CpSc 418 **Final Exam** December 4, 2018

Graded out of **100 points**.

Q	Pts
1.	
2.	
3.	
4.	
5.	
Total	

Time for the exam: 2 hours and 30 minutes.

Answer questions on the exam pages. Keep your answers brief – they should fit in the space provided.

Open book: anything printed on paper may be brought to the exam and used during the exam. This includes the textbook, other books, printed copies of the lecture slides, lecture notes, homework and solutions, and any other material that a student chooses to bring.

Calculators are allowed: no restriction on programmability or graphing. There are a few simple calculations needed in the exam, a calculator will be handy, but the fancy features will not make a difference.

No communication devices: That's right. You may not use your cell phone for voice, text, web-surfing, as a calculator, or any other purpose. Likewise, the use of computers, tablets, etc. is not permitted during the exam.

Some questions on this exam have many subquestions. This is to reduce the amount of reading that you need to do to understand the problem set-ups. The subquestions are mostly independent. If you get stuck on a part of a question, look at the subsequent parts – you can probably solve them.

Figures are at the end of the exam. You can rip-off those pages so you can hold the figure next to the question or your solution if you want.

Bug bounties are in effect and given in final-exam points. Only report a suspected error if it affects your ability to complete the exam. To report an error, raise your hand. Due to Mark's hearing difficulties, he may need to step out in the hall to hear you – he doesn't understand whispers. We will post any corrections to the whiteboard.

Minor spelling, grammar, and similar errors should be posted to piazza after the exam. If the "error" does not impact your understanding of what question is being asked of how to solve it, then it is a minor error.

Good luck!

CpSc 418 **Final Exam** December 4, 2018

Graded out of **100 points**.

Five questions.

0. **Who are you?** (2 points)
   (a) What is your name? _____
   (b) What is your student number? _____

1. **Matrix-Multiplication** (20 points)
   Consider matrix multiplication, $Z = X\,Y$ on a message passing machine with P processors. Let X, Y, and Z be $N \times N$ matrices. Furthermore, we assume that P is a perfect square and that N is a multiple of P. Each matrix has N^2 elements with N^2/P elements stored on each processor. In particular, we will have each processor store a $\frac{N}{\sqrt{P}} \times \frac{N}{\sqrt{P}}$ submatrix (i.e. "tile") of each of the X, Y, and Z matrices. We will write $\widehat{X}_{I,J}$ to indicate the submatrix of X that is stored on processor $p(I, J)$

$$\widehat{X}_{I,J}(i,j) \quad = \quad X((N/\sqrt{P}) * I + i, (N/\sqrt{P}) * J + j), \quad 0 \le I, J < \sqrt{P},\ 0 \le i, j < N/\sqrt{P}$$

   and likewise for Y and Z.

   (a) **(2 points)** How many multiply-add operations are required to compute the product of two $N \times N$ matrices? Your answer should be a formula using the variable N.

   (b) **(2 points)** How many multiply-add operations are required to compute the product of two $\frac{N}{\sqrt{P}} \times \frac{N}{\sqrt{P}}$ matrices?
   Your answer should be a formula using the variables N and/or P.

   In a parallel implementation:

   - Each processor, $p(I, J)$, sends its submatrix, $\widehat{X}_{I,J}$, to each processor in its "row", i.e. to processors $p(I, 0)$, $p(I, 1), \ldots, p(I, \sqrt{P} - 1)$. Likewise, $p(I, J)$, sends its submatrix, $\widehat{Y}_{I,J}$, to each processor in its "column": i.e. to processors $p(0, J), p(1, J), \ldots, p(\sqrt{P} - 1, J)$.
   - After these matrices have been sent (and received), each processor, $p(I, J)$, has matrices $\widehat{X}_{I,K}$ and $\widehat{Y}_{K,J}$ for $0 \le K < \sqrt{P}$.
   - Each processor, $p(I, J)$, computes

$$\widehat{Z}_{I,J} \quad = \quad \sum_{K=0}^{\sqrt{P}-1} \widehat{X}_{I,K}\, \widehat{Y}_{K,J}$$

Assume that the matrix-adds can be done as part of the fused multiply-adds for the matrix multiplications (they can).

(c) **(4 points)** How many fused multiply-adds does each processor (e.g. $p(I, J)$) perform to compute its part of the product (e.g. $\widehat{Z}_{I,J}$)?
Your answer should be a formula using the variables N and/or P.

(d) **(4 points)** If we ignore communication time, then the time to compute $X\,Y$ is just the time for the fused multiply-adds. Ignoring communication time, what is the speed-up when computing the product $Z = X\,Y$ using P processors?
Your answer should be a formula using the variables N and/or P.

(e) **(4 points)** Now, consider communication time. Assume that it takes time $\lambda + Mt_0$ to send (and receive) a message of M matrix-elements from one process to another process. With the algorithm described above, each processor sends (and receives) $\sqrt{P} - 1$ messages to (and from) the other processors in its row, and another $\sqrt{P}-1$ messages to (and from) the other processors in its column. Each of these messages conveys one of the $\widehat{X}$ or $\widehat{Y}$ matrices. What is the time spent to send (and receive) messages? Your answer should be a formula using the variables N, P, λ, and/or t_0.

(f) **(4 points)** If the communication time is less than or equal to the computation time, we'll assume that the communication can be fully overlapped by the computation. Let $\lambda = 10000$ and $t_0 = 10$, and $P = 64$. How large must N be for the communication time to be roughly the same as the computation time? Your answer should be a number.

Hint: You don't need to figure out the exact answer. 75% credit if you work out the equation in N to solve no matter what your value is for N. 100% credit if you work out the equation in N, estimate N to within a factor of 2, and give a one or two-sentence justification for your answer.

2. **Convolution** (40 points)
Often, when computing the convolution of sequences x and y, it is convenient to think of x as being a sequence with 0-based indices: $x = x_0, x_1, \ldots, x_{n-1}$, and y being a sequence with indices *centered* at 0: $y = y_{-k}, y_{-(k-1)}, \ldots, y_{-1}, y_0, y_1, \ldots, y_k$. Let $z = x\#y$ denote the convolution of x and y, and we have:

$$z_i = \sum_{j=-k}^{k} x_{i-j} y_j$$

where we treat x_{i-j} as 0 if $i - j < 0$ or $i - j \geq n$.

Although z_i could have non-zero elements for any i with $-k \leq i < (n-1) + k$, we often just want the values of z_i for $0 \leq i < n$. That way, z is an array of the same size as x. Figure 1 shows a CUDA implementation of this convolution computation.

Simple Example:

(a) **(2 points)** Let $n = 5$, $k = 1$, $x_i = i$, $y_{-1} = -1$, $y_0 = 2$, $y_1 = 1$. Let $z = x\#y$.

i. What is z_0?

ii. What is z_2?

Control flow:

(b) **(2 points)** What is the purpose of the `if`-statement on line 6 in Figure 1? In other words, what could go wrong if that test were omitted?

(c) **(4 points)** What is thread divergence? Give a short (i.e. two or three sentence) explanation.

(d) **(4 points)** Give two examples of statements that could cause thread divergence in Figure 1. For both of those statements, give conditions that cause the divergence – for example,

"If <state values for some variables>, then <state what happens>"

__syncthreads:

(e) **(4 points)** What does the `__syncthreads()` function do? Give a short (i.e. two or three sentence) explanation.

(f) **(4 points)** The code in Figure 1 needs a call to `__syncthreads()`. State where the call should be added and give a short explanation (no more than five sentences).

Memory:

(g) **(1 points)** Is the reference to `x[myId]` on line 8 in Figure 1 a reference to global memory or shared memory?

(h) **(3 points)** If the reference to `x[i]` on line 8 in Figure 1 is a reference to global memory, is it a *coalesced* reference? If it is a reference to shared memory, does it cause a bank-conflict? Give a one or two sentence justification for your answer.

(i) **(1 points)** Is the reference to `y_sh[i+k]` on line 19 in Figure 1 a reference to global memory or shared memory?

(j) **(3 points)** If the reference to `y_sh[i+k]` on line 19 in Figure 1 is a reference to global memory, is it a *coalesced* reference? If it is a reference to shared memory, does it cause a bank-conflict? Give a one or two sentence justification for your answer.

Kernel launch:

(k) **(3 points)** Complete the kernel launch on line 32 in Figure 1. In particular, write in your the CUDA code that should replace `your_answer(Question2k)` here:

CGMA:

(l) **(3 points)** What is the CGMA for `conv_kernel` when `n=1024` and `k=4`?

(m) **(3 points)** Write a formula for the CGMA for `conv_kernel` in terms of `n` and `k`.

(n) **(3 points)** What is the CGMA for `conv_kernel` when

i. `n=1024` and `k=90`?

ii. `n= 256` and `k=90`?

3. **Sorting on a GPU** (15 points) Consider sorting on an array of `floats` on a GPU using the bitonic sort algorithm. For its artistic value, Figure 2 shows the sorting network for bitonic sort with 8 inputs.

   (a) (**1 point**) How many compare-and-swap operations are performed by the 8-input bitonic sort?

   (b) (**1 point**) What is the span for the 8-input bitonic sort?

   (c) (**2 points**) How many compare-and-swaps are performed by a N-input bitonic sort? Check one of the following:

   - ☐ N
   - ☐ $N \log_2 N$
   - ☐ $\frac{N}{4}(\log_2 N)(1 + \log_2 N)$
   - ☐ $\frac{N}{2}(\log_2 N)^2$
   - ☐ $\frac{N^2}{2}$
   - ☐ $N!$

   (d) (**4 points**) A compare-and-swap of `x` and `y` can be performed as `min(x,y)` and `max(x,y)`. Assume that `min(x,y)` counts as a single floating point operation, and likewise for `max(x,y)`. Thus, a compare-and-swap counts as two floating point operations. How many floating operations are performed by bitonic sort for an input with 2^k values?

   (e) (**2 points**) If we load 2^k values from the global memory into shared memory, sort them, and then write the sorted result back to the global memory, what is the total number of global memory accesses performed by the sort?

   (f) (**2 points**) On a GPU, we will store the data to be sorted in the shared memory of an SM. A block can have at most 48 Kbytes of shared memory, and a `float` has a size of 4 bytes. What is the largest integer k such that 2^k `floats` can be stored in the shared memory of one SM?

(g) (**2 points**) For the value of k from your answer to Question 3f, what is the CGMA?
Note: this is the CGMA for a single block. We could perform a sort of more than 2^k elements by sorting groups of 2^k as described here, and launching a subsequent kernel or two to merge those results. That would make this question way more complicated Furthermore, the CGMA for a single block is pretty close to the CGMA for the entire computation – adding more blocks scales up the number of floating point operations and the number of memory accesses at the same rate.

(h) (**3 points**) For GPUs like the GTX 1060s in the `linXX` machines, is the critical bottleneck for bitonic sort memory bandwidth or the number of computations performed? Is it likely that another algorithm will be much faster than bitonic sort when run on a GPU? Give a short justification for your answer.

4. **Bitonic Sequences** (10 points)

(a) (**6 points**) Let $n > 0$, and let $x_0, x_1, \ldots, x_{n-1}$ be a bitonic sequence of n 0s and 1s. You can assume that x is of the form $0^*1^*0^*$. Define y as the sequence with:

$$\begin{aligned} y_0 &= x_{n-1} \\ y_i &= x_{i-1}, \quad 1 \leq i < n \end{aligned}$$

Prove that y is bitonic.

(b) (**4 points**) Let $x_0, x_1, \ldots, x_{n-1}$ be a bitonic sequence of n numbers, and let y be defined as in Question 4a. Show that you can choose x such that y is not bitonic.

5. **Other Stuff** (15 points)

(a) Fused multiply-add **(4 points)**

i. **(1 point)** What is a fused multiply-add? Give a one or two-sentence definition.

ii. **(1 point)** When counting the number of floating point operations performed by a computation, how many floating point operations is one fused multiply-add?

iii. **(1 point)** Give an example of an expression consisting of one multiply and one add where the operations **can** be combined into a single fused multiply-add operation.

iv. **(1 point)** Give an example of an expression consisting of one multiply and one add where the operations **cannot** be combined into a single fused multiply-add operation.

(b) **(6 points)** Consider a problem whose sequential version takes time $N^{3/2}$. A parallel implementation running on P takes time time $\frac{N^{3/2}}{P} + (\sqrt{N} + \lambda) \log_2 P$. Let $N = 10^6$ and $\lambda = 10^4$. Calculate the speed-up for

i. $P = 2^6 = 64$

ii. $P = 2^{10} = 1024$

iii. $P = 2^{14} = 16384$

(c) True or false **(4 points)**

_____ If a cache line in the MESI protocol is in state M, then no other cache has data for this address.

_____ The cross-section width of a $N \times N$ 2D-mesh is $\Theta(N \log N)$.

_____ Using energy scaling, it is possible to have a parallel algorithm that performs more operations in less time and using less energy than the best sequential algorithm.

_____ All blocks of a CUDA grid must execute at the same time.

(d) Confused multiply-add **(1 point)**.
Give an example of a confused multiply-add. Extra credit will be considered for creative answers.

```
 1:     __shared__ float x_sh[1024];
 2:     __shared__ float y_sh[1024];

        // convolution kernel
 3:     __global__ void conv_kernel(uint n, int k, float *x, float *y, float *z) {
 4:        uint myId = blockDim.x*blockIdx.x + threadIdx.x;
 5:        uint k2p1 = 2*k+1;
 6:        if(myId < n) {
 7:           // load x
 8:           float xx = x[myId];
 9:           x_sh[myId] = xx;
10:           // load y
11:           if(myId < k2p1) {
12:              float yy = y[myId];
13:              y_sh[myId] = yy;
14:           }
15:           float sum = 0.0f;
16:           for(int i = -k; i <= k; i++) {
17:              int ii = myId - i;
18:              if((0 <= ii) && (ii < n))
19:                 sum += x_sh[ii] * y_sh[i+k];
20:           }
21:           z[myId] = sum;
22:        }
23:    }

        // wrapper function to call from C
24:    void conv(uint n, int k, float *x, float *y, float *z) {
25:       size_t sz_x = n*sizeof(float);
26:       size_t sz_y = k*sizeof(float);
27:       cudaMalloc((void**)(&dev_x), sz_x);
28:       cudaMalloc((void**)(&dev_y), sz_y);
29:       cudaMalloc((void**)(&dev_z), sz_x);
30:       cudaMemcpy(dev_x, x, sz_x, cudaMemcpyHostToDevice);
31:       cudaMemcpy(dev_y, y, sz_y, cudaMemcpyHostToDevice);
32:       conv_kernel<<<your_answer(Question2k),1024>>>(n, k, dev_x, dev_y, dev_z);
33:       cudaMemcpy(z, dev_z, sz_x, cudaMemcpyDeviceToHost);
34:       cudaFree(dev_x);
35:       cudaFree(dev_y);
36:       cudaFree(dev_z);
37:    }
```

Figure 1: CUDA implementation of convolution

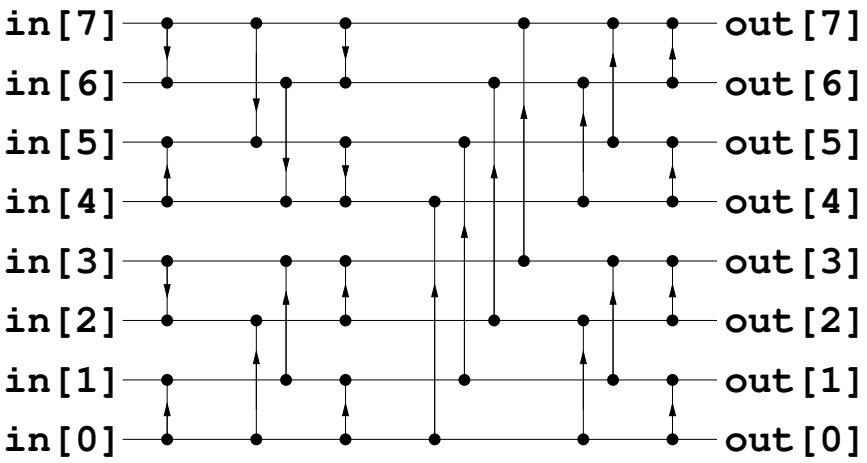

Figure 2: Bitonic Sort with 8 inputs