CSCI4320 HW2 Performance Report

Zhiqi Wang

February 17, 2023

1 Implementation of CLA Adder Functions as CUDA Kernel Calls

There are few things I've modified to make this cla adder parallel;

1. I allocated the memory for all the input, generate, propagate, and carry arrays on the device using `cudaMallocManaged`. This allows the arrays to be accessed by both the host and the device.
2. For the first function calculation for generate and propagate `compute_gp`, I removed a layer of for loop and let each thread handle one element of the array. This is done by setting the index at `threadIdx.x + blockIdx.x * blockDim.x`. Then I just need to launch this kernal with `bits / threadPerBlock + 1` blocks and `threadPerBlock` threads per block. I also did the same modification for `compute_sum`.
3. For all other calculation for generate and propagate, I removed a layer of for loop and let each thread handle one cla block of work. This is done by the same indexing method. I also didn't use the `grab_slice` as cla-serial did. Instead, I used the fact that each `slice` is corresponding to a block of work. So instead of having those slice grabbed and allocated, I just used pointer arithmetic to access the original array.
4. For compute carrys, the carry for super super section is not parallelizable since we don't have a higer level carry to split the work. So I just let one thread to handle this. For the rest of the carry, I reduced the for loop to each carry to a for loop that handles one block of work since the dependency of carry within a cla-block.

2 Performance Comparison

Here is the performance comparison between serial CLA adder and serial rca.

Method	Cycles to Complete
CLA	2745347322
RCA	217716686

Table 1: Comparison of CLA and RCA Methods

I also measured the number of cycles for the CUDA CLA adder function in my cla-parallel program for different block sizes as noted below: 32, 64, 128, 256, 512, and 1024. The results are shown in Table 2.

Thread	CLA Cycles	RCA Cycles
32	78795402	226816092
64	67399818	233531768
128	67807188	232415664
256	77141881	227556736
512	71873059	225999606
1024	46014563	235561807

Table 2: Cycles consumed by CLA and RCA in serial with different thread configurations

Based on the result above(Table 2) the best block size is 1024. There could be two possible reasons for it:

First, shared memory is accessable within on cuda block but not across blocks. Having a larger block size means that we have more threads within a block and more threads can access the shared memory at the same time. This could reduce the number of memory access and increase the performance.

Second, there could be overhead caused by launching more cuda blocks.

3 Speedup Comparison

Now, we are going to compute the speedup for using CUDA CLA with serial CLA and serial RCA. The speedup of CUDA CLA to serial CLA is 2745347322/46014563 = 59.6. The speedup of CUDA CLA to serial RCA is 217716686/46014563 = 4.7.

I abserve that the speedup of CUDA CLA to serial CLA is much larger than the speedup of CUDA CLA to serial RCA. And it seems like serial RCA is faster than serial CLA in this case. CUDA CLA is 59.6 times faster than serial CLA. I think this is because we didn't implement CLA in hardware level, meanwhile RCA is just a simple adder in serial. Since this version of CLA is not parallel, it is not as fast as RCA. And we can't enjoy the benefit of it.