Lab6: CUDA Advanced

助教: 鄭禎<zzchman@gmail.com>

2017.07.27

Outline

1. Target
2. Problems and Solutions

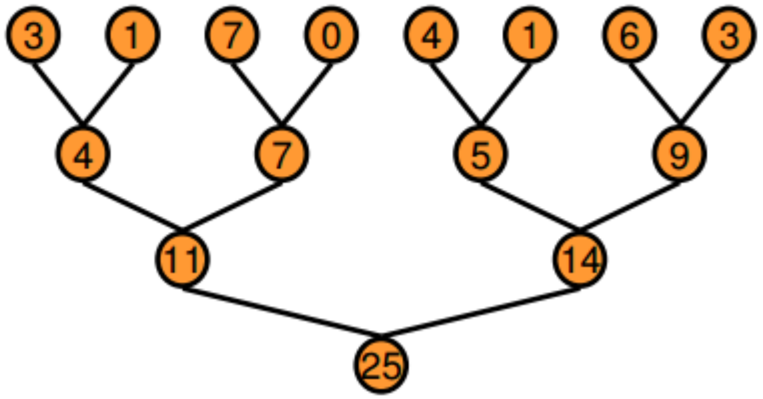
Outline

1. Target
2. Problems and Solutions

Target

Implement parallel reduction with CUDA.

- Use multiple thread blocks, each of which reduces a portion of the array.
- Be able to process very large arrays.



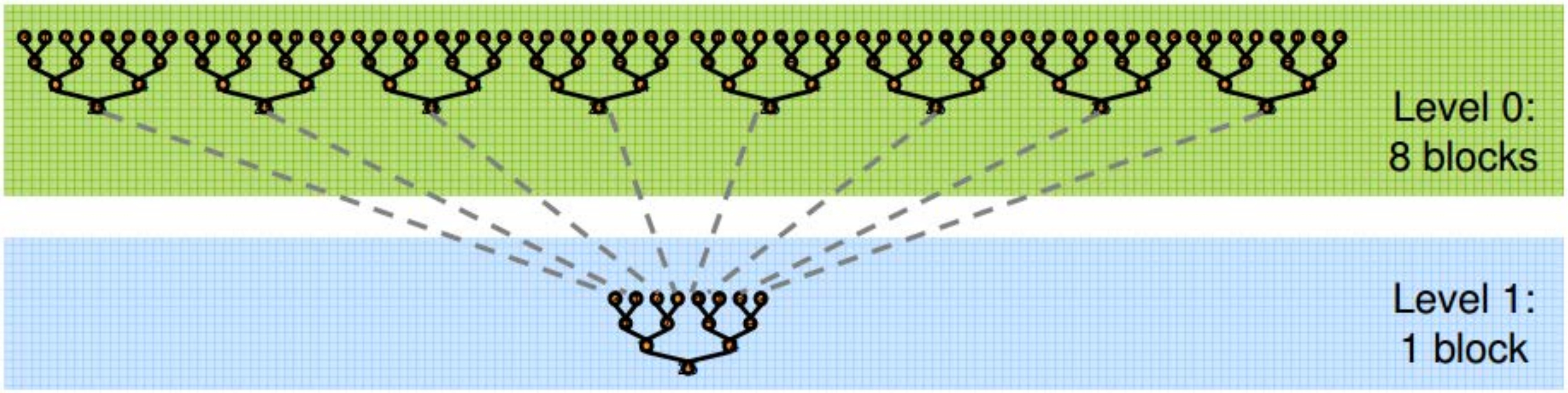
Outline

1. Target
2. Problems and Solutions

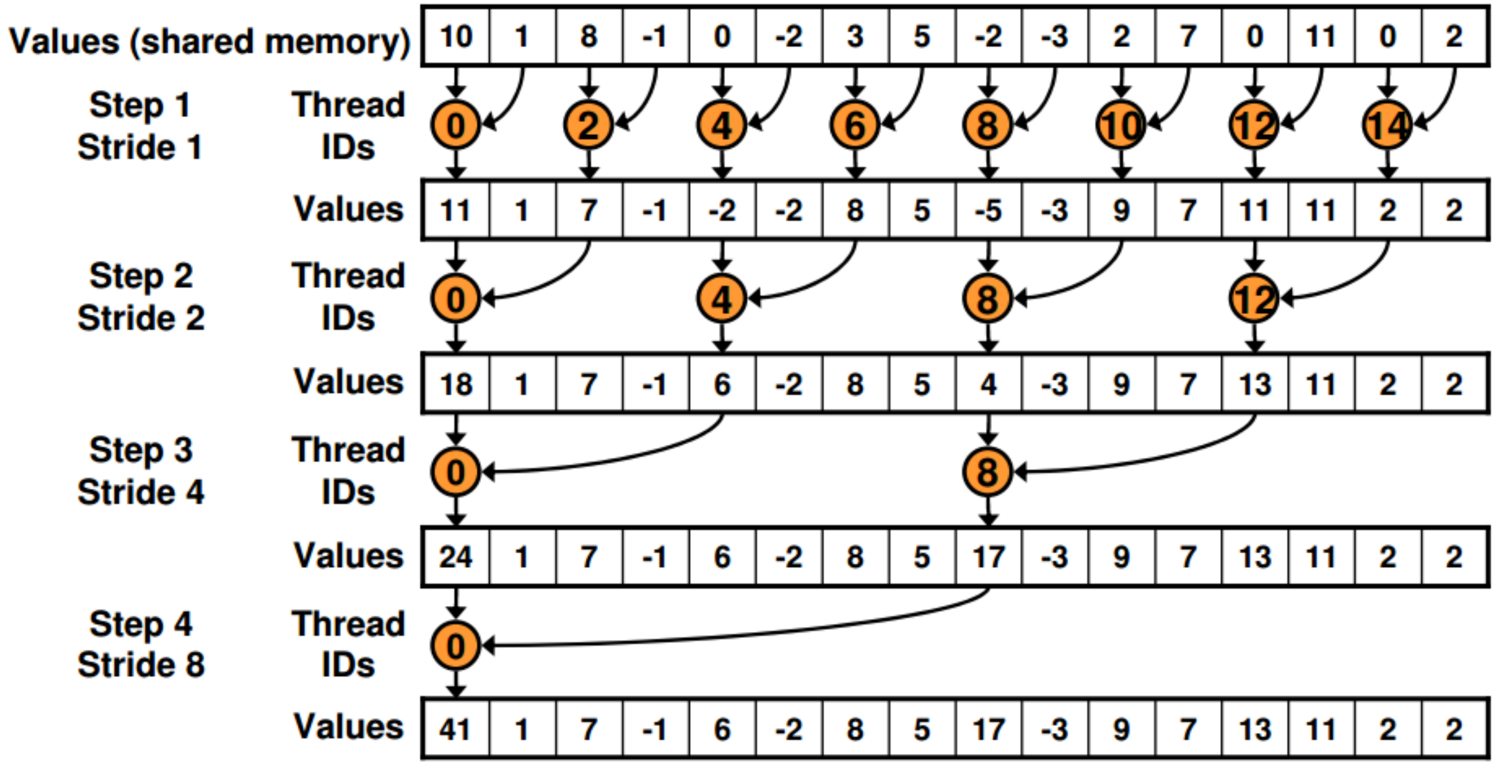
Problem 1

CUDA has no global synchronization.

Solution: decompose into multiple kernels.

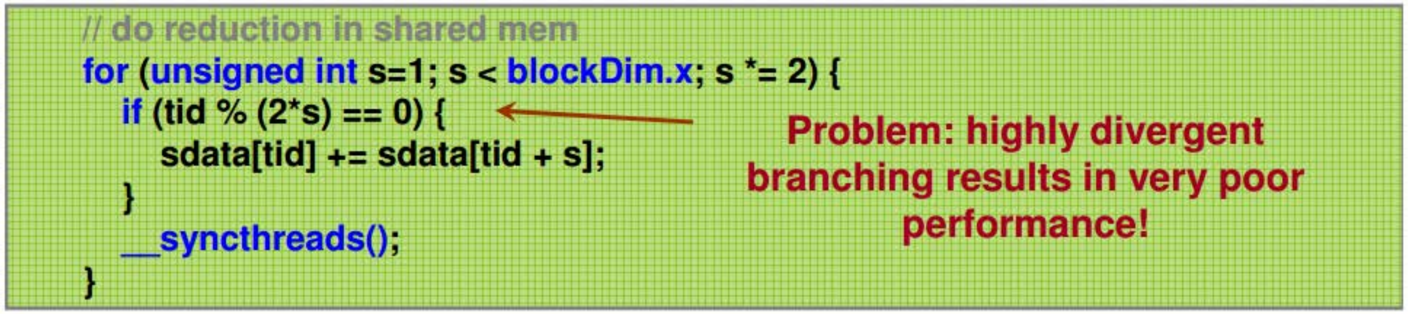


Problem 2: Thread divergence



Problem 2: Thread divergence

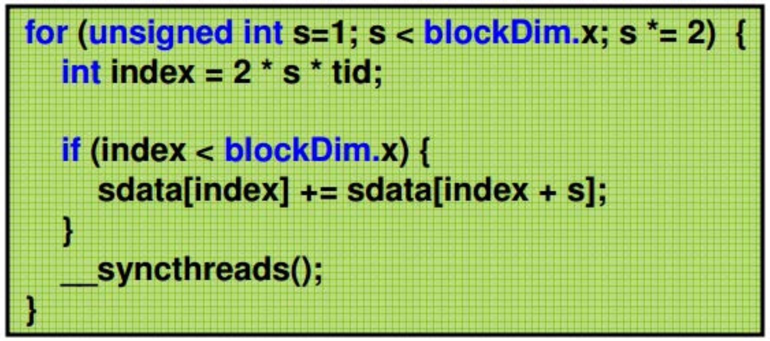
Thread divergence.



```
// do reduction in shared mem
for (unsigned int s=1; s < blockDim.x; s *= 2) {
    if (tid % (2*s) == 0) {
        sdata[tid] += sdata[tid + s];
    }
    __syncthreads();
}
```

Problem: highly divergent branching results in very poor performance!

Solution: strided index



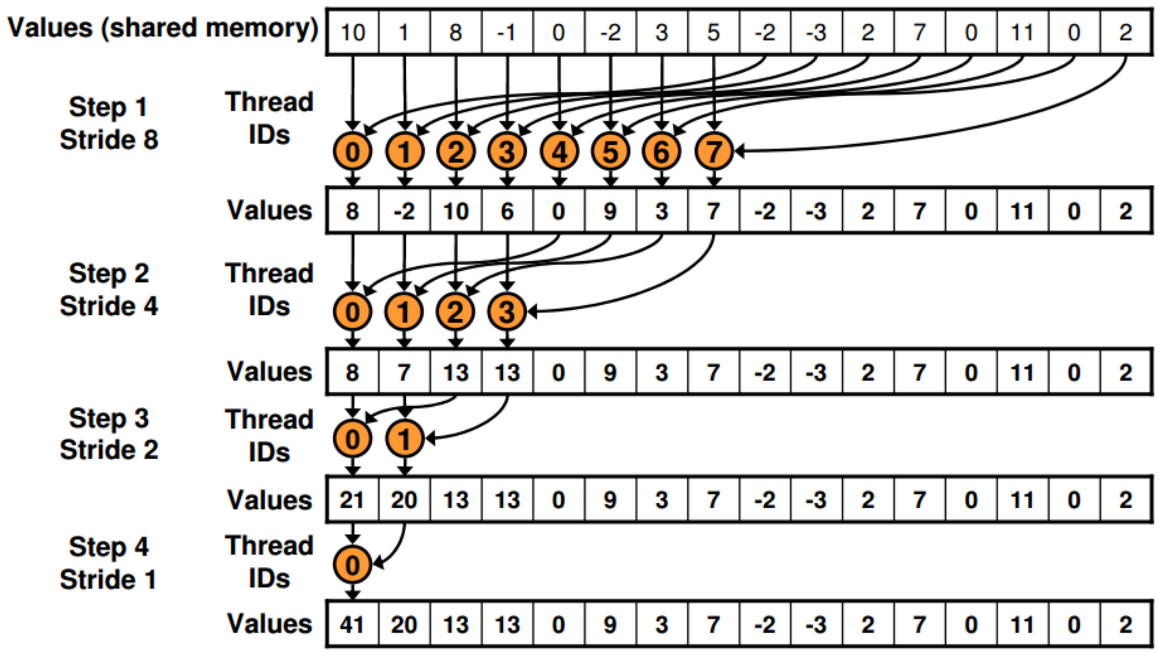
```
for (unsigned int s=1; s < blockDim.x; s *= 2)  {
    int index = 2 * s * tid;

    if (index < blockDim.x) {
        sdata[index] += sdata[index + s];
    }
    __syncthreads();
}
```

Problem 3: bank conflict

Shared memory bank conflict.

Solution: sequential addressing



Problem 3: bank conflict

Just replace strided indexing in inner loop:

```
for (unsigned int s=1; s < blockDim.x; s *= 2)  {
    int index = 2 * s * tid;

    if (index < blockDim.x) {
        sdata[index] += sdata[index + s];
    }
    __syncthreads();
}
```

With reversed loop and threadID-based indexing:

```
for (unsigned int s=blockDim.x/2; s>0; s>>=1) {
    if (tid < s) {
        sdata[tid] += sdata[tid + s];
    }
    __syncthreads();
}
```

Requirements

- Solving the 3 problems, and verify your correctness with TA
- If you can’t complete today, make another appointment with TA
- You are not required, but encouraged to further optimize the code as described in the course slides