

# Lab4 CUDA Advance

---

*Nov, 2022 Parallel Programming*

# Overview

- ❖ Further optimize the code with following techniques:
  - Coalesced Memory Access
  - Lower Precision
  - Shared Memory
- ❖ Lab4

# Coalesced Memory Access

## ❖ In short,

- Concurrent memory accesses in a warp are continuous



## ❖ Why

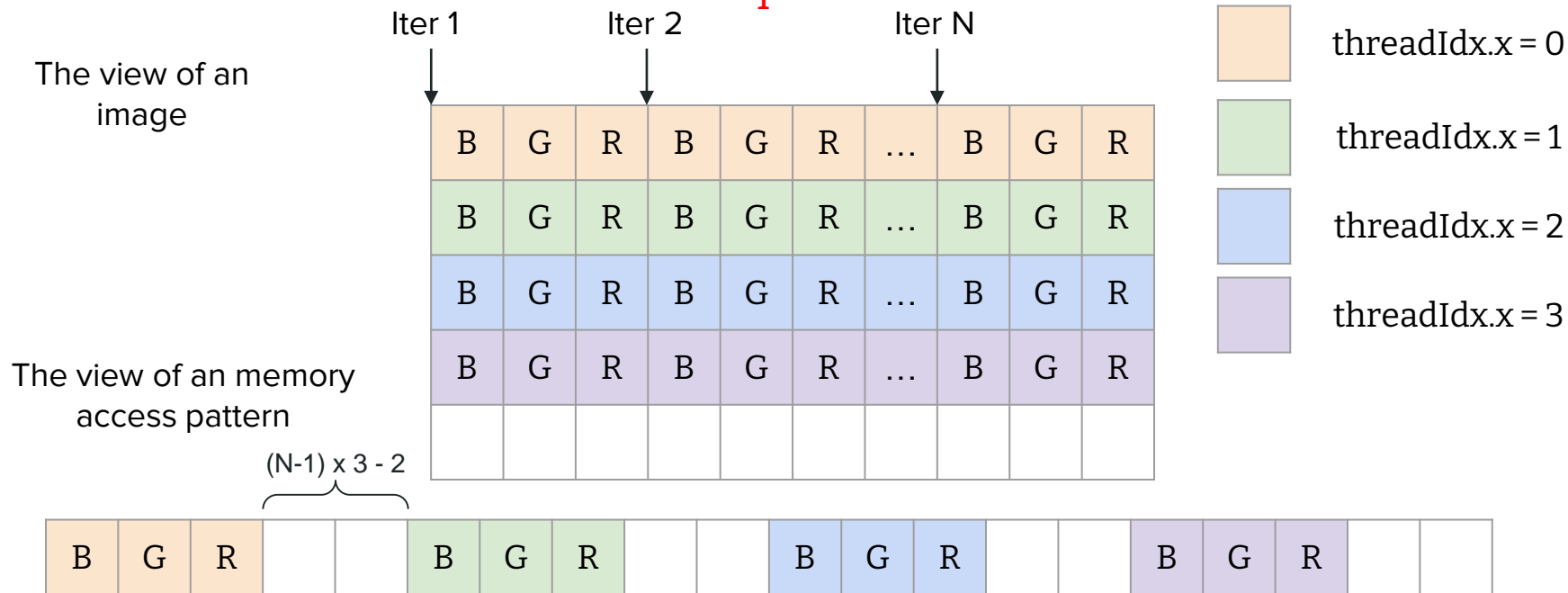
- GPU has L2 (32 bytes), L1 (128 bytes) cache
- If memory accesses in a warp are continuous, it can utilize the cache

## ❖ Details

- [CUDA Best Practices](#)

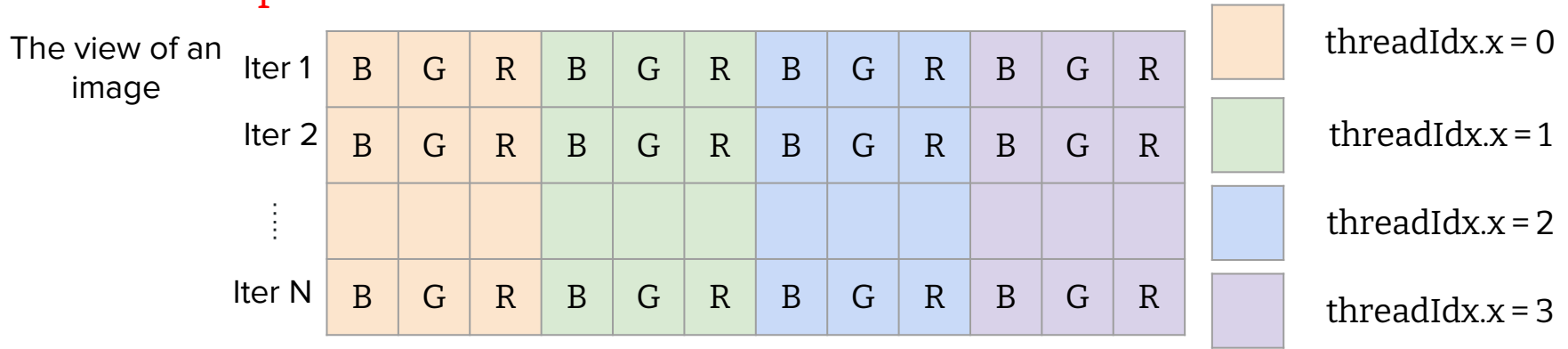
# Access without Coalesced Memory

- ❖ If each thread compute a single row -> Failed to combine the locations with different colors into one request

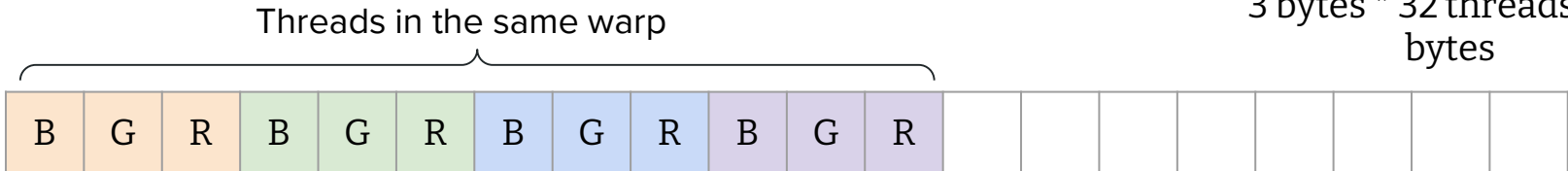


# Coalesced Memory Access

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



The view of an memory access pattern



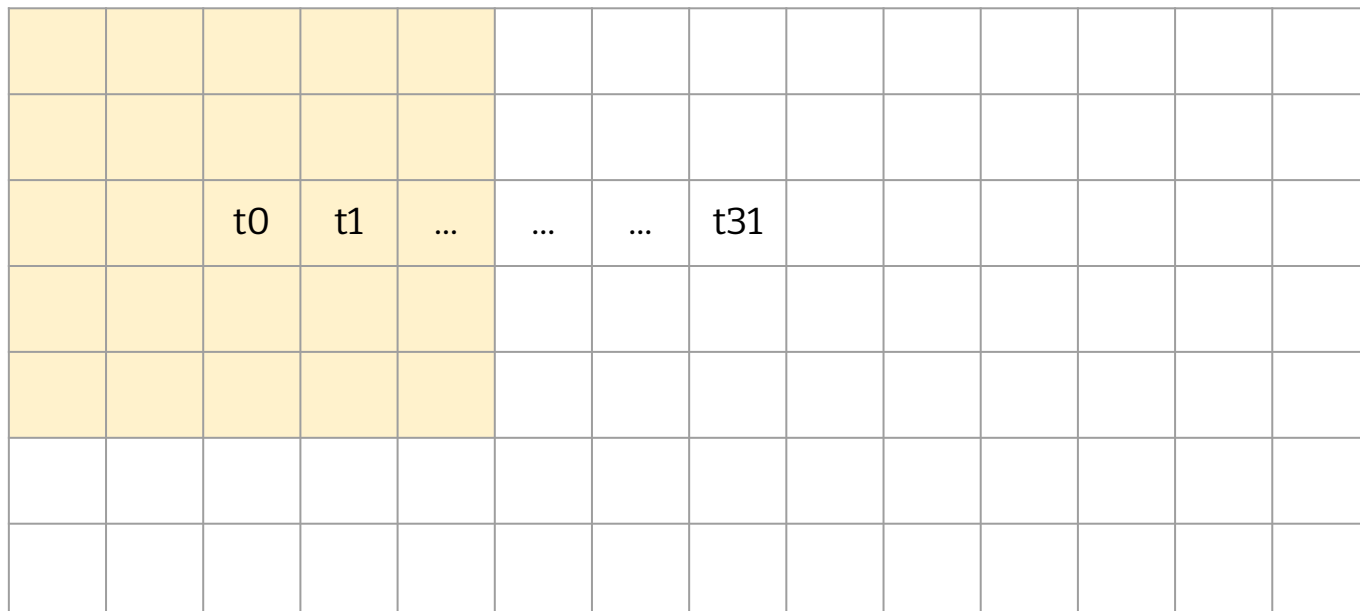
# Mixed-Precision

- ❖ Use lower precision when available
- ❖ Use float to replace double
- ❖ Use fp16 to replace float
- ❖ Make sure using lower precision does not affect the results

# Shared Memory

- ❖ Shared memory can greatly reduce the access time of a reused data item
- ❖ Find the reuse data from your code and move it to shared memory before accessing

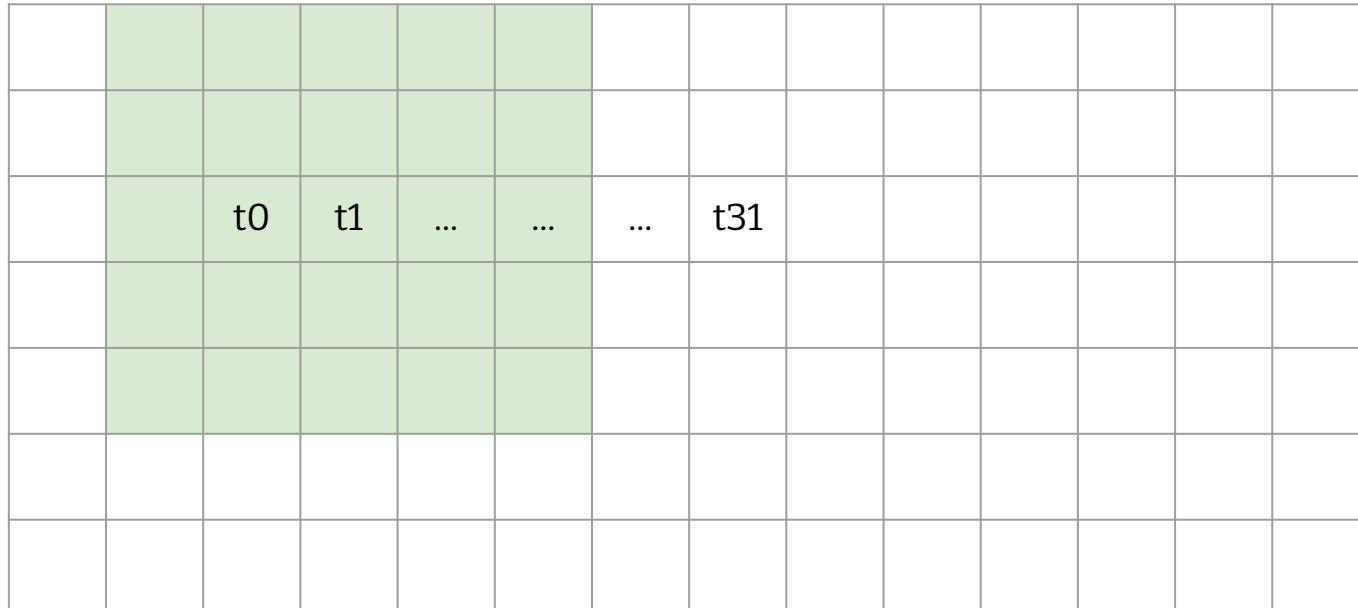
# Shared Memory with Sobel



Required data by t0

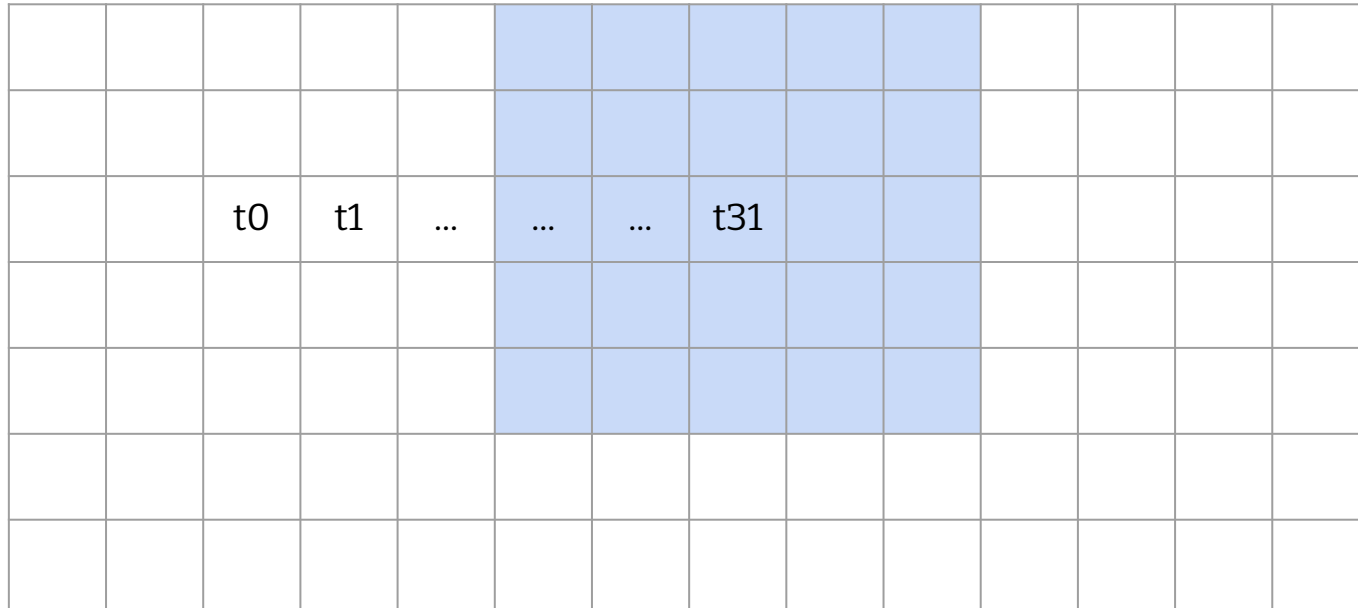


# Shared Memory with Sobel



Required data by t1

# Shared Memory with Sobel



Required data by t31

# Using Shared Memory in Sobel

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

[illegible]

# Lab4

- ❖ Optimize the sobel CUDA implementation
- ❖ TAs provide simple CUDA version
  - You are asked to accelerate it over **13x**
  - Materials are under `/home/pp22/share/lab4`
- ❖ Name your kernel as “sobel”
- ❖ We accept little pixel errors

# Submission

- ❖ Finish it before 12/1 23:59
- ❖ Submit your code and Makefile (optional) to eeclass
- ❖ You can use **lab4-judge** for pre-check