EE 5885 GPGPU Exam 2 Take Home

Josh Blaney

December 3, 2022

1 Performance Data

Table 1: Execution Times

B	Z	CPU (msecs)	Multiple (usecs)	Single (usecs)	Add (usecs)	GPU Total (usecs)	Speedup
256	64	0.0329	4.13	3.71	3.1	10.94	3.01
256	128	0.0558	5.12	3.74	3.2	12.06	4.63
256	256	0.1228	5.29	4.26	3.42	12.97	9.47
512	128	0.1185	6.27	3.97	3.39	13.63	8.69
512	256	0.2298	9.09	4.29	3.78	17.16	13.39
512	512	0.3151	13.63	5.02	4.64	23.29	13.53
1024	256	0.3134	20.06	4.42	5.31	29.79	10.52
1024	512	0.641	32.93	5.09	7.65	45.67	14.04
1024	1024	1.3391	59.65	6.69	13.38	79.72	16.80

From Table 1 it can be seen that the speedup offered by the GPU program increases with increasing data size. It can also be seen that for small data sizes the execution time of all kernels is similar but diverges with increasing data sizes. Specifically, the Multiple Block Scan kernel (Multiple from Table 1) requires most of the computation time for this algorithm because it is doing most of the computations required to produce the result; whereas, the Single Block Scan kernel (Single from Table 1) operates on a significantly reduced size array limited to the number of blocks used to segment the data and the Add Block Scan kernel (Add from Table 1) operates on array size minus block size elements in parallel only once.

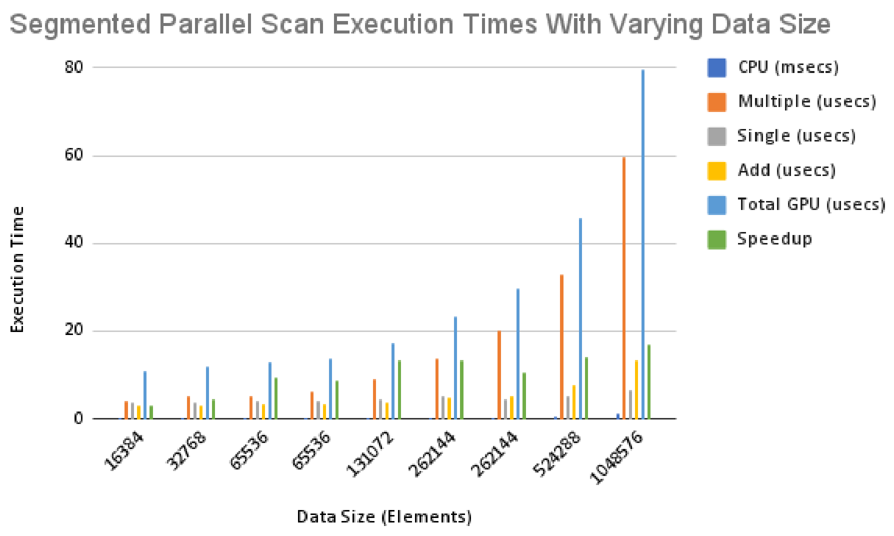

Figure 1: Chart of Execution Times For Sequential and Parallel Segmented Reduction

It can be seen from Figure 1 that the total GPU execution time and the sequential execution time follow similar trends with increasing data size. It can also be seen that the speedup for similar data sizes is greater when the block size (B from Table 1) is small and the integer multiplier (Z from Table 1) is large.

Table 2: GPU Precision

B	Z	Average Precision
256	64	1.0E-4
256	128	1.0E-4
256	256	1.0E-3
512	128	1.0E-3
512	256	1.0E-3
512	512	1.0E-2
1024	256	1.0E-2
1024	512	1.0E-2
1024	1024	1.0E-2

From Table 2 the average precision seen when running this program starts at $1.0E-4$ when the dataset is small and decreases as the dataset is expanded. One reason for the loss in precision as the dataset size increases is out of order execution inherent in parallel programs. Because computer memory is a physical entity and is limited in its scope, all numbers can only be stored with a finite precision. The finite precision in computer memory makes it such that if the numbers currently being stored require more precision than the system allows there will be overflow and the number will be truncated to the closest storable value. This truncation may become a problem when adding a significant number of data elements because the system will only be able to store up to a certain precision and the rounding error will compound with each addition. Thus, if it is assumed that the sequential programs answer is correct the rounding error generated when adding the data elements in the sequential order is also assumed to be correct and when the parallel program executes each scan block will have its own rounding error which will not necessarily be equivalent to the sequential rounding error for the same data. Therefore, as the number of data elements increases the rounding error should also increase for both the sequential and parallel programs but the two errors may not be equivalent due to out of order execution in the parallel program.