September 1st, 2015 Pre-Class Questions

Elliot Cartee

September 4, 2015

Question 1

Each of the Xeon Phi 5110P boards has a frequency of 1.053 GHz with 60 cores

Source:

`http://ark.intel.com/products/71992/Intel-Xeon-Phi-Coprocessor-5110P-8GB-1_053-GHz-60-core`

Each core can compute 16 double precision floating point operations per cycle

Source:

`https://software.intel.com/en-us/articles/intel-xeon-phi-core-micro-architecture`

Each of the 8 nodes has a Intel Xeon E5-2620 v3 processor, which has 12 cores at a base frequency of 2.4 GHz

Source:

`http://ark.intel.com/products/83352/Intel-Xeon-Processor-E5-2620-v3-15M-Cache-2_40-GHz`

The above source also says it uses the AVX 2.0 instruction set. According to the source below, this can compute 16 double precision floating point operations per clock.

Source:

`http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/performance-xeon-e5-v3-advanced-vector-extensions-paper.pdf`

(Note that I had to split the URL into two lines because it is too long for the page)

This means that the theoretical peak flop rate from the accelerators is:

$$\text{(\# of cores)*(cycles per second)*(flops per cycle)}$$

$$= (15 * 60) * (1.053 \text{ GHz}) * (16) = 15163.2\text{GFlop/sec} = 15.2\text{TFlop/sec}$$

And the theoretical peak flop rate from the nodes is:

$$(\#\text{ of cores})*(\text{cycles per second})*(\text{flops per cycle})$$

$$= (8 * 12) * (2.4\text{ GHz}) * (16) = 3686.4\text{GFlop/sec} = 3.7\text{TFlop/sec}$$

Together this comes out to a theoretical peak flop rate of:

$$15.2\text{TFlop/sec} + 3.7\text{TFlop/sec} = 18.9\text{TFlop/sec}$$

Question 2

My machine is a mid-2012 13-inch Macbook Air. It has a 1.8 GHz Intel Core i5 processor with two cores Source:

`http://ark.intel.com/products/64903/Intel-Core-i5-3427U-Processor-3M-Cache-up-to-2_80-GHz`

Since this CPU uses the AVX instruction set, it can execute 8 double precision flops per cycle, So the theoretical peak flop rate is:

$$(\#\text{ of cores})*(\text{cycles per second})*(\text{flops per cycle})$$

$$= (2) * (1.8\text{ GHz}) * (16) = 57.6\text{GFlop/sec}$$

Question 3

Suppose there are t tasks that can be executed in a pipeline with p stages.

Suppose each stage takes time s. Then serial execution takes time $t * p * s$, as each stage is executed independently.

Meanwhile, in a pipeline with p stages, each processor would execute the first stage of each task one at a time, so the last processor would begin working at time $(t-1) * s$ and work for a time $p * s$, thus taking a total time of $(p + t - 1) * s$. Therefore the ideal speedup is:

$$\frac{t * p * s}{(t + s - 1) * p} = \frac{t * p}{t + p - 1}$$

Question 4

Given the list of tasks, the minimum serial time would be the sum of the times to compile each component, which is

$$1 \text{ hour } + 0.5 \text{ hours } + 0.25 \text{ hours } + 0.5 \text{ hours} + 0.5 \text{ hours } = 2.75 \text{ hours}$$

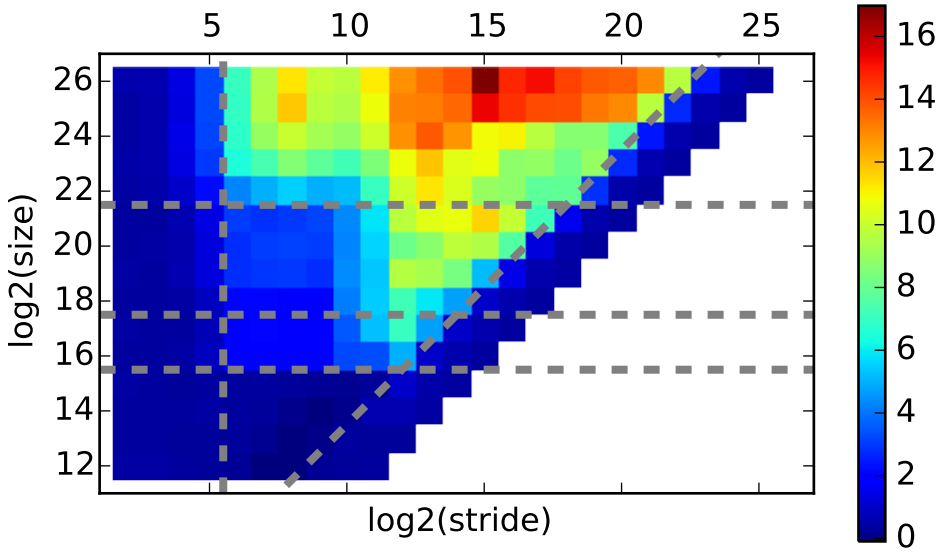
We will now calculate the minimum amount of time given an arbitrary number of processors.

First, we must compile GCC, which will take 1 hour. Then we can simultaneously compile both OpenMPI and OpenBLAS, the latter will take 0.25 hours. Once OpenBLAS is compiled, only then can we compile LAPACK which will take 0.5 hours. OpenMPI will have finished compiling by then. Only once those are all done can the application be compiled which will take 0.5 hours. Therefore the total time in the parallel case is the length of the longest dependency chain:

$$1 \text{ hour } + 0.25 \text{ hours } + 0.5 \text{ hours} + 0.5 \text{ hours } = 2.25 \text{ hours}$$

Question 5

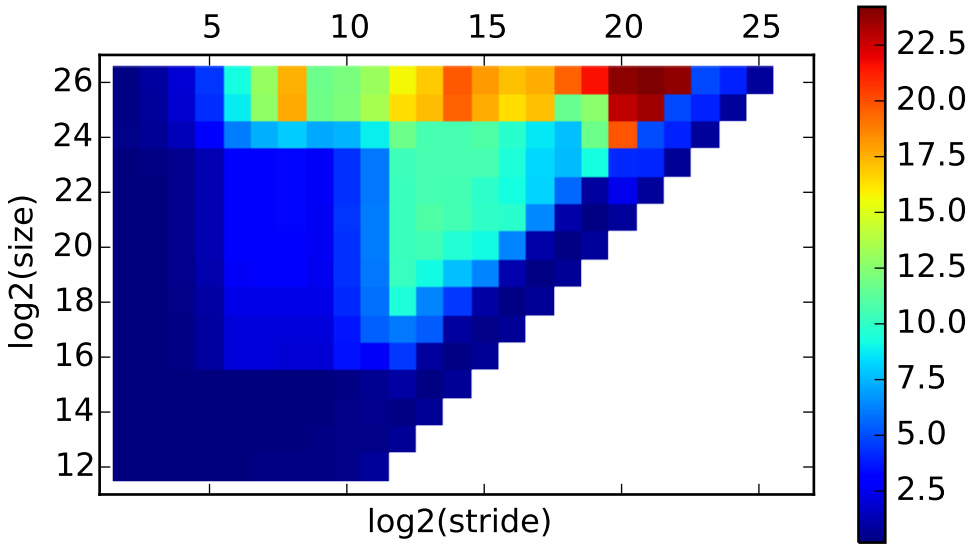
A `membench` heatmap for my machine can be found below and in `timings-heat-mymachine.pdf` in this directory.



Here we can see horizontal lines for the cache sizes, with a 32KB L1 cache, a 256KB L2 cache, and a 3MB L3 cache. You can also see a diagonal line corresponding to the 8-way associativity of the L1 and L2 caches, as well as a vertical line corresponding to the cache line size of 64B.

Question 6

A `membench` heatmap for the totient cluster can be found below and in `timings-heat-totient.pdf` in this directory.



Here we can also see that we have 64B cache lines and 8-way set associativity, but we see that the caches are slightly larger, with horizontal lines at a size of 2^{15} and 2^{24}

Question 7

I'm unable to get `centroid.c` working on the cluster at the moment.