Midterm Examination

Instructor: Randal Burns
EN 600.[3,4,6]20 Parallel Programming

20 October 2021

This is an open book untimed exam. As such, the questions are designed to require thoughtful answers. In some cases, the answers will not be obvious. Some questions may require additional research. You are welcome to use any online resources. You are required to turn in answers that are *solely your own* and prepared individually. You must cite any external resources from which you draw data or use concepts in your answers.

Each question mark indicates that a response is required. Look for directives such as **Explain**, **Define**, or **Give** and follow these instructions. Please keep your answers as brief as possible. Answers that include extraneous facts and irrelevant details will lose credit even if a correct answer is included in the response.

Due: Friday October 22, 2021, 5:00 pm EDT. Submit a PDF of your typeset answers to **Gradescope** `https://www.gradescope.com/courses/311862/assignments/1593510/submissions`. Handwritten answers will not be accepted.

1. (*20 pts*) Figure 2 in `https://arxiv.org/pdf/1706.10086` demonstrates sub-matrix tiling during matrix multiplication. The method assigns an output tile associated with an input row and column of blocks to a thread. You should assume that each tile of all matrixes (A, B, C) are stored independently in their own memory area. The paper describes that they set the tile size at compile time for a specific architecture. It also asserts that "As long as the tiles of the two matrices A, B fit completely in the cache memory, increasing the tile size will usually result in better performance."

   (a) The latest AMD Zen 3 processors (`https://en.wikipedia.org/wiki/Zen_3`) have 32 KB of L1 cache per core, 512 KB of L2 cache per core and a shared 32 MB L3 cache. Based on these specifications, what would you choose as a tile size measured in the number of 64-bit elements in a 2-d tile? Justify your answer.

   Note: You should assume that you are operating on arrays that are much larger than the L3 cache.

   (b) The technique as described in the paper does not require the output tile to be in cache. Why is it not necessary to cache the output tile? Is there a performance benefit from maintaining output data in the cache between sucessive loop iterations? If yes, what unit of data shold be cached? Justify your answer

2. (*15 pts*) In the BlockingQueue implementation of Project 2, we used notifyAll() to wake up all threads. However, only a single String was put on the queue. Why is it not safe to use notify() to wake up a single thread? Give an example.

3. (*10 pts*) Read `https://sgp.fas.org/othergov/doe/lanl/pubs/00326993.pdf`. In class, we said that Amdahl's law is a simplification that models programs into two portions, an optimized portion and an unoptimized portion. We attribute all loss of speedup or parallel inefficiency to the unoptimized portion. This article refines the loss of parallel efficiency.

   (a) In the two-state machine either "all p processors are operating or only one processor is operating." In this model, what factor(s) against parallelism (interference, startup costs, skew) does $1 - a$ represent? Justify your answer.

   (b) They further introduce a degradation term, what factor(s) against parallelism does $\sigma(p)$ represent? Justify your answer.

4. (*15 pts*) Pipelines are one form of instruction level parallelism in modern microprocessors. The Wikipedia page does a nice job presenting a simplified version of the topic–`https://en.wikipedia.org/wiki/Instruction_pipelining`. This question considers a n-stage pipeline as an n resource parallel computer providing up to n-times speedup over a processor executing one instruction at a time to completion before starting the next instruction.

   (a) How does an interpreted language like Python prevent a processor from realizing pipeline parallelism?

   (b) Based on the Wikipedia example in which the "processor determines that decoding depends on results produced by the execution of the green instruction", what factor against parallelism (startup, skew, interference) best matches the loss of parallelism from the pipeline bubble? Justify your answer.

   (c) Looking at experimental values for instruction latencies (`https://www.agner.org/optimize/instruction_tables.pdf`), simple instructions (one operation) can take many clock cycles. This also creates pipeline bubbles. What factor against parallelism arises from long-running instructions? Justify your answer.

   Note: You do not need to read the instructions tables. This is a reference.

5. (*20 pts*) Rewrite the reduction code from stencil.c to perform an efficient reduction without the OpenMP reduce construct. Your code should have (nearly) the same performance properties as the reduction.

```
#pragma omp parallel for reduction ( max: max_el )
for (int x=0; x<DIM; x++) {
    for (int y=0; y<DIM; y++) {
        max_el = max_el > input_ar[x*DIM+y] ? max_el : input_ar[x*DIM+y];
    }
}
```

   Specifically:

```
// TODO any setup code
#pragma omp parallel for
// ** you'd like to put per-thread setup code here, but that won't compile.
for (int x=0; x<DIM; x++) {
    // TODO ** so this is the best you can do for per-thread setup code
    for (int y=0; y<DIM; y++) {
        // TODO you'll probably need to change some stuff in the loop.
    }
}
// TODO any exit code
// at this point, max_el should contain the maximum element
```

   ** This is why your implementation won't be as efficient as OpenMP. You have no access to per-thread setup before the loop runs.

   (a) Rewritten code here.

   (b) What are the important design principles that your code implements?