

Lab4 CUDA Advance

Nov, 2024 Parallel Programming

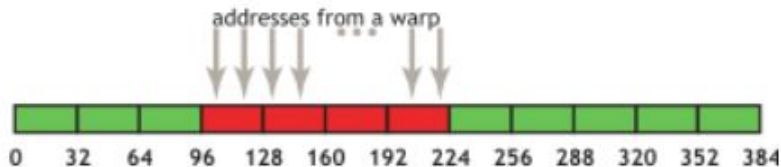
Overview

- ❖ Techniques that can further optimize a CUDA program
 - ❖ Coalesced Memory Access
 - ❖ Lower Precision
 - ❖ Shared Memory
 - ❖ Multiple Blocks
- ❖ Lab4

Coalesced Memory Access

❖ In short,

- Concurrent memory accesses in a warp should be continuous



❖ Why

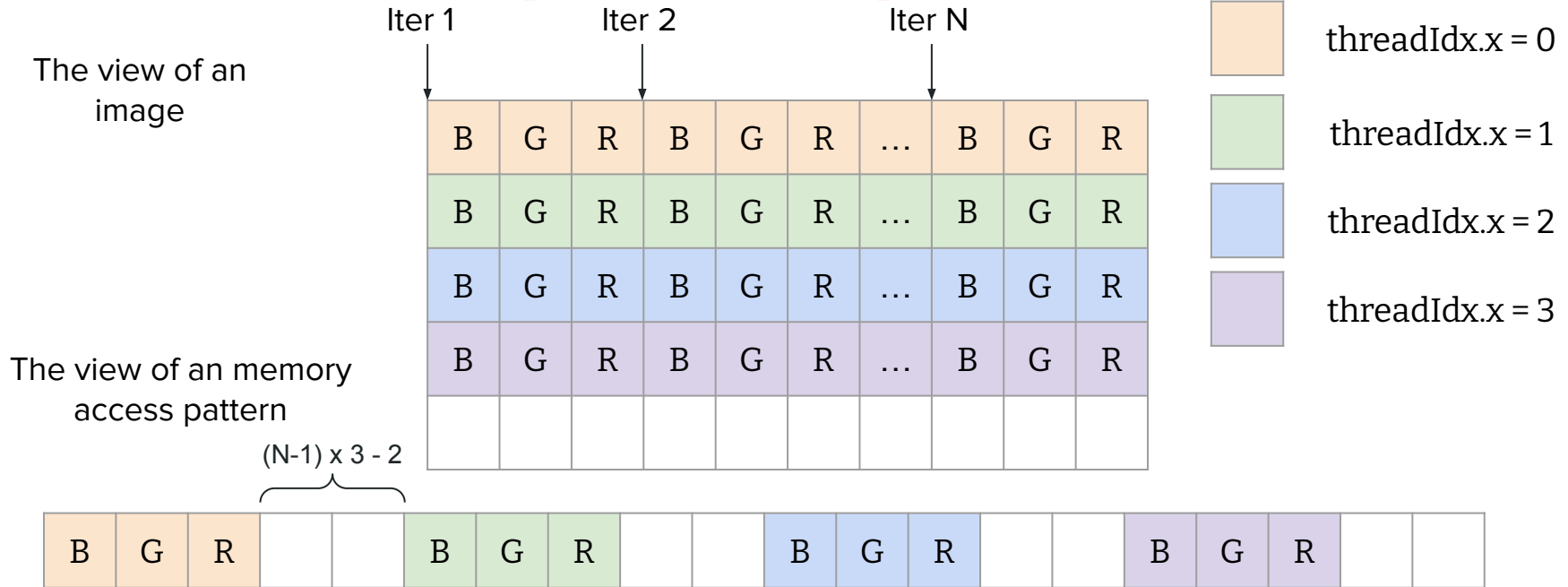
- GPU has L2 (32 bytes), L1 (128 bytes) cache
- If memory accesses in a warp are continuous, it can
 - merge memory requests from all threads into a single memory request
 - utilize the cache

❖ Details

- [CUDA Best Practices](#)

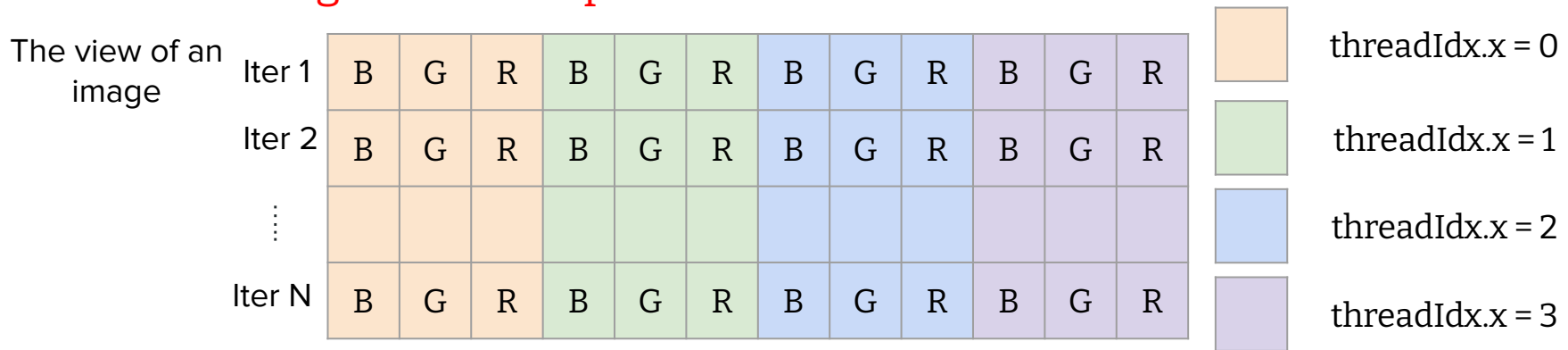
Access without Coalesced Memory

- ❖ If each thread compute a single row ->
Failed to combine requests into one request

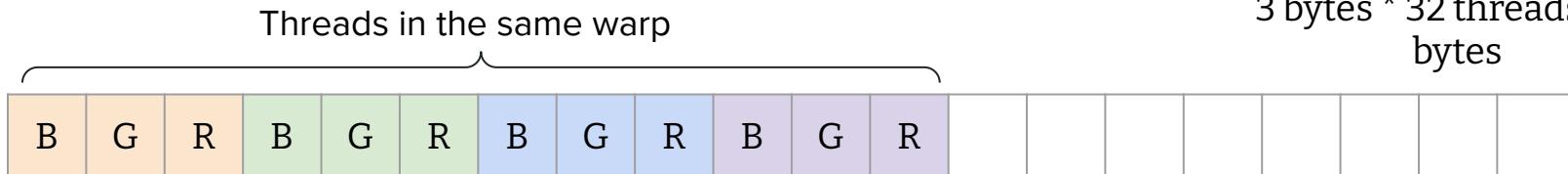


Coalesced Memory Access

- ❖ The accesses can be combined into a single request if we change the access pattern



The view of an memory access pattern



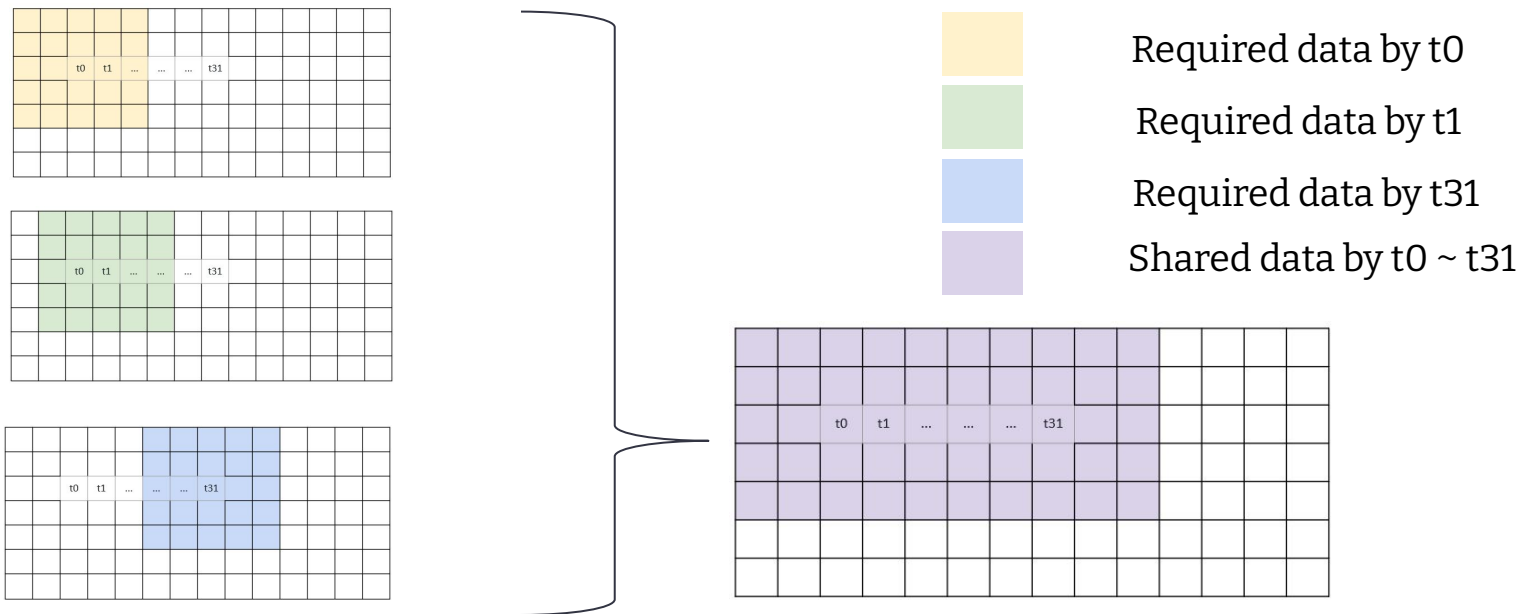
3 bytes * 32 threads = 96 bytes

Mixed-Precision

- ❖ Lower the precision of variables could reduce the computing time and also the computing accuracy
- ❖ Try to
 - ❖ Use float to replace double
 - ❖ Use fp16 to replace float
 - ❖ Make sure using lower precision does not corrupt the results

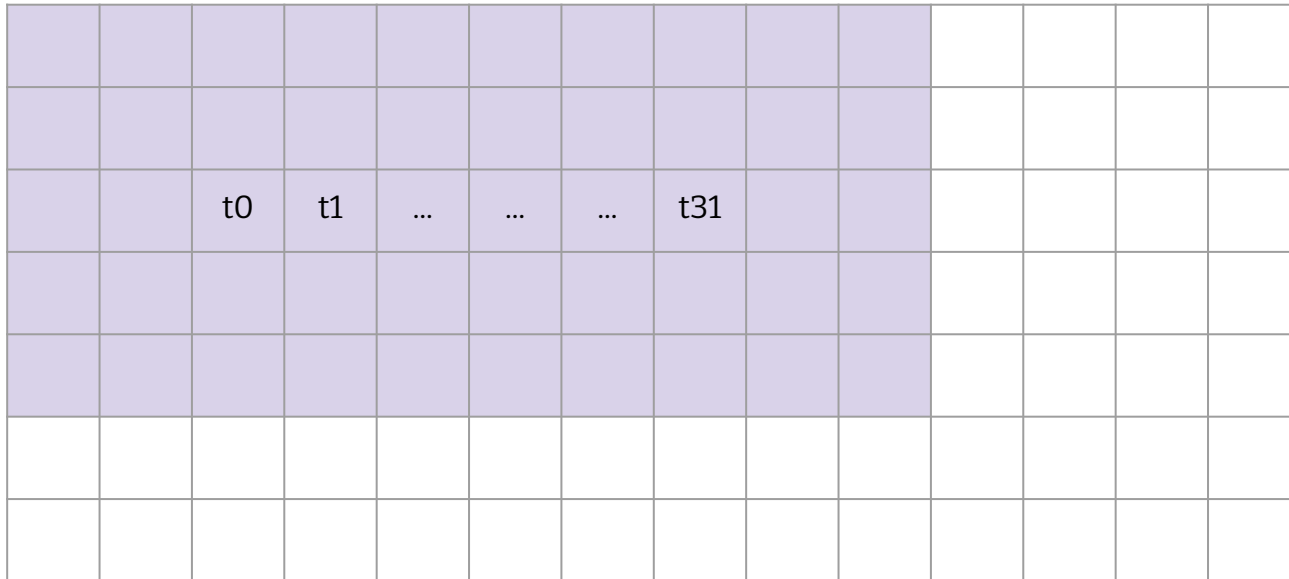
Shared Memory

- ❖ Shared memory can greatly reduce the access time of a **reused data item**



Using Shared Memory in Sobel

- ❖ Move the required data into shared memory
- ❖ Compute
- ❖ Update shared memory



Multiple Blocks

- ❖ The number of threads per block is limited comparing to the number of blocks per grid.
 - ❖ Therefore, we can launch more blocks for the higher level of parallelism
 - ❖ E.g., Break the computation into multiple blocks with the size of 32 x 32

```
Maximum number of threads per block:      1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
```

- ❖ Denoting the x, y coordinate of a pixel by threadIdx and blockIdx
 - ❖ E.g., `int x=blockIdx*blockDim+threadIdx;`
 - ❖ **Hint: you should choose the right indexing method to ensure coalesced memory access**

Lab4

- ❖ Optimize the sobel operator with the following
 - Coalesced Memory
 - Lower Precision
 - Shared Memory
- ❖ TAs provided a sample CUDA program
 - optimize it to be at least **13x faster**
 - Materials are under </home/pp24/share/lab4>
- ❖ Name your kernel as “sobel”
- ❖ We accept little pixel errors

Submission

- ❖ Finish it before **11/7 23:59**
- ❖ Submit your code and Makefile (optional) to eeclass
- ❖ You can use **lab4-judge** for pre-check