Answers.md

CO332 - Heterogenous Parallel Computing

Assignment 2


```
Sagar Bharadwaj - 15CO141
Aneesh Aithal   - 15CO107
```


NOTE For questions asking "FLOPS per kernel", "Number of reads per kernel" etc it has been assumed that a kernel refers to the opeartions happening in a single thread.

Q1

1. Tried using shared memory to reduce global memory accesses.
2. It was not possible to store 4096 bins in shared memory
3. Code uing two dimensional thread blocks gave maximum benefit
4. *1* global memory read for reading input value.
5. 1 write to increment histogram value. 0 if `BIN_CAP` is reached.
6. Per kernel, *1* atomic add for incrementing histogram value. *0* if `BIN_CAP` is reached.
7. The kernel execution will become very slow due to every thread trying to write in the same location. Hence, it will be akin to sequential execution as atomic operation is done
8. The program will run optimally because of the random distribution. Hence, atomic add will cause least number of thread locks.

Q2

1. Analysing the problem to realise where `__syncthreads()` should be used and also where `atomicAdd()` to be used was a challenge.
2. First optimisation was to parallelise the sequential code using global memory. Secondly shared memory was used to create privatized histograms. However tranferring this data to global memory was done sequentially. Finally `threadIdx.x` was used to parallelise transferring privatised histogram to global memory using `atomicAdd()`. The last optimisation was most beneficial.
3. *1* global memory read is performed per thread. Total of *n* global memory reads, where n is the *number of ASCII characters*.
4. *1* global memory write is performed per thread.
5. *Two* atomic operations are being performed per thread. One atomic operation updates the privatised histogram bins maintained in the shared memory. The second one transfers the shared memory's histogram data to global memory.
6. The algorithm gives a poor performace in this case. Atomic access contention of privatised histogram bins would yield a result similar to the serial counterpart of the algorithm.

Q3

1. Yes. When declaring the device vector, the host array pointer is passed as an argument to be copied.
2. O(n * logn) due to sorting. (n = size of array)
3. Worst case for atomic approach is when all elements in the array have the same value. Hence, worst case is O(n). Complexity of thrust approach is O(n * logn).
4. Because doing it serially is probably faster due to the time being wasted to avoid race condition.

Q4

1. Applications of Convolution :

- Convolutional neural networks apply multiple cascaded convolution kernels.
- To add blurring effect in images
- Edge detection in images

2. Per kernel, 2 operations per pixel in the mask (add and multiply). Also, there is a single multiplication being done during clamp. Hence, 5 x 5 x 2 + 1 = **51**. 5 is mask radius.
3. 1 per kernel. 1 to read pixel value and store it in shared memory.
4. 0/1 per kernel. 1 write to store value after applying mask. 0 if the thread if outside the tile but inside the block.
5. Floating point operations are real operations here. Minimum is 0. Maximum is 50. There are TILE_WIDTH x TILE_WIDTH threads where 50 operations are done and in (BLOCK_WIDTH x BLOCK_WIDTH) - (TILE_WIDTH x TILE_WIDTH) kernels, it is zero. Hence, the average is:
   (50 x TILE_WIDTH x TILE_WIDTH) / BLOCK_WIDTH x BLOCK_WIDTH.
6. *Size of Input = M X N*
   CPU application takes a runtime of O(M * N). GPU application has a Parallel complexity of O(1) with CPU overhead of(M * N).
7. The overhead is a couple of milliseconds. It increases with input size because of the time taken to read the input matrix and transfer it to the GPU memory. Overhead for transferring back is also higher.
8. If mask width is 1024, a couple of megabytes is required to store it. Hence, it can't be stored in constant memory. This means that every single kernel has to read the mask from global memory. This causes a huge slowdown in kernel execution. You may be forced to read the mask in batches to use the constant memory approach.
9. Because changing it in place will modify the values of a pixel before it is used in another kernel in another block.
10. A square matrix of zeroes except the middle element which is 1.

Q5

Dimension of Matrix : width X height X depth

1. Approximately, for every 3 X 3 X 3 tile, 5 X 5 X 5 elements are read from global memory.
   Therefore, total global memory reads = **(width * height * depth)/27 * 125**
2. Approximately, every element (except border) would read 7 elements (including itslef) from shared memory.
   Therefore, total shared memory reads = **7 * (width-2) * (height-2) * (depth-2)**
3. (No Question)
4. Our code makes fewer global memory accesses than a 3 X 3 convlution code would. However it makes more shared memory accesses.