SEPTEMBER 10TH, 2015 PRE-CLASS QUESTIONS

ELLIOT CARTEE

Question 1. For each of the totient nodes, they have the specifications given here:
`http://ark.intel.com/products/83352/Intel-Xeon-Processor-E5-2620-v3-15M-Cache-2_40-GHz`
We see that they have a max memory bandwidth of 59 GB/s.

They also have a peak flop rate of:

$$(12 \text{ cores })(16 \text{ flops per cycle })(2.4\text{Ghz}) = 460.8 \text{ Gflops/s}$$

Therefore at an arithmetic intensity below about 7.81 flops/byte, we get a performance of:

$$(\text{Arithmetic Intensity}) \times (59\text{GB/s})$$

while above that arithmetic intensity, we get a performance of the peak flops rate 460.8 Gflops/s.

Here is a plot of the roofline diagram generated by `roofline.py` in this directory:

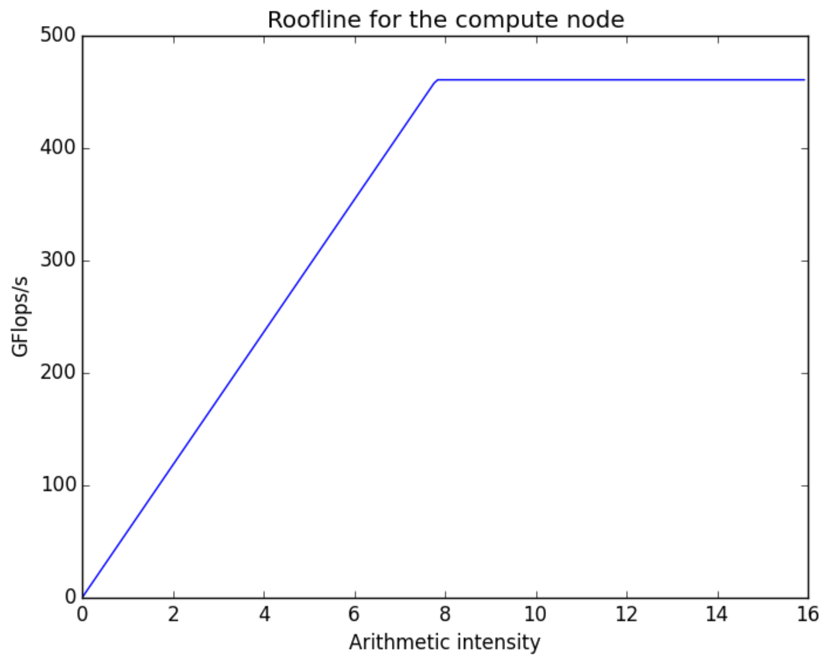

FIGURE 1. Roofline diagram for just the compute node

Now, if we also include the two Phi coprocessors, each has a peak flop rate of:

$$(60 \text{ cores})(16 \text{ flops/cycle})(1.053\text{GHz}) = 1010.88\text{GFlops/s}$$

giving us a total peak flop rate of:

$$2 \times (1010.88\text{GFlops/s}) + 460.8 \text{ Gflops/s} = 2482.56 \text{ Glops/s}$$

Now, the compute node has a max memory bandwidth of 59 GB/s, and each Phi coprocessor has a max memory bandwidth of 320 GB/s, which I believe should give a max memory bandwidth of:

$$2 \times (320 \text{ GB/s}) + (59 \text{ GB/s}) = 699 \text{ GB/s}$$

Using `roofline.py` again, we get a roofline diagram:

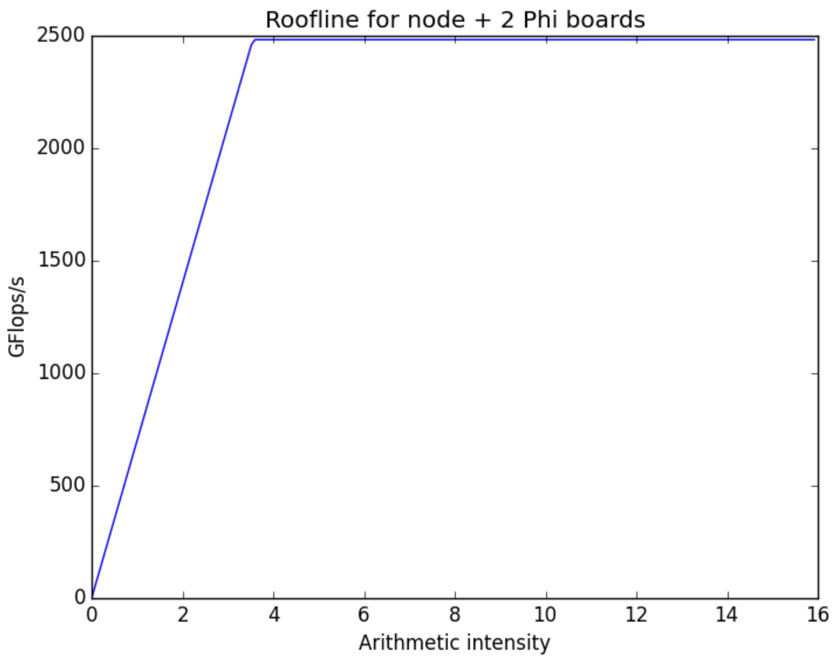

FIGURE 2. Roofline diagram for the compute node + two Phi coprocessors

Question 2. What is the difference between two cores and one core with hyperthreading?

With hyperthreading, there is only one physical core, but multiple threads are scheduled onto the same CPU functional units to give a higher throughput. This allows multiple threads to execute in parallel if they are not sharing the same resources, but is not the same as having multiple physical cores.

Question 3. Do a Google search to find a picture of how memories are arranged on the Phi architecture. Describe the setup briefly in your own words. Is the memory access uniform or non-uniform?

I was able to find a picture of the layout of the Phi architecture here:

`http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-phi-coprocessor-da`

In particular, on page 12. Here we see that the L2 caches are logically shared, but physically distributed, which means the memory access is non-uniform. (At least for the L2 caches).

Question 4. Consider the parallel dot product implementations suggested in the slides. As a function of the number of processors, the size of the vectors, and typical time to send a message, can you predict the speedup associated with parallelizing a dot product computation? [Note that dot products have low arithmetic intensity – the roofline model may be useful for reasoning about the peak performance for computing pieces of the dot product]

First, let p be the number of processors, n the size of the vectors, and α the time to send 8 bytes of data between processors. Let us assume that $n >> p$ and that p divides n for simplicity. Let M be the memory bandwidth of the processors (in doubles per second), and let us suppose that these machines are memory-bound for an arithmetical intensity of 1 flop/double.

Then each processor computes the partial sum of n/p elements, and then communicates these partial sums to each other.

Since our dot product implementation computes one add and multiply for every two accesses to memory, this calculation has an arithmetic intensity of 1 flop/double. Therefore the time for the partial sums is going to be limited by the time for the processor to read in n/p doubles, which is $M * n/p$. Then the time to communicate this is going to be α, for each processor, since each processor sends 8 bytes to the other processors. If the vector is large compared to the number of the processors, the time required to compute the sum of the partial sums will be negligible. Therefore the total parallel time will be:

$$\frac{Mn}{p} + p\alpha$$

since the serial execution time will similarly be Mn. Thus we get a speedup of:

$$\frac{\text{Serial time}}{\text{Parallel time}} = \frac{Mn}{\frac{Mn}{p} + p\alpha} = \frac{Mnp}{Mn + p^2\alpha}$$

So we see that the communication costs increase drastically for large p, but that for small p and large n, the speedup is nearly linear in the number of processors.