Casey Adams, Kevin Connell
ECEC 622
Project 7- CUDA Counting Sort
3/17/2021

CUDA: Counting Sort

Implementation of CUDA

First the necessary allocations are made for both the CPU and GPU. The number of threads are decided so that too many aren't used but should the number exceed 1024, multiple blocks will be created to accommodate larger amounts. Then the kernels are run using atomicAdd() so that while adding to the same bin no Read-Write overlaps and causes an error. From there the threads are synced and the inclusive prefix is created using only one thread and a for loop that adds the previous histogram number to the current one. Then a second GPU call places all the correct values into the sorted array using the inclusive prefix. Then all the memory is freed.

Performance

Table 1 shows the seconds it takes to perform count sort. The values are measured in seconds with the columns representing number of elements and the rows representing devices.

Number of Elements vs Device

	10^6	10^8
CPU	0.002314s	0.20427s
GPU	0.1285s	0.6836

Conclusion

The overhead of talking to the GPU is too much at these lower values. However should the trend hold upwards, at numbers such as 10^10 elements, the gpu should be speedier. Given that an increase of 100 times resulted in only an 5x increase for the GPU vs a 100x increase for the CPU.