Report on the usage of Beacon Intel Xeon Phi cluster at University of Tennessee, Knoxville.

Prakashan Korambath

June 13, 2014

Principal Investigator: Prakashan Korambath, Institute for Digital Research and Education, University of California, Los Angeles, CA 90095
E-mail: ppk@idre.ucla.edu
Investigation period: Jun and July 2013.

Project: Benchmarking Linear Algebra subroutines on Intel Xeon Phi Coprocessors.

The main motivation behind this investigation is to get an exposure on Intel's Xeon architecture and to measure the performance of basic linear algebra subroutines using Intel's MKL library.

The Xeon Phi processor used in this investigation had 61 cores with 4 threads per core, high speed bidirectional ring connecting the cores, 512 bit register for SIMD operations, running Linux OS, Intel Fotran and C/C++ compilers and debuggers, 8 GB GDDR5 memory, connected to the host Xeon processor through PCIe x16 TCP/IP socket, 512 KB L2 cache, 32 KB L1 data cache and CPU speed of 1.053 GHz.

The host machine had Intel Xeon CPU E5-26700 with CPU Speed 2.60 GHz, 256 GB of RAM and 32 processors per node with hyper threading turned on. The network interconnect was made of FDR IB fabric and storage file system used Lustre for scratch space.

The benchmarks were run in native mode as well offline mode. In the native mode all jobs were run on the coprocessors and the compiler flage '-mmic' was used during compilation. In the offline mode, the jobs were started on the host process the parallel region of the code was manually specified using pragmas or compiler directives. Offline mode also involves data transfer from host node to coprocessors. There is also a run time variable called KMP_AFFINITY with values compact, scattered or balanced.

Some of the compiler time and run time options are given below.

```
icc      -openmp hello_omp.c      -o a.out.omp_cpu
icc      -mmic -openmp hello_omp.c      -o a.out.omp_mic

export OMP_NUM_THREADS=16
  ./a.out.omp_cpu

export KMP_AFFINITY="granularity=balanced"
export OFFLOAD_REPORT=2

export MIC_PREFIX=MIC
```

export MIC_OMP_NUM_THREADS=240
./a.out.omp_off

Results

Figure 1 below shows the performance with and without Intel MKL subroutine library on the host operating system for matrix matrix multiply using 16 OpenMP threads. The MKL libraries shows significant advantage when matrix size is larger than 5000.

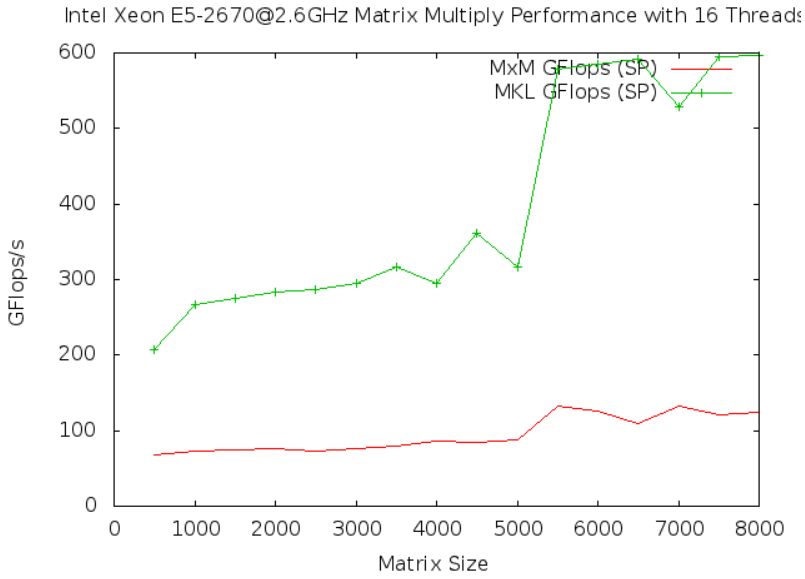

Figure 1. GFlops vs matrix size plot for the performance of Matrix Matrix multiple on the host machine using 16 threads.

In Figure 2 below we plotted the performance of matrix multiply on Xeon Phi Coprocessor using different number of threads. The highest performance was obtained around 240 cores or at 120 cores.

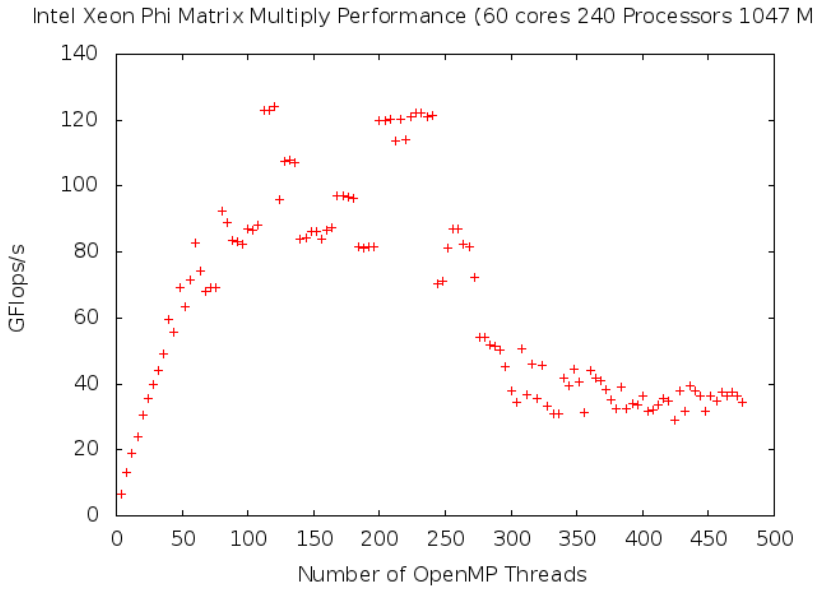

Figure 2. Performance vs number of OpenMP threads on Xeon Phi Coprocessor

In Figure 3 below we plotted the performance obtained when part of the code is offloaded from host machine to the coprocessor. Intel MKL library subroutines performed much better on coprocessors.

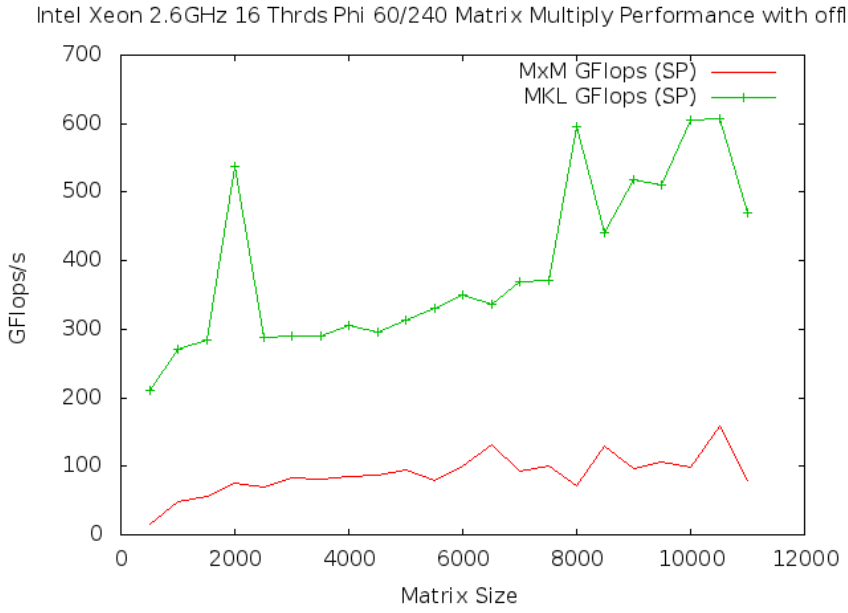

Figure 3. Matrix Matrix multiply performance using offloading from host CPU to Xeon Phi Coprocessor on 240 threads.

In Figure 4 below we plotted the matrix matrix multiply performance vs size of the matrix. The Intel MKL library performance was much higher as soon as matrix size started increasing.

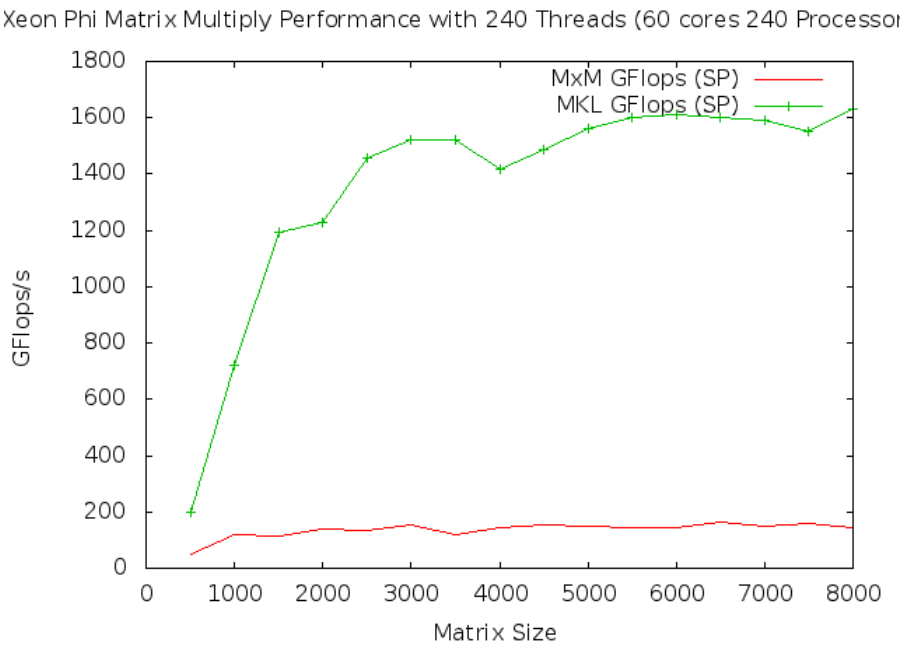

Figure 4. Matrix Matrix multiply performance on Intel Xeon Phi processor run natively.

Conclusion:

The coprocessor combined with Intel MKL library has approximately 2 to 3 times performance advantage over the host processor. However, the co-processor has limitations of 8 GB RAM.