FOR THE INTEL® XEON PHI™ COPROCESSOR

In this example dialog, we will collect power and execution data from a native application executing on the Intel Xeon Phi coprocessor, and then extract and reduce the resultant data into a form (CSV or Comma Separated Values) that can be analyzed using a spread sheet program. We use "MIC" or "KNC" as a shorthand for the Intel Xeon Phi coprocessor.

The host system (named host6) has two coprocessors (named knightscorner6-mic0 and knightscorner6-mic1). This experiment uses knightscorner6-mic1. A separate data analysis system (named twkidd-MOBL5) is used to do the data reduction. See Figure 1.

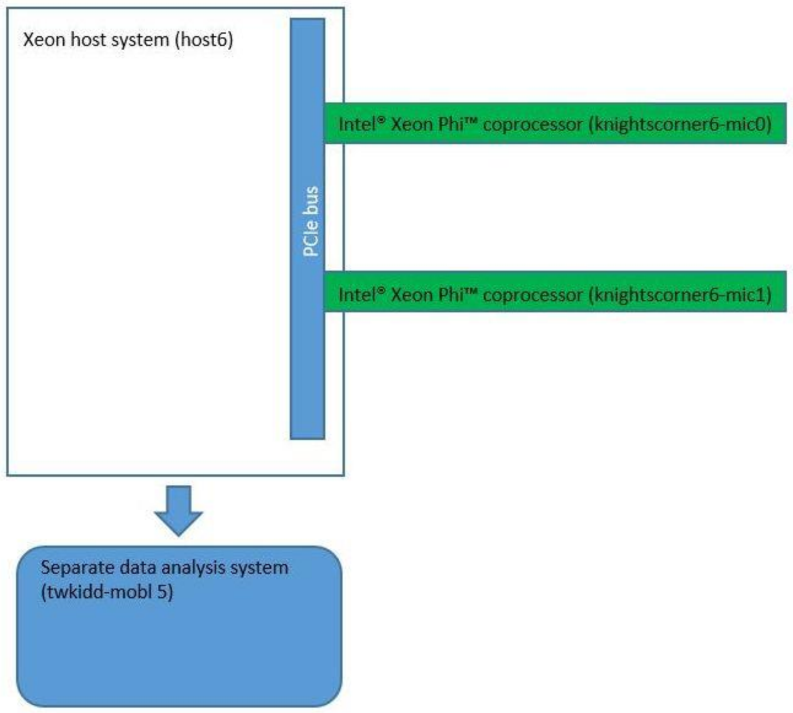
Xeon host system (host6)
PCIe bus
Intel® Xeon Phi™ coprocessor (knightscorner6-mic0)
Intel® Xeon Phi™ coprocessor (knightscorner6-mic1)
Separate data analysis system (twkidd-mobl 5)

Figure 1 Experimental setup.

PROGRAMS/SCRIPTS USED:

libiomp5.so	OpenMP runtime library compiled for KNC
collectData.sh	Bash script to collect data from multiple runs. extractDataKNC.sh parses this data into a form useable for analysis.
ep.B.knc, ep.C.knc	NAS Parallel Benchmark EP compiled for problem class size B and C, respectively
extractDataKNC.sh	Script to extract the raw data from the MIC data collection and convert it into a CSV form

STARTING ASSUMPTIONS:

- You have the KNC versions of the NAS EP benchmarks already compiled for native execution.*
- You have placed these KNC versions as well as this lab's "collectData.sh" script into the host directory ~/test.

STEP 1: (on host) Find the OpenMP library cross-compiled for the MIC and copy it over to ~/test.

```
# get the proper libiomp5.so library for the coprocessor
[twkidd@host6 ~]$ find /opt/intel/ -iname "libiomp5.so"
/opt/intel/composer_xe_2015.0.090/compiler/lib/ia32/libiomp5.so
/opt/intel/composer_xe_2015.0.090/compiler/lib/intel64/libiomp5.so
/opt/intel/composer_xe_2015.0.090/compiler/lib/mic/libiomp5.so
[twkidd@host6 ~]$ cd test/
[twkidd@host6 test]$ cp
/opt/intel/composer_xe_2015.0.090/compiler/lib/mic/libiomp5.so ./
[twkidd@host6 test]$ ls
collectData.sh  ep.B.knc*  ep.C.knc*  libiomp5.so*
[twkidd@host6 test]$
```

Your version of the compiler will probably be newer. Make sure you copy the 'mic' version.

STEP 2: (on host) Copy the data collection script (collectData.sh), the NAS EP benchmarks (ep.B.knc and ep.C.knc) and the MIC OpenMP library (libiomp5.so) to the coprocessor. Note that these instructions assume that the user modified collectData.sh to collect data from the coprocessor (and not the Xeon processor). Instructions for collecting for both the Intel Xeon processor and Intel Xeon Phi coprocessor are embedded in collectData.sh as comments.

This host has two coprocessors, mic0 and mic1. We will use mic1.

```
[twkidd@host6 test]$ su
Password:
[root@host6 test]# ls
collectData.sh  ep.B.knc  ep.C.knc  libiomp5.so
[root@host6 test]# scp * mic1:/tmp/
collectData.sh                              100% 1879     1.8KB/s   00:00
ep.B.knc                                    100%  837KB 836.9KB/s   00:00
ep.C.knc                                    100%  837KB 836.9KB/s   00:00
libiomp5.so                                 100% 1182KB   1.2MB/s   00:00
[root@host6 test]#
```

STEP 3: (on coprocessor) Log into and setup the coprocessor for collecting the data. Note that there may already be other files (e.g. foo) in the /tmp directory. Just ignore them.

Make sure you modify the collectData.sh script for collecting data from the coprocessor. The script may be setup for collecting data from an Intel Xeon processor. Embedded in the script are instructions for how to do this. It involves little more than commenting out the Xeon and uncommenting the MIC setup parameters.

```
[root@host6 test]# ssh mic1
[root@knightscorner6-mic1 ~]# cd /tmp
[root@knightscorner6-mic1 tmp]# ls
COI2MB          collectData.sh  ep.C.knc        libiomp5.so
coi_procs       ep.B.knc        foo
[root@knightscorner6-mic1 tmp]# export LD_LIBRARY_PATH=.
[root@knightscorner6-mic1 tmp]# more collectData.sh

#This script is very simple intentionally. Making a bullet
#    proof script would have obscured the key statements

#To use, change to reflect the benchmark you are using (e.g. ep.D.x)
#    and the naming of the log files (e.g. epC_${sys_name}_...)

affinity=scatter
#affinity=compact

#for KNC
sys_name=knc
num_expts=4 #for KNC
num_threads=244 #for KNC
num_threads_core=4
benchmark=./ep.B.knc
log_prescript=epB

#for Xeon
#sys_name=xeon
#num_expts=4
#num_threads=32
#num_threads_core=2
#benchmark=./ep.C.x
#log_prescript=epD

#sample the nth thread per core
#  Xeon (hyperthreading) n = {1|2}
#  KNC (4 threads/core) n = {1|2|3|4}
sample_thread=1

#
^C
```

Note: For practical reasons, the above listing of "`more collectData.sh`" is interrupted with a '`^C`' and does not show the entire script.

STEP 4: (on coprocessor) Collect and log performance and power data. Depending upon the benchmark class you use, it can take long time.

```
[root@knightscorner6-mic1 tmp]# sh ./collectData.sh
Experiment Parameters
```

```
   affinity(scatter)
   number of experiments(4)
   total number of threads per experiment(244)
   number of threads per core(4)
   sampling thread number 1 of 4
   benchmark (./ep.B.knc)
   log files are epB*



EXPT 1
15913172125293337414549535761656973778185899397101105109113117121125129133137
14114514915315716116516917317718118518919319720120520921321722122522923323724
1

EXPT 2
15913172125293337414549535761656973778185899397101105109113117121125129133137
14114514915315716116516917317718118518919319720120520921321722122522923323724
1

EXPT 3
15913172125293337414549535761656973778185899397101105109113117121125129133137
14114514915315716116516917317718118518919319720120520921321722122522923323724
1

EXPT 4
15913172125293337414549535761656973778185899397101105109113117121125129133137
14114514915315716116516917317718118518919319720120520921321722122522923323724
1[root@knightscorner6-mic1 tmp]#  ls
COI2MB                         epB_knc_1_244_4_scatter_1.log
coi_procs                      epB_knc_1_244_4_scatter_2.log
collectData.sh                 epB_knc_1_244_4_scatter_3.log
collectData.sh_old             epB_knc_1_244_4_scatter_4.log
ep.B.knc                       foo
ep.C.knc                       libiomp5.so
[root@knightscorner6-mic1 tmp]#
```

STEP 5: (on host) Copy the collected data log files back to the host

```
[root@host6 test]# scp mic1:/tmp/*.log ./
epB_knc_1_244_4_scatter_1.log                 100%   93KB  93.4KB/s   00:00
epB_knc_1_244_4_scatter_2.log                 100%   93KB  93.4KB/s   00:00
epB_knc_1_244_4_scatter_3.log                 100%   93KB  93.4KB/s   00:00
epB_knc_1_244_4_scatter_4.log                 100%   93KB  93.4KB/s   00:00
[root@host6 test]# ls
collectData.sh                 ep.C.x                         nohup.out
dummy.sh                       ep.D.x                         pwr-read
epB_knc_1_244_4_scatter_1.log  epD_xeon_1_32_2_scatter_1.log  rtdummy.sh
epB_knc_1_244_4_scatter_2.log  epD_xeon_1_32_2_scatter_2.log  sav
epB_knc_1_244_4_scatter_3.log  epD_xeon_1_32_2_scatter_3.log
epB_knc_1_244_4_scatter_4.log  epD_xeon_1_32_2_scatter_4.log
[root@host6 test]#
```

STEP 6: (on data analysis system) Parse and convert power performance data into a CSV form for analysis using the extractDataKNC.sh script. The output below is done from a Cygwin bash shell. If you are doing your analysis on a Microsoft Windows system, depending upon your path, you may or may not be able to execute 'excel' from your command line.

```
twkidd@twkidd-MOBL5 /cygdrive/c/@usr/writing/whitePapers/PearlsPwrChapter/webSiteCode
$ ls
#extractDataKNC.sh#*      epB_knc_1_244_4_scatter_1.csv  extractedData.csv
#readMe#                  epB_knc_1_244_4_scatter_1.log  pwr-read-xeon
collectData.sh            extractDataKNC.sh*             pwr-read-xeon.c
collectData.sh_150529a*   extractDataKNC.sh~*            pwr-read-xeon.c~
collectData.sh_150529b    extractDataXeon.sh*            sav/
collectData.sh~           extractDataXeon.sh_r0          test.sh
cview-example-file.cview  extractDataXeon.sh~*

twkidd@twkidd-MOBL5 /cygdrive/c/@usr/writing/whitePapers/PearlsPwrChapter/webSiteCode
$ sh ./extractDataKNC.sh
This script parses and extracts only log data obtained from
    the Intel Xeon Phi coprocessor

Usage: ./extractDataKNC.sh <source_log_file> <generated_extracted_csv_data_file>

twkidd@twkidd-MOBL5 /cygdrive/c/@usr/writing/whitePapers/PearlsPwrChapter/webSiteCode
$ sh ./extractDataKNC.sh epB_knc_1_244_4_scatter_1.log epB_knc_1_244_4_scatter_1.csv
This script parses and extracts only log data obtained from
    the Intel Xeon Phi coprocessor
epB_knc_1_244_4_scatter_1.log data extracted and placed into
epB_knc_1_244_4_scatter_1.csv in csv form

twkidd@twkidd-MOBL5 /cygdrive/c/@usr/writing/whitePapers/PearlsPwrChapter/webSiteCode
$ head -15 epB_knc_1_244_4_scatter_1.csv
"(blank)","CPU time","Threads Total","Threads Avail","Mops/s","Mops/s/thread","Win1
Pwr (uW)","Win2 Pwr (uW)"
,274.73,1,1,7.82,7.82,101000000,101000000
,61.91,5,5,34.69,6.94,101000000,102000000
,34.71,9,9,61.86,6.87,101000000,102000000
,24.28,13,13,88.46,6.80,102000000,101000000
,18.69,17,17,114.89,6.76,103000000,102000000
,14.79,21,21,145.20,6.91,102000000,102000000
,11.04,25,25,194.57,7.78,136000000,135000000
,9.51,29,29,225.88,7.79,138000000,137000000
,8.37,33,33,256.67,7.78,143000000,142000000
,7.46,37,37,287.80,7.78,147000000,146000000
,7.33,41,41,293.05,7.15,106000000,106000000
,6.83,45,45,314.22,6.98,104000000,103000000
,6.32,49,49,339.91,6.94,102000000,103000000
,5.80,53,53,370.04,6.98,104000000,109000000

twkidd@twkidd-MOBL5 /cygdrive/c/@usr/writing/whitePapers/PearlsPwrChapter/webSiteCode
$ excel epB_knc_1_244_4_scatter_1.csv &
[1] 6304

twkidd@twkidd-MOBL5 /cygdrive/c/@usr/writing/whitePapers/PearlsPwrChapter/webSiteCode
$
```

STEP 7: (on data analysis system) Open the CSV file using your favorite spreadsheet program. Try to reproduce the analysis presented in the Chapter using the various plot functions.

===

General Notes:

- For the Xeon, the maximum value for “energy” is 16 bits for a maximum value of 2^16). As such, the “Energy Used” value (energy_after-energy_before) can be a negative value. For example, if energy_before is 65480 J and energy_after is 44 J, the “Energy Used” value becomes -65436 J. In such a case, adjust the value accordingly.
- Verify the maximum value is 2^16 by doing your own experiments.

FOR THE INTEL XEON PROCESSOR

A very similar dialog is used to collect and reduce data from an Intel® Xeon® Processor based server. To collect and extract data from the server, comment out the MIC coprocessor code, and uncomment out the Intel Xeon Processor code in “collectData.sh”. Note that “pwr-read-xeon.c” only works on newer microarchitectures, namely Sandy Bridge (SNB) or later. Earlier microarchitectures, Nehalem (NHM) or earlier, use non-standard PMU event names or do not have the events.

FOOTNOTES:
* You can find the NAS Parallel Benchmarks at http://www.nas.nasa.gov/publications/npb.html. The binaries are not included in the lab’s zip file because the benchmarks are constantly being updated and moving location. Instructions for creating the Intel® Xeon Phi™ coprocessor versions are included in the zip file you obtain from the site.