CMPUT 382 Wi20 - INTRO TO GPU PROGRAMMING Combined LAB LEC Wi20

Lab 5 tips

Summary the problem:

- There are N inputs values. All of them will be from 0 to (NUM_BINS - 1)

- Your task is build a histogram of the input values. The histogram has NUM_BINS bin, each of the bin correspond to a number.

- Finally, do a post-processing task: if a bin value is greater than 127, make it 127.

- You may need to syncthread between these two steps to make sure the program work correctly

```
    // Naive version:
    // Divide the input to numThreads consecutive partitions. Each thread will handle 1 partitions.
    // No coalesced memory because each thread in a warp will access memory that is very far from the others

    // Non privatized version: 80%
    // all thread will handle a consecutive portion of input
    // on bigger picture: each thread will handle input in strided fashion: ie thread 1 will handle postion 1, 1 + numThreads, 1 + numThreads * 2 ...
    // each thread will write back to global memory bin value accordingly to the inputs that it handles
    // need atomic operation
    // strides pattern is to make sure access to global memory is coalesced
    // each time the GPU will load 128 consecutive bytes at the same time
    // The simple rule (on Fermi and later GPUs) is that if the addresses for all threads in a warp fall into the same aligned 128-byte range, then a single memory transaction will result (assuming caching is enabled for the load, which is the default). If they fall into two aligned 128-byte ranges, then two memory transactions result, etc.
    // that's why make all the thread access nearby memory
    // lot of global memory access

    // Privatized bins: 20%
    // all thread in a block share a privatized histogram with the number of bins equal to in the global memory
    // first initlize privatized histogram of block to 0
    // then each thread handle the strided pattern the same as in the non privatized version but instead of writing to global memory it writes to privatized block histogram
    // still need atomic operation for this write
    // after done calculating, thread in each blocks will add the value of privatized bins to the global histogram using atomic operations
    // you may need to allocate shared memory dynamically for your kernel. use "extern __shared__"
    // extern __shared__ mean shared array that initialized dynamically (ie array size is determined at kernel launch instead of compile time)
    // can only have 1 extern __shared__ in 1 kernel
    // need to put the allocation size of extern __shared__ at kernel launch code
```