Xeon Phi Basics

Adrian Jackson

June 22, 2016

Contents

1 Aims

The aim of this exercise is to get you logged onto and set up on the Hartree Xeon Phi machine. You will then run a series of tests to get you familiarised with the Hartree environment and working with Xeon Phis.

You can find more details on the Hartree system at:

- http://community.hartree.stfc.ac.uk/wiki/site/eecrp/xeon phi user guide.html

The Hartree machine currently has 42 Xeon Phi cards installed. These are connected to 42 hosts nodes. Each host node has one Xeon Phi card called *mic0* connected to it. We will be using the batch system to spread the load across Xeon Phi machines so you should all be able to get access to your own node.

2 Introduction

In this exercise you will use your provided guest account to:

1. log onto the host nodes;
2. copy practical material from a central location to your account;
3. unpack the practical material archive;

4. load compilers and other Intel libraries necessary for using the Xeon Phi;
5. use runtime tools to check the status of the Xeon Phi cards;
6. login to the Xeon Phi;
7. run parallel codes on the host and on the Xeon Phi;

Please do ask questions if you do not understand anything in the instructions - this is what we are here for.

3 Instructions

3.1 Log into login nodes host and run commands

You should have been given an account ID – referred to generically here as `myaccount` – and password.

3.1.1 Procedure for Mac and Linux users

Open a command line *Terminal* and enter the following command:

```
user@laptop$ ssh -Y myaccount@phase2-login1.wonder.hartree.stfc.ac.uk
Password:
```

you should be prompted to enter your password. This will log you into Hartree compute cluster. From there you can login to one of the nodes with a Xeon Phi connected to it using:

```
user@laptop$ bsub -q phiq -Is bash
user@laptop$ source /etc/profile.d/modules.sh
```

3.1.2 Procedure for Windows users

Windows does not generally have SSH installed by default so some extra work is required. You need to download and install a SSH client application - PuTTY is a good choice:

- http://www.chiark.greenend.org.uk/~sgtatham/putty/.

When you start PuTTY you should be able to enter the Hartree login address (myaccount@phase2-login1.wonder.hartree.stfc.ac.uk). When you connect you will be prompted for your username and password.

By default, PuTTY does not send graphics back to your local machine. We will need this later for viewing the sharpened image, so you should "Enable X11 Forwarding" which is an option in the "Category" menu under "Connection -> SSH -> X11". You will need to do this each time before you log in with PuTTy.

3.2 Download and extract the exercise files

Once you have logged into the host machine you need to obtain a copy of the source code. You can obtain this using *wget* as follows:

```
export http_proxy=http://wwwcache.dl.ac.uk:8080
export https_proxy=$http_proxy
export ftp_proxy=$http_proxy
wget https://www.archer.ac.uk/training/course-material/2016/06/
xeonphi_soton/exercises/exercise1/setup.tar.gz
```

This package includes executables for CPU-only, Xeon Phi native, and offload versions of a simple hello world application. Also included in the package is an env.sh file which can be used to setup the environment for Xeon Phi application development.

To unpack the archive use:

```
[host]$ tar -xzvf setup.tar.gz
setup/cpu_only.exe
setup/env/env.sh
setup/env/env_mic.sh
setup/mic_native.exe
setup/offload.exe
..
```

To enter the extracted folder:

```
[host]$ cd setup
```

3.3 Setup host environment

First we need to setup the environment to use Intel compilers and Intel MPI. This will be useful for following practicals so you are more comfortable with the process. We will not be compiling any source code, but we will run some test applications to make sure your system is up and running as expected.

On this system the environment can be easily set up using module loads and swaps. For the purposes of this practical try:

```
[host]$ module load intel/16.0.047_mic intel_mpi/5.1.3_mic
```

To confirm that you are using the expected setup, try:

```
  [host]$ which mpirun
/gpfs/stfc/local/apps/intel/intel_cs/2016.2.062/impi/5.1.3.181/intel64/bin/mpirun
  [host]$ which ifort
/gpfs/stfc/local/apps/intel/intel_cs/2016.0.047/bin/ifort
```

And compare your system's output to the above.

ALTERNATIVES:

Instead of switching modules manually, you could use a setup file such as the provided `env.sh` file (in the `env` folder) by executing:

```
[host]$ source env/env.sh
```

Please note that not all systems are set up this way. A more general way of loading environment variables for Intel compilers and MPI (and other Intel tools) is by sourcing the scripts provided by Intel in the Intel setup folders. The process above is equivalent to:

```
  [host]$ source /gpfs/stfc/local/apps/intel/intel_cs/
2016.2.062/impi/5.1.3.181/intel64/bin/mpivars.sh
  [host]$ source /gpfs/stfc/local/apps/intel/intel_cs/
2016.0.047/compilers_and_libraries_2016/linux/bin/compilervars.sh intel64
```

3.4 Use some runtime tools to check mic status

One very commonly used tool when you first use a machine with Xeon Phi cards is the `micinfo` command which is part of the Intel MPSS (Manycore Platform Software Stack).

Enter the command:

```
[host]$ micinfo
```

Study the output to confirm the number of Xeon Phi cards and their status, memory and other characteristics.

Similarly, one can use the following commands for more information on the Xeon Phi cards of a system.

```
[host]$ miccheck
[host]$ micsmc
[host]$ cat /etc/hosts | grep mic
```

Finally, we will log into a Xeon Phi card and see how many cores it has by using the commands:

```
[host]$ ssh mic0
[micX]$ cat /proc/cpuinfo | grep proc | tail -n 3
processor : 237
processor : 238
processor : 239
```

Now return to host to continue with the next exercise, using the command:

```
[mic0]$ exit
[host]$
```

3.5 Run host test

First try running the simple OpenMP hello world application provided `cpu_only.exe` on the host. Also, change the number of OpenMP threads to something sensible.

```
[host]$ export OMP_NUM_THREADS=6

[host]$ ./cpu_only.exe

Running on HOST
---------------
Hello from thread 0
Hello from thread 4
..
Hello from thread 3
Total number of threads used is 6
Maximum number of threads available is 6
================================
```

3.6 Run offload test

Now lets try running the offload application. It will a similar OpenMP region on the host, followed by an OpenMP region which is offloaded to the Xeon Phi.

```
[host]$ ./offload.exe

Hello from HOST process 0 of 1
Hello from HOST thread 0 on process 0
..
Hello from HOST thread 2 on process 0

-----------------------------------------------------------
```

```
Number of HOST threads used = 6 on process 0
Max HOST threads available  = 6 on process 0
-----------------------------------------------------------

Hello from XeonPhi thread = 0
Hello from XeonPhi thread = 4
..
Hello from XeonPhi thread = 2

-----------------------------------------------------------
Number of XeonPhi threads used = 6
Max Xeon Phi threads available = 6 on process 0
-----------------------------------------------------------
```

As you can see, the number of OpenMP threads on the Xeon Phi is automatically set to be the same as that on the host (this may vary from setup to setup). To change this we must execute the following commands:

```
[host]$ export MIC_ENV_PREFIX=MIC
[host]$ export MIC_OMP_NUM_THREADS=10
```

(The first command need only be run once.)

Now try running the offload application again.

```
[host]$ ./offload.exe

Hello from HOST process 0 of 1
..
Hello from HOST thread 5 on process 0

-----------------------------------------------------------
Number of HOST threads used = 6 on process 0
Max HOST threads available  = 6 on process 0
-----------------------------------------------------------

Hello from XeonPhi thread = 0
..
Hello from XeonPhi thread = 1

-----------------------------------------------------------
Number of XeonPhi threads used = 10
Max Xeon Phi threads available = 10 on process 0
-----------------------------------------------------------
```

In cases where we are running a more complex code we may be interested to see statistics of how much we utilised the Xeon Phi card vs the CPU. A handy tool for such quick feedback is the `OFFLOAD_REPORT` environment variable.

```
[host]$ export OFFLOAD_REPORT=1
```

enables the output of a brief offload report. Running the `offload.exe` application again should now give you additional output similar to:

```
[Offload] [MIC 0] [File]                        offload.c
[Offload] [MIC 0] [Line]                        36
[Offload] [MIC 0] [Tag]                         Tag 0
[Offload] [HOST]  [Tag 0] [CPU Time]            1.932612(seconds)
[Offload] [MIC 0] [Tag 0] [MIC Time]            0.039789(seconds)
```

3.7 Run native test

For the native tests, again, each user should ssh into their assigned mic card.

```
[host]$ ssh mic0
[mic0]$ cd setup
```

Now we need to setup the environment for the mic card:

```
  [mic0]$ ulimit -s unlimited
  [mic0]$ source /gpfs/stfc/local/apps/intel/intel_cs/
2016.0.047/mkl/bin/mklvars.sh mic
  [mic0]$ source /gpfs/stfc/local/apps/intel/intel_cs/
2016.2.062/impi/5.1.3.181/mic/bin/mpivars.sh
```

ALTERNATIVES:

Instead of sourcing Intel scripts manually, you could use a setup file such as the provided `env_mic.sh` file (in the `env` folder) by executing:

```
[host]$ source env/env_mic.sh
```

Now execute the binary which has been compiled for the Xeon Phi:

```
[micX]$ export OMP_NUM_THREADS=32
[micX]$ ./native.exe
Running on Xeon Phi
---------------
Hello from thread 24
Hello from thread 5
..
..
Hello from thread 31
Total number of threads used is 32
Maximum number of threads available is 32
================================
```

Finally, to give you a better feel of the core/thread distribution, enable affinity information output using the `KMP_AFFINITY` environment variable and re-run the `native.exe` application:

```
[micX]$ export KMP_AFFINITY=verbose
[micX]$ ./native
OMP: Info #204: KMP_AFFINITY: decoding x2APIC ids.
OMP: Info #205: KMP_AFFINITY: cpuid leaf 11 not supported - decoding
 legacy APIC ids.
```

```
OMP: Info #149: KMP_AFFINITY: Affinity capable, using global cpuid info
..
..
```

Which gives you a full report on the thread affinities for all cores and threads of the Xeon Phi card, given your (default) affinity settings.