CS 5220 Applications of Parallel Programming — Fall 2015

Kenneth Lim (kl545) — Computer Architecture Basics

1. See previous submission.

2. See previous submission.

3. In only serial execution, i.e. $p = 1$, the total time taken T_s would be $t\tau$, where τ is the time taken per task. In pipelined execution with arbitrary p, the time taken for the very first task to complete is still τ because the pipeline is cold. However, each subsequent task will then come off the pipeline with a takt time of τ/p. Thus the total time taken would be:

$$T_p = \tau + \frac{(t-1)\tau}{p}$$

The speedup is:

$$\frac{T_s}{T_p} = \frac{t\tau}{\tau + \frac{(t-1)\tau}{p}} = \frac{pt}{p+t+1}$$

4. The minimum serial time is 2.75h. The minimum parallel time is 2.25h.

5. See Figure 1.

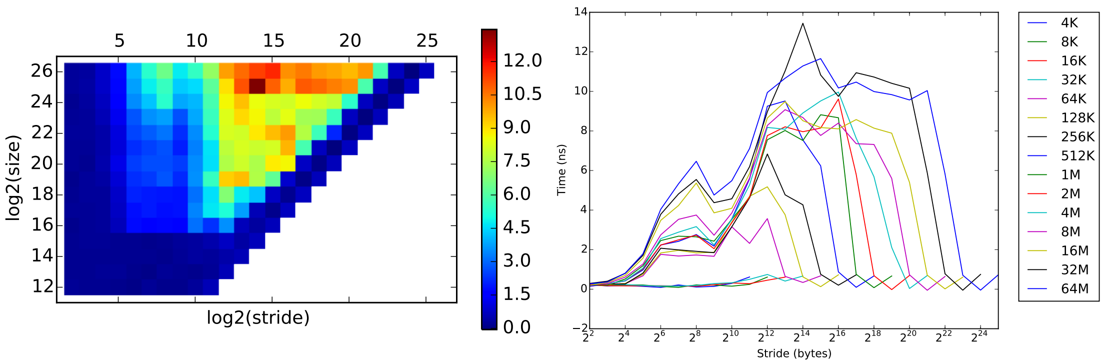

Figure 1: Membench results for the local system

6. See Figure 2.

7. A caveat: when attempting to run `centroid.c` verbatim from the course repo on the cluster, one found that it was consistently giving a timing of 0 for all three functions. The initial hypothesis was that the resolution of the default C timer was insufficient. Following up on Piazza, `centroid.c` was modified to use the OpenMP timer, giving the following results:

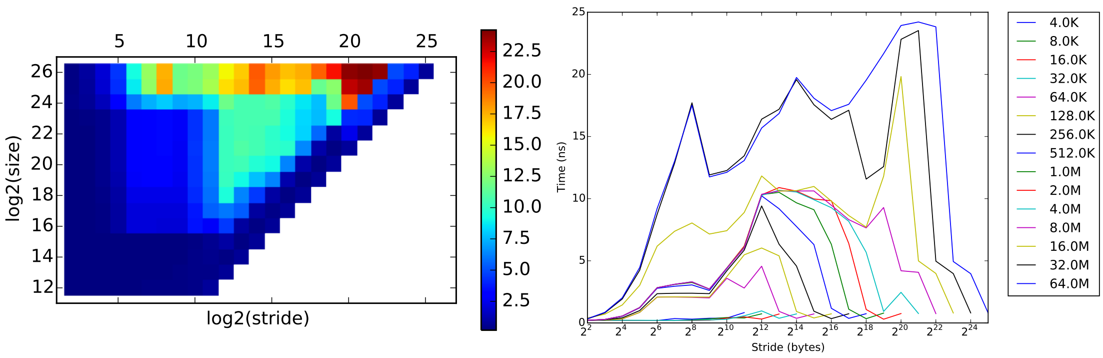

Figure 2: Membench results for the totient cluster

(a) 4.440892e-16 to 1.443290e-15

(b) 4.440892e-16 to 1.443290e-15

(c) 1.332268e-15 to 1.443290e-15

also known as the unfortunate case where the numbers do not really make sense. Theoretically, Implementation B should be the slowest because it does not exploit memory locality or look-ahead, wheres Implementations A and C should be faster because the compiler can optimize for array access from contiguous blocks of memory.