Alexander Cook

Programming on Massively Parallel Systems

Project 3 – Parallel Radix Partition

Description of Implementation:

Algorithm 1:

Creating the histogram for radix partitioning is a far simpler process than the histogram in the SDH algorithm. Each datum from the input array is read only once so there is no benefit to using shared memory here. However, using shared memory for the output is very important. The number of buckets may be as low as two, so the benefit of output privatization is high.

Algorithm 2:

For this algorithm I referenced the lecture slides on prefix scans as well as a relevant Nvidia developer article[1]. The method I used attempts to improve on the naïve approach which yields O(nlogn) due to redundant additions. It does this by splitting the process into two steps, a downsweep (reduction) and upsweep. This yields O(2n-1). The kernel also uses shared memory for the array that it performs these "in place" operations as it reads and writes the same data multiple times.

This implementation has major limitations. It only works for input data in powers of two and runs with 1 block of at most 1024 threads. However, this is not a problem in this application.

Algorithm 3:

Like algorithm 1, algorithm 3 does not read the input multiple times. It must reference the prefix sum array while updating its value many times. I predicted shared memory would be beneficial here, but it did not improve performance, so I have omitted it.

Alexander Cook

Testing results:

For testing I am comparing algorithm 1 with and without privatized output.

Input size	#Partitions	Average Difference
1024	2	0.007333333
1024	32	0.009666667
1024	1024	-0.003
100000	2	0.069
100000	32	0.038666667
100000	1024	-0.170666667
1000000	2	0.77
1000000	32	0.276666667
1000000	1024	-1.556

Yellow is low difference in performance, red is worse, and green is improvement.

As expected, when the number of partitions is small the number of threads competing for global memory access is very high, resulting in serial access and slower performance. For 2 partitions shared memory use is very low, and it worth using. For 1024 partitions the race condition is much less a problem. In the case of large input it actually makes things much worse because the amount of shared memory needed is high.

Sources:

[1]https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch39.html