This survey collects your answers for the PIKA on Kirk & Hwu chapter 9 in CPSC 418 2017-2. In order to receive PIKA credit, you must enter your name and student number below.

Please note that this survey may have multiple pages. Please keep going until you see the completion message.

Enter your last name (as it appears in Connect).

Enter your student number.

Let *n* be the number of input elements and *m* be the number of histogram bins. The most efficient serial implementation of histogram has complexity:

- O(m)
- O(lg n)
- O(lg m)
- O(nm)
- O(n+m)
- O(m lg n)
- O(1)
- O(m^2)
- O(n)
- O(n^2)
- O(n lg m)

Let *n* be the number of input elements and *m* be the number of histogram bins. The most efficient serial implementation of histogram requires storage (including input, temporary and output data):

- ○ O(m)
- ○ O(nm)
- ○ O(lg n)
- ○ O(n)
- ○ O(n lg m)
- ○ O(1)
- ○ O(n^2)
- ○ O(lg m)
- ○ O(n+m)
- ○ O(m lg n)
- ○ O(m^2)

We have not previously used atomic memory operations because (check all that apply):

- ☐ The risk of radioactive debris should our code crash the GPU.
- ☐ Atomic memory operations can only be accomplished by intrinsics and we have not yet used intrinsics.
- ☐ In our previous GPU problems each thread was assigned an independent subset of the input data at each step.
- ☐ In our previous GPU problems each thread was assigned an independent subset of the output data at each step.
- ☐ Atomic memory operations are an order of magnitude slower than even global memory reads and writes.
- ☐ In our previous GPU problems we could fit the output data into shared memory (for example, by tiling) and hence atomic memory operations were unnecessary.

→

For the questions on this page, imagine solving a histogram problem with 2^20 = 1048576 input elements (assume each input element requires 4 bytes) and 2^6 = 64 histogram bins (assume the count for each bin is stored in 4 bytes). Consider using a grid with 2^7 = 128 blocks each with 2^8 = 256 threads to compute a solution.

Write your answer in the form of an integer: write "8" not "2^3".

In order to balance the workload of each thread, how many input elements should be assigned to each thread?

In order to balance the workload of each thread, how many output elements should each thread be able to write to?

In order to achieve coalesced global memory reads, how far apart should the input elements be (in bytes) that we assign to thread 0 of block 0?

How much memory is required for a basic CUDA implementation of this problem, such as that shown in K&H figures 9.6 or 9.8? Do not count register or cache memory, but do count shared, constant and/or global memory. Include input, temporary and output data in your calculation.

How much memory is required for a CUDA implementation of this problem using privatization, such as that shown in K&H figure 9.10? Do not count register or cache memory, but do count shared, constant and/or global memory. Include input, temporary and output data in your calculation.

Assume (very optimistically) that you have sufficient SPs to run all of the threads in the grid in parallel and that the atomic sum operations are perfectly evenly distributed among all of the histogram bins. According to Amdahl's law, what is the maximum speedup possible over a serial version implemented on the same hardware (so you may assume that memory operations take the same time on the serial or parallel versions). Assume that you are using the basic CUDA implementation, such as that shown in K&H figures 9.6 or 9.8.

→