Offload Mode: Estimation of Execution Time and Cost of Data Movement

Arturo Argueta, Roberto Camacho Barranco, Esthela Gallardo, and Pat Teller
UTEP Stampede Technology Insertion Project
Leonardo Fialho and Jim Browne
UT-Austin Stampede Technology Insertion Project

To determine whether or not to execute a code segment on a Xeon Phi, you need to know the code's execution time on a Phi, the number of bytes moved between a Sandy Bridge (SB) CPU and the Phi, and the execution-time cost of the data movement. This information and other details can be obtained using the environment variable `OFFLOAD_REPORT`. To attain the execution time of offloaded code and the amount of data transferred, use the following command, which sets `OFFLOAD_REPORT` to 2.

```
export OFFLOAD_REPORT = 2
```

For more information regarding `OFFLOAD_REPORT` see: *http://software.intel.com/en-us/node/463216*. For more information on how to set the `OFFLOAD_REPORT` variable see: *http://software.intel.com/en-us/node/463218*

Cost per Byte Transferred

As described below, using the following code designed to execute in Offload Mode on a Phi, we determined that the average cost per byte transferred from the SB to the Phi is 2.11724E-09 seconds/byte, the median value is 2.12762E-09, and the standard deviation is 8.82518E-11.

To get this information we conducted a set of experiments using the following code that, using `OpenMP`, adds two arrays, `a` and `b`, of floating-point numbers to a third array, `c`. In this case, an experiment is the execution of the code with a specific value of `size`.

```
#include <stdio.h>
#include <omp.h>
#include <time.h>
#include "offload.h"
#include <sys/time.h>

__attribute__ ((target(mic))) const int size = 650000000;
__attribute__ ((target(mic))) float a[size];
__attribute__ ((target(mic))) float b[size];
__attribute__ ((target(mic))) float c[size];

int main()
{
```

```
    struct timeval start, end;
  // Start timing buffer initialization loop.
  gettimeofday(&start, NULL);

  // Initialize buffers.
  for (int i = 0; i < size; ++i) {
      a[i] = (float)i;
      b[i] = (float)i;
      c[i] = 0.0f;
  }

  // End timing buffer initialization loop.
  gettimeofday(&end, NULL);
  // Print execution time of buffer initialization loop.
  printf ("Total time before pragma = %f seconds\n",
          (double) (end.tv_usec - start.tv_usec) / 1000000 +
          (double) (end.tv_sec - start.tv_sec));

  // Start timing offloaded computational loop.
  gettimeofday(&start, NULL);

  /* Offload the following code to mic 0. Arrays a, b, c are
     transferred to the Phi and back to the SB. */
  // Compute vector add; C <- C = A + B
  #pragma offload target(mic:0) inout(a,b,c)
  #pragma omp parallel for default(none) shared(a,b,c)
  for (int i = 0; i < size; ++i) {
      c[i] = a[i] + b[i];
  }

  // End timing offloaded computational loop.
  gettimeofday(&end, NULL);
  // Print execution time of offloaded computational loop.
  printf ("Total time after pragma = %f seconds\n",
          (double) (end.tv_usec - start.tv_usec) / 1000000 +
          (double) (end.tv_sec - start.tv_sec));

  return 0;
}
```

To execute this code on a Phi, it was compiled using the following command:

```
icc vectorAddOffload.cc -O3 -openmp -mcmodel medium -o
vectorAddOffload
```

To determine the data transfer cost between the SB CPU and Phi, we executed `vectorAddOffload` for various values of `size`, including values that cause more than 2GB of data to be transferred to the Phi. As a result, the compiler flag `-mcmodel medium` had to

be used in order to eliminate the memory restriction on the size of transferred data. For more information see:
http://software.intel.com/sites/products/documentation/doclib/stdxe/2013/composerxe/compiler/fortran-mac/GUID-4F6CCA3F-42C1-4DF4-9847-67D43DD9652B.htm

Executing with different values of `size` provided the data shown in Table 1 below. Note that this reported data is the product of executing each instantiation of the program (i.e., the program with a particular value of `size`) 10 times and computing the average of the 10 reported execution times of the pragma offload code section, the volume of data transferred, i.e., number of offloaded bytes, the average execution time on the Xeon Phi, and average execution time to set up the vectors on the Phi. The cost per byte transferred (seconds) is presented in the last column of Table 1 for the various values of `size`, shown in the first column. An entry in the column labeled "Total Data Transfer Cost" was calculated by subtracting the average execution time of `vectorAddOffload` with the specified value of `size` on the Xeon Phi (pure computation time without the data movement overhead) from the total average execution time of the offload code section.

Value of `size`	**Total Number of Bytes Offloaded**	**Total Data Transfer Cost (seconds)**	**Cost/Byte (seconds)**
150,000,000	1,800,000,000	4.0515398	2.25086E-09
160,000,000	1,920,000,000	4.2261224	2.20111E-09
170,000,000	2,040,000,000	4.4405635	2.17675E-09
175,000,000	2,100,000,000	4.5691335	2.17578E-09
180,000,000	2,160,000,000	4.6639486	2.15924E-09
200,000,000	2,400,000,000	4.9719436	2.07164E-09
300,000,000	3,600,000,000	7.3488369	2.04134E-09
400,000,000	4,800,000,000	9.6832266	2.01734E-09
500,000,000	6,000,000,000	12.5760721	2.09601E-09
600,000,000	7,200,000,000	14.2725937	1.98231E-09

Table 1. vectorAddOffload Data Movement Costs for Different Values of `size`.

Over all the runs (with different values of `size`) the average cost per byte is 2.11724E-09 seconds/byte, the median value is 2.12762E-09, and the standard deviation is 8.82518E-11.