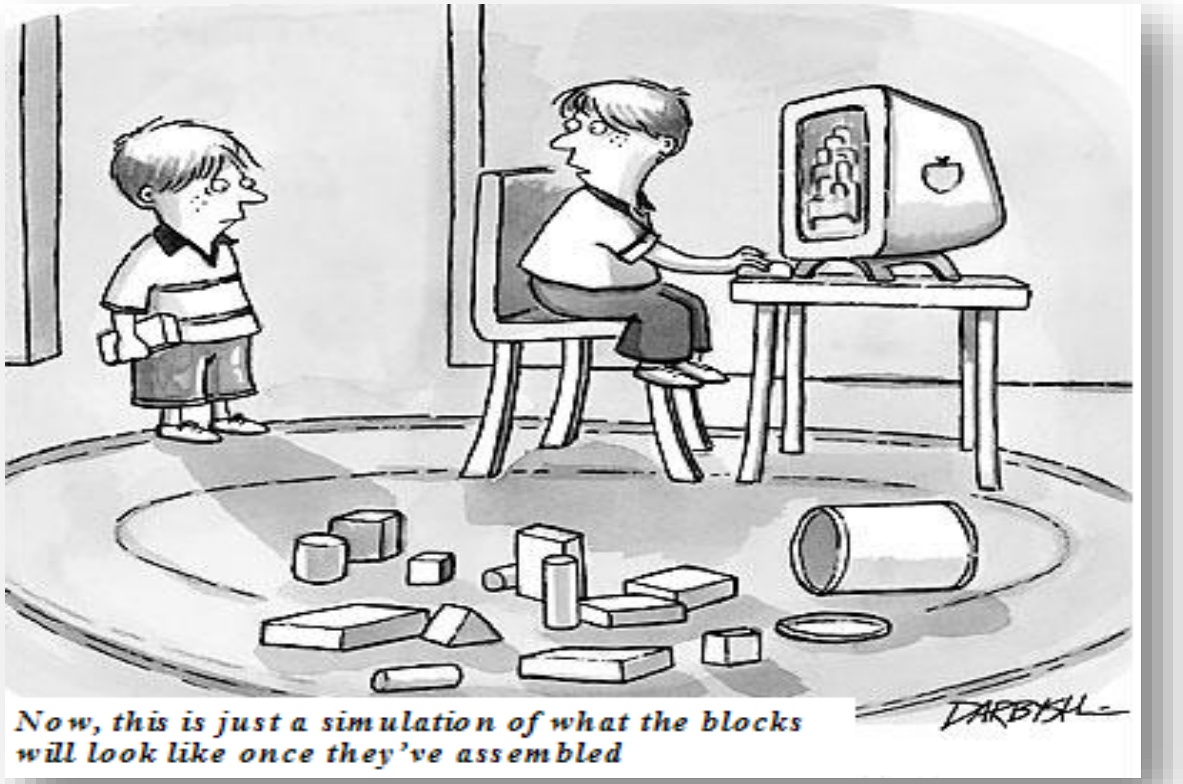
ECE569
Module 30



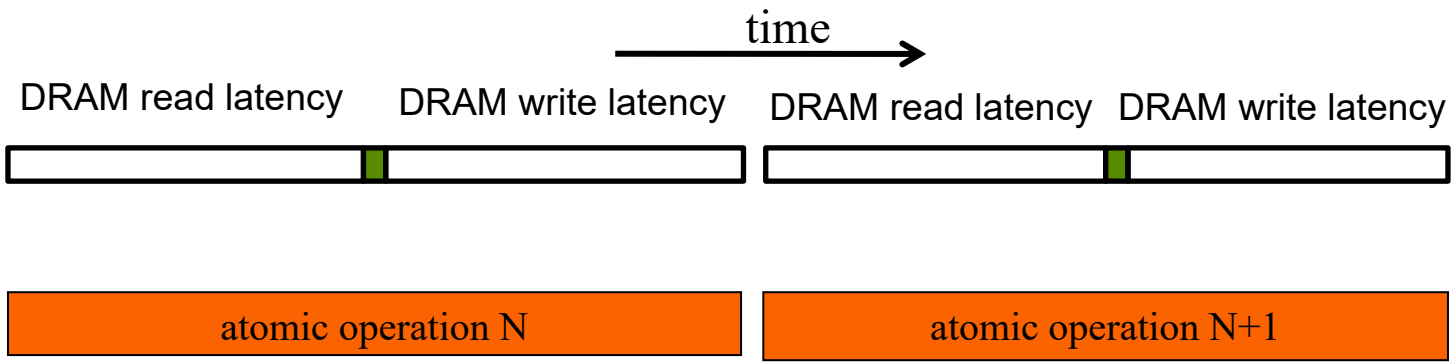
- Atomic Operations and DRAM

Atomic Operations on Global Memory (DRAM)

- As long as we have **many threads** whose **memory access latencies** can **overlap** with each other, the execution speed is limited by the throughput of the memory system.
  - It is important that GPUs make full use **DRAM bursts, banks, and channels** to achieve very high memory access throughput
- Unfortunately, this assumption **breaks down** when many atomic operations update the same memory location.
  - the read-modify-write sequence of a trailing thread cannot start until the read-modify-write sequence of a leading thread is complete.

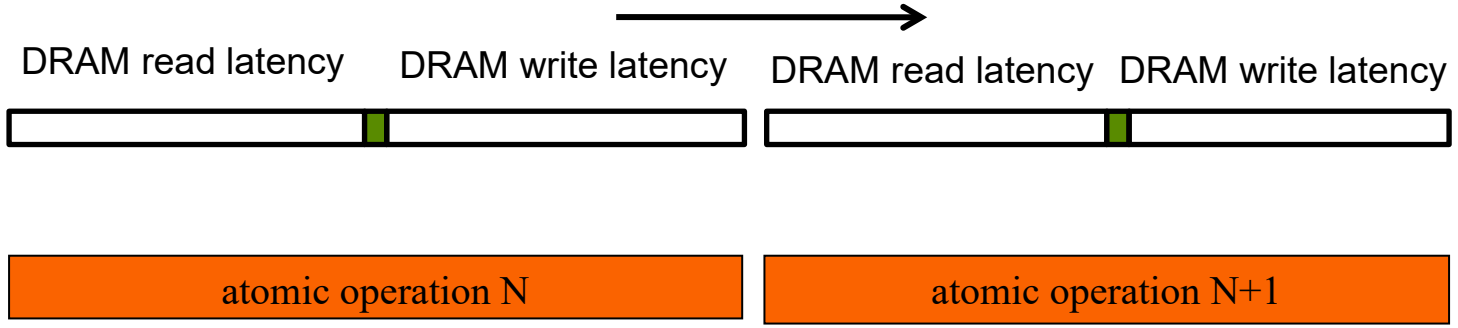
Atomic Operations on Global Memory (DRAM)

- **Each Read-Modify-Write has two full memory access delays**
  - All atomic operations on the same variable (DRAM location) are serialized
  - During this whole time, no one else can access the location



Latency determines throughput

- The rate for atomic operation on a particular location is limited by the total latency of the read-modify-write sequence, typically more than **1000 cycles** for global memory (DRAM) locations.
- This means that if many threads attempt to do atomic operation on the same location (contention), the memory throughput is reduced to < **1/1000** of the peak bandwidth of one memory channel!



Example

- Assume a memory system with
  - 64-bit Double Data Rate DRAM interface,
  - 8 channels,
  - 1 GHz clock frequency,
  - Access latency of 200 cycles.
- The peak access throughput of the memory system:

Example

- Assume a memory system with
  - 64-bit Double Data Rate DRAM interface,
  - 8 channels,
  - 1 GHz clock frequency,
  - Access latency of 200 cycles.
- The peak access throughput of the memory system:
  - 8 (bytes/transfer)*2 (transfers per clock per channel)* 1G (cycles per second) * 8 (Channels) = 128 GB/sec. Assuming each data accessed is 4 bytes, the system has a peak access throughput of 32G data elements per second.

Example

- Assume a memory system with
  - 64-bit Double Data Rate DRAM interface,
  - 8 channels,
  - 1 GHz clock frequency,
  - Access latency of 200 cycles.
- The peak access throughput of the memory system:
  - 128 GB/sec (32G data elements per second)
- When performing atomic operations on a particular memory location, what is the highest throughput (atomics/second)? (Ignore computation latency)

Example

- Assume a memory system with
  - 64-bit Double Data Rate DRAM interface,
  - 8 channels,
  - 1 GHz clock frequency,
  - Access latency of 200 cycles.
- The peak access throughput of the memory system:
  - 128 GB/sec (**32G data elements per second**)
- When performing atomic operations on a particular memory location, what is the highest throughput (atomics/second)?
  - 1 atomic operation every 400 cycles (200 cycles for the read and 200 cycles for the write). This translates into a time-based throughput of 1/400 atomics/cycle * 1G (clocks/second) = **2.5M atomics/second**.

Example

- Assume a memory system with
  - 64-bit Double Data Rate DRAM interface,
  - 8 channels,
  - 1 GHz clock frequency,
  - Access latency of 200 cycles.
- The peak access throughput of the memory system:
  - 128 GB/sec (**32G data elements per second**)
- Highest throughput (atomics/second)?
  - **2.5M atomics/second**.
- What is the throughput for the histogram for evenly distributed pixel intensity values?

Example

- Assume a memory system with
  - 64-bit Double Data Rate DRAM interface,
  - 8 channels,
  - 1 GHz clock frequency,
  - Access latency of 200 cycles.
- The peak access throughput of the memory system:
  - 128 GB/sec (**32G data elements per second**)
- Highest throughput (atomics/second)?
  - **2.5M atomics/second**.
- What is the throughput for the histogram equalization for evenly distributed pixel intensity values?
  - 2.5M*256 =**640M** atomic operations/sec

Improving Performance of Atomic Operations

- **L2 Cache**
  - Shared among all thread blocks
  - Programmer benefits without any coding effort
  - 10x faster than DRAM
- **Shared memory**
  - Shared among threads in a block
  - Requires code revisions
  - 10x faster than L2

Shared Memory based Atomic Operations

- **write a high performance kernel by privatizing outputs**
  - Privatization as a technique for reducing latency, increasing throughput, and reducing serialization
  - Practical example of using shared memory and L2 cache atomic operations

Next

- **Shared memory based histogram**