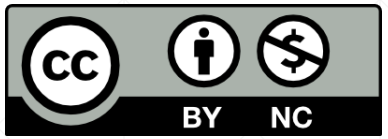
us4us®

IPPT PAN

Introduction to Parallel Programming with NVIDIA CUDA

GPU Memory

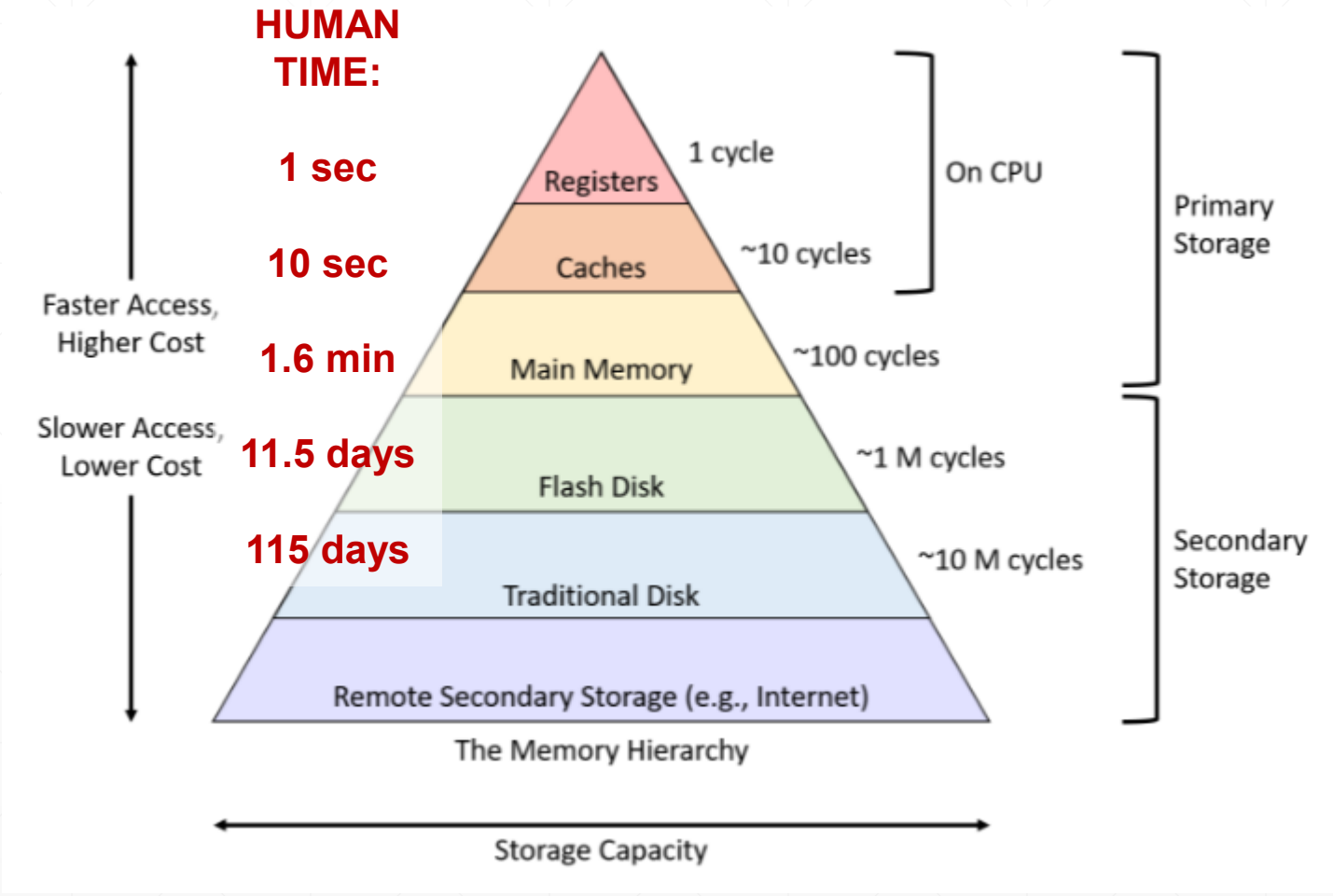
License / Attribution



- Materials for the short-course **„Digital Signal Processing with GPUs — Introduction to Parallel Programming”** are licensed by us4us Ltd. the IPPT PAN under the Creative Commons Attribution-NonCommercial 4.0 International License.

- Some slides and examples are borrowed from the course **„The GPU Teaching Kit”** that is licensed by NVIDIA and the University of Illinois under the Creative Commons Attribution-NonCommercial 4.0 International License.
  - All the borrowed slides are marked with NVIDIA ILLINOIS

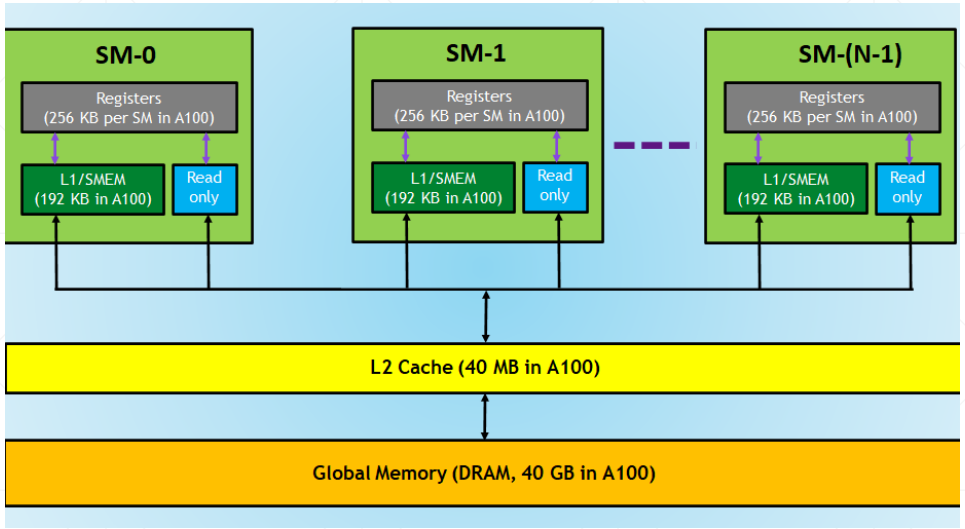
Computer Memory hierarchy

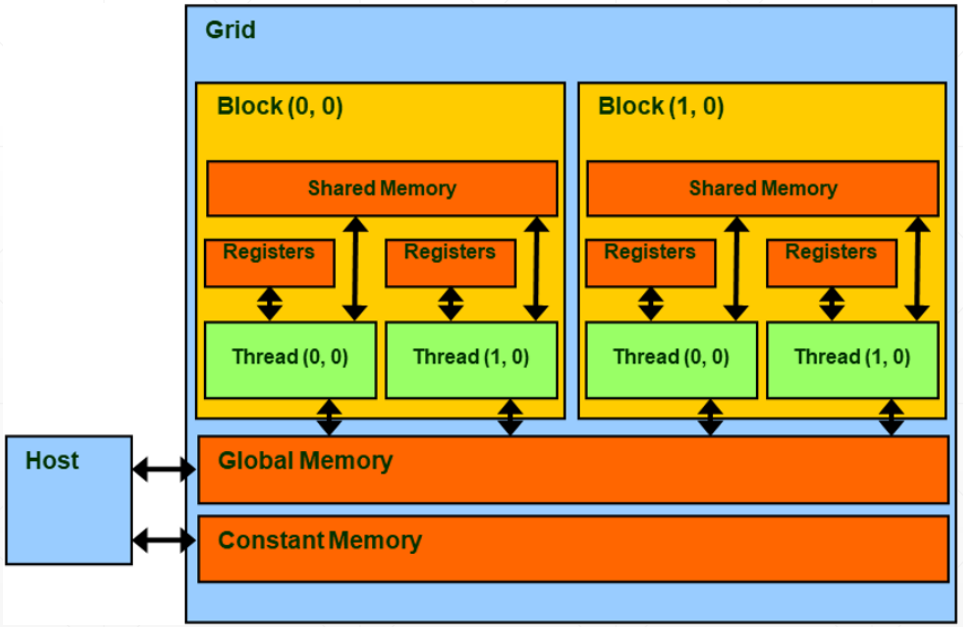


Source: https://diveintosystems.org/antora/diveintosystems/1.0/MemHierarchy/mem_hierarchy.html

GPU Memory Hierarchy

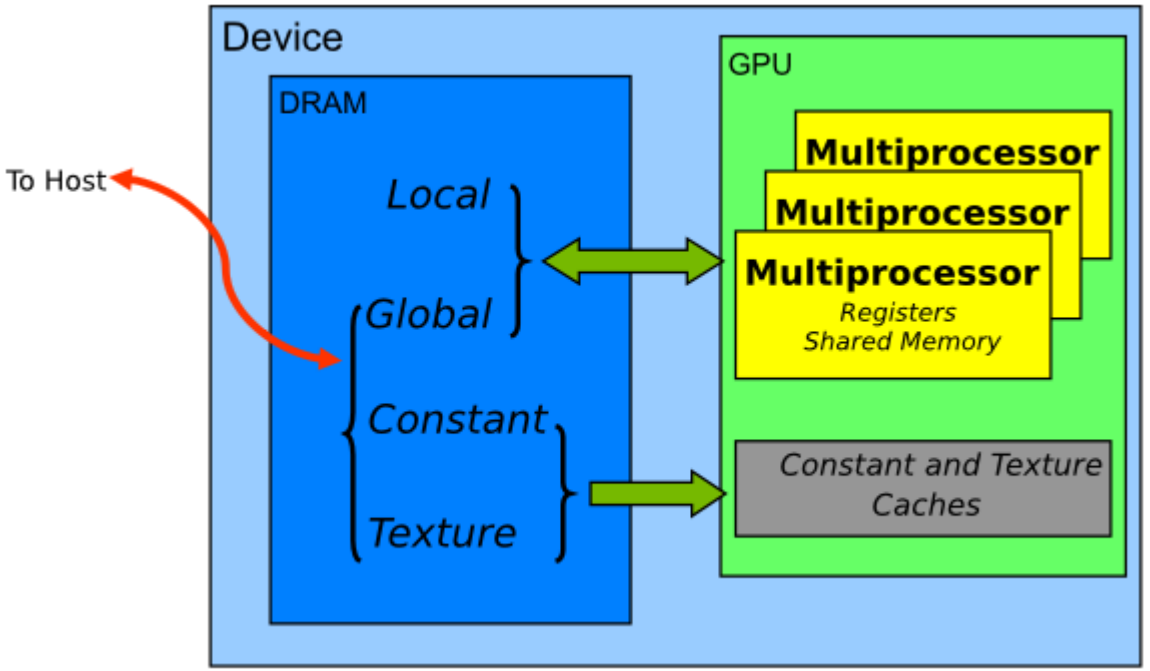
- **Registers**—These are private to each thread, which means that registers assigned to a thread are not visible to other threads. The compiler makes decisions about register utilization.
- **L1/Shared memory (SMEM)**—Every SM has a fast, on-chip scratchpad memory that can be used as L1 cache and shared memory. All threads in a CUDA block can share shared memory, and all CUDA blocks running on a given SM can share the physical memory resource provided by the SM..
- **Read-only memory**—Each SM has an instruction cache, constant memory, texture memory and RO cache, which is read-only to kernel code.
- **L2 cache**—The L2 cache is shared across all SMs, so every thread in every CUDA block can access this memory. The NVIDIA A100 GPU has increased the L2 cache size to 40 MB as compared to 6 MB in V100 GPUs.
- **Global memory**—This is the framebuffer size of the GPU and DRAM sitting in the GPU.

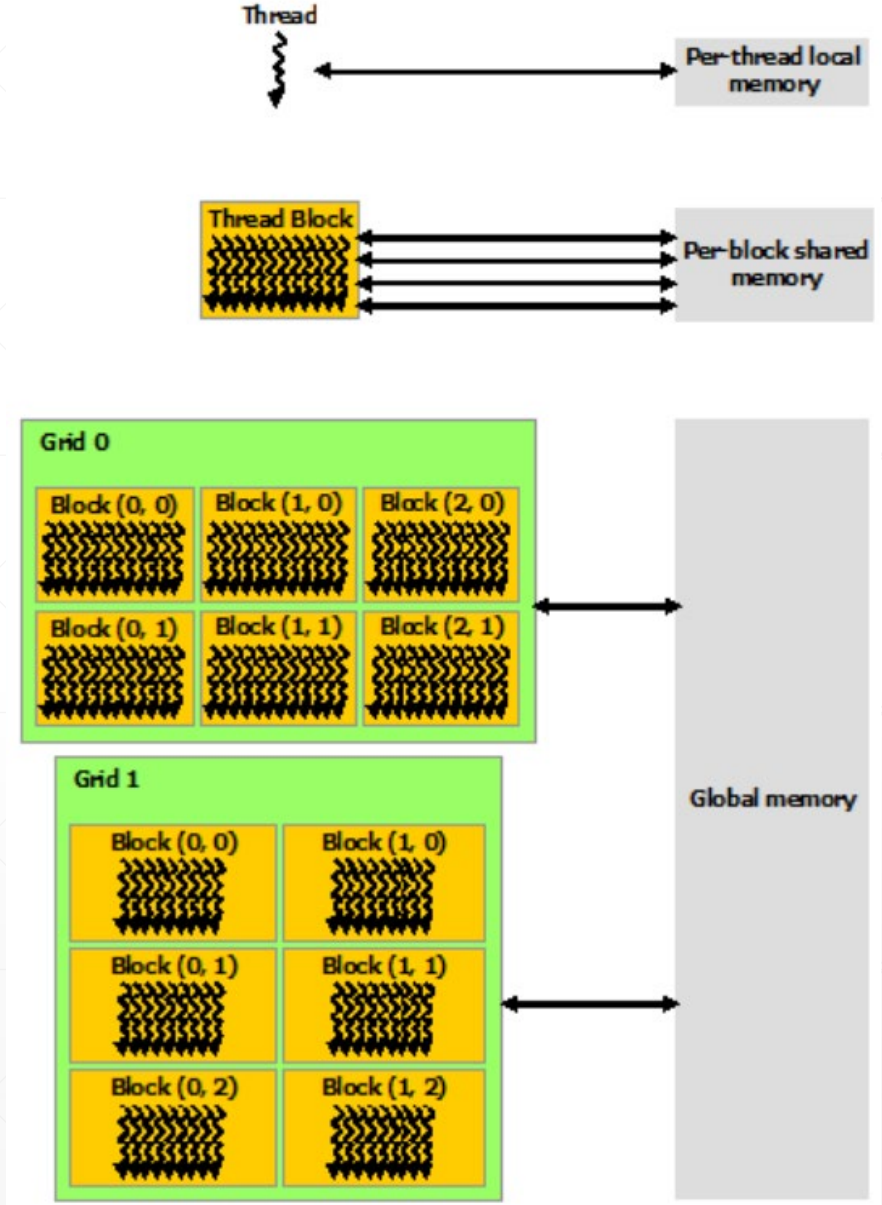




Source: https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/

GPU memory access





Source: https://docs.nvidia.com/cuda/

GPU Memory Features by Type

Table 1. Salient Features of Device Memory

Memory	Location on/off chip	Cached	Access	Scope	Lifetime
Register	On	n/a	R/W	1 thread	Thread
Local	Off	Yes††	R/W	1 thread	Thread
Shared	On	n/a	R/W	All threads in block	Block
Global	Off	†	R/W	All threads + host	Host allocation
Constant	Off	Yes	R	All threads + host	Host allocation
Texture	Off	Yes	R	All threads + host	Host allocation

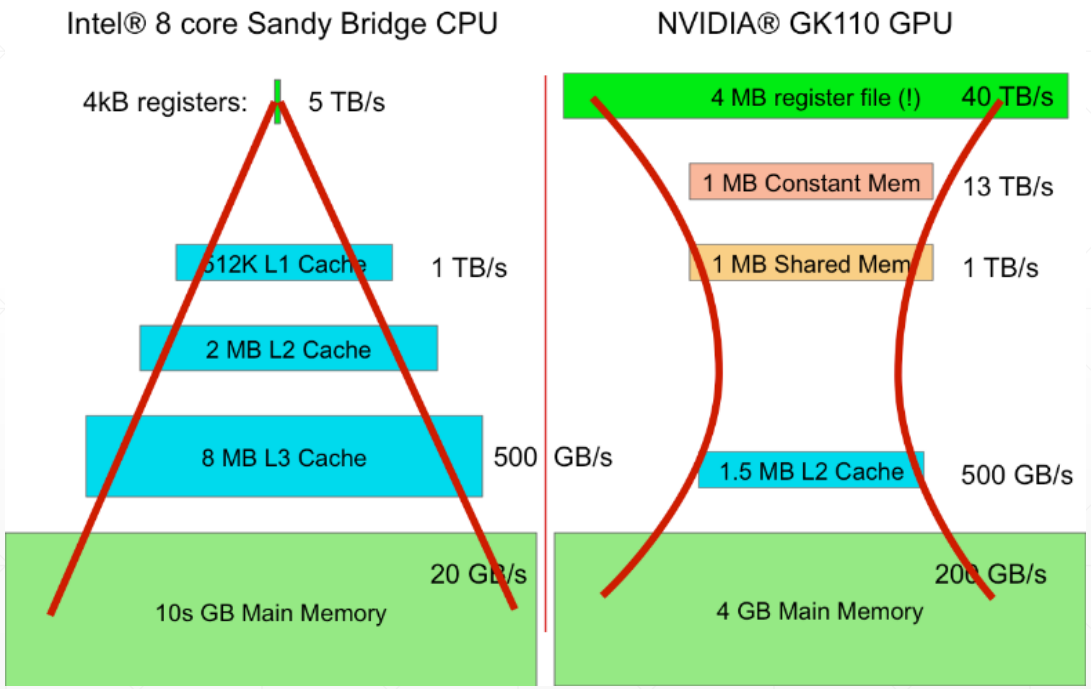
† Cached in L1 and L2 by default on devices of compute capability 6.0 and 7.x; cached only in L2 by default on devices of lower compute capabilities, though some allow opt-in to caching in L1 as well via compilation flags.

†† Cached in L1 and L2 by default except on devices of compute capability 5.x; devices of compute capability 5.x cache locals only in L2.

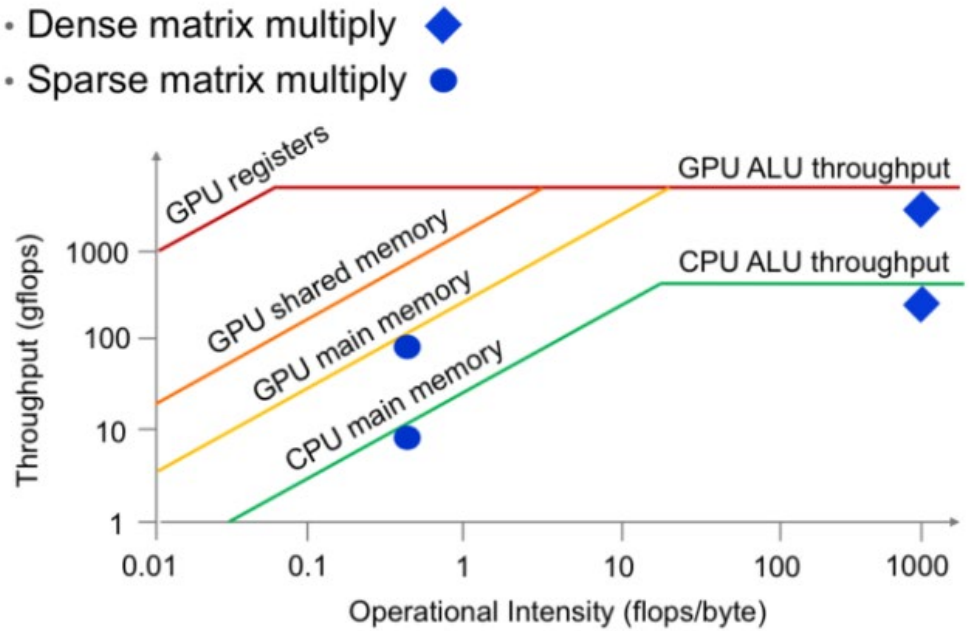
Source: https://docs.nvidia.com/cuda/

Memory / Arithmetic intensity / Performance

Where is my Memory?



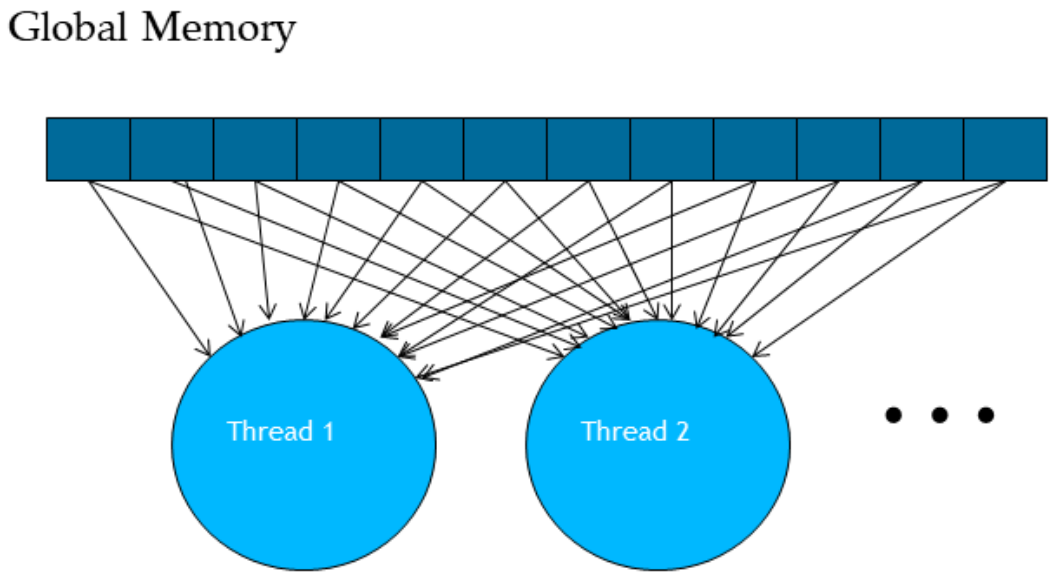
Roofline Design – Matrix kernels



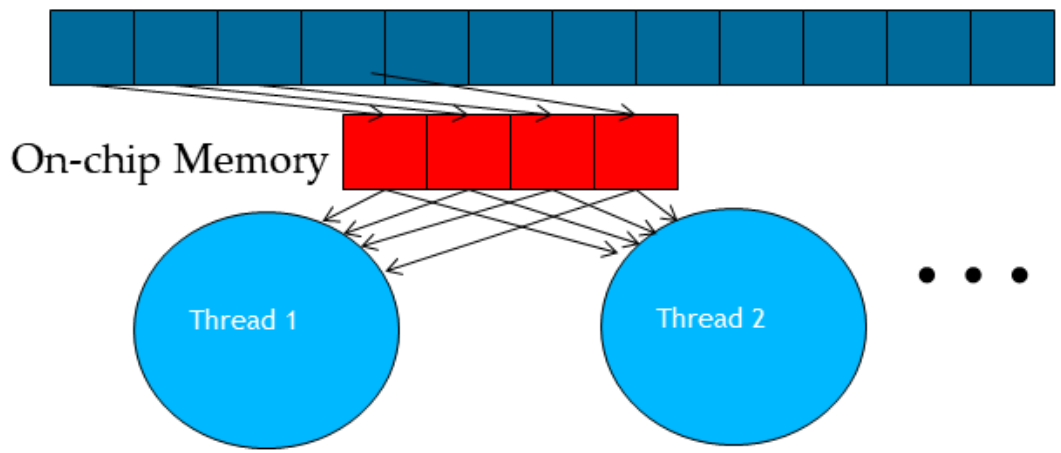
Source: https://developer.nvidia.com/blog/bidmach-machine-learning-limit-gpus/ | https://en.wikipedia.org/wiki/Roofline_model

Tiling/Blocking memory

Global Memory Access Pattern of the Basic Matrix Multiplication Kernel



Tiling/Blocking - Basic Idea



Divide the global memory content into tiles

Focus the computation of threads on one or a small number of tiles at each point in time

Outline of Tiling Technique

- Identify a tile of global memory contents that are accessed by multiple threads
- Load the tile from global memory into on-chip memory
- Use barrier synchronization to make sure that all threads are ready to start the phase
- Have the multiple threads to access their data from the on-chip memory
- Use barrier synchronization to make sure that all threads have completed the current phase
- Move on to the next tile