # threads = 240
Hello, World from Phi 1 # threads = 240
```

The first two lines are printed from the host, the remainder of the lines are from the coprocessor

Now set the environment variable **OFFLOAD_DEVICES** and re-run **hello_offload**. *What do you notice?*

```
export  OFFLOAD_DEVICES=0
```

Building an application

- To make the Parallel Studio XE tools available from a shell, source the following scripts (or add the commands to your ./bash_profile):

```
source /opt/intel/composerxe/bin/compilervars.sh  intel64
source /opt/intel/vtune_amplifier_xe/amplxe-vars.sh
source /opt/intel/impi/4.1.3/bin64/mpivars.sh
```

- Now let's create a pure coprocessor (native) application:

```
$ icc -mmic hello.c -o hello_mic
```

The option **-mmic** to instructs the compiler to create a native Intel® MIC application. This application can't be executed on the host system because of the different architecture.

```
$ ssh mic0
 cd <to the path where you built your application>
 ./hello_mic
  Hello World!
```

Congratulations, you've now executed your first two applications for the coprocessor!

Activity 4: Hello World – Different Flavours

In the Lab0 directory, you will find hello world written using different parallel languages\constructs.

- Briefly look at the contents of each source file, then build and run each example.
- Build and run each application using the instructions below

1. Host OpenMP

```
Build:     icc hello_openmp.c –openmp -o hello_openmp_host

Run:       ./hello_openmp_host
```

2. MIC OpenMP

```
Build:     icc hello_openmp.c –openmp -mmic -o hello_openmp_mic

Run:       ./hello_openmp_mic
```

3. **MPI Host**

```
Build:     mpiicc hello_mpi.c -o hello_mpi

Run:       mpirun -np 2 ./hello_mpi
```

4. **MPI MIC – run from MIC**

```
Build:     mpiicc hello_mpi.c -mmic -o hello_mpi.mic

Run (from Phi):
           ssh mic0
           source /opt/intel/impi/4.1.3/mic/bin/mpivars.sh
           mpirun -np 2 ./hello_mpi.mic
```

5. **MPI MIC – run from HOST**

```
Build:     (use same binary as previous)

Run:       export I_MPI_MIC=enable

           mpirun -host mic0 -np 2 ./hello_mpi.mic

(or see **):

           mpirun -hostfile machines_mic -np 2 ./hello_mpi.mic
```

6. **MPI on MIC and HOST**

```
Build:     (use same binaries as previous two examples)

Run:       export I_MPI_MIC=enable
           export I_MPI_FABRICS=shm:tcp

           mpirun -host mic0 -np 1 ./hello_mpi.mic : -host
             localhost –np 1 ./hello_mpi

(or see **):

      export I_MPI_MIC_POSTFIX=.mic

      mpirun -hostfile machines_all -np 2 –ppn 1 ./hello_mpi
```

** Sample Hostfile(s)

```
### machines_mic
mic0
mic1

### machines_all
# NOTE replace next line!
# use the hostname command to find out the proper name

localhost
mic0
```