CUDA: Memory

Mark Greenstreet

CpSc 418 – Mar. 16, 2016

- GPU Memory Hierarchy: In Practice

GPU Memory Hierarchy

- A few topics about the memory hierarchy, left over from March 16.
  - Global memory: coalescing references
  - Other memories on the GPU
- An example, and lessons learned
  - The example: shared-memory bank conflicts
  - Lessons learned

Global memory: coalescing references

Other Memory

- Constant memory: cached, read-only access of global memory.
- Texture memory: global memory with special access operations.
- L1 and L2 caches: only for memory reads?
- We won't cover these any further in class
  - Nor are you expected to use them.
  - But you're welcome to try them if you want.

Shared Memory: Bank Conflicts

The code is at .

What I learned

- The scope of `__shared__` variables.
- Keeping the SMs busy.
- A few notes about floating point.

The scope of `__shared__` variables

- A shared variable is visible by all threads in the same block.
- That means there's a different instance of the shared variable for each block.
- This puts a limit on the number of blocks that can run on a SM.
  - With CUDA 2.1, each SM has 48K bytes of shared memory.
  - If a block needs 12K bytes of shared memory, then at most 4 blocks can execute on the SM at the same time.
  - Note that a SM has 32K, 4-byte registers – the register storage is more than 2.5$\times$ the shared memory capacity.

Keeping SMs busy

- Occupancy
- Limits to occupancy
  - How many blocks per SM.
  - How much shared-memory per block.
  - How many threads per block.
  - How many registers per thread.
- Figuring it out
  - `nvcc -O3 -c --ptxas-options -v examples.cu`
  - The nVidia occupancy calculator (link to be added)

Remarks about floating point