21 GPUs

Lecture 21 GPUs

Three Main Goals for Current Architectures

Maintain execution speed of old sequential programs → CPU

Increase throughput of parallel programs → GPU

Reduce execution time of moderately or non-GPU-friendly parallel programs → Multicore

Winning Applications Use Both CPU and GPU

CPUs for sequential parts, or low or non-data-level parallelism, where latency matters

- CPUs can be 10X+ faster than GPUs for sequential code

GPUs for parallel parts where throughput wins

- GPUs can be 10X+ faster than CPUs for parallel GPU-friendly code

What are GPUs good for?

You will get the best performance from GPU if your application is:

- Computation intensive
- Many independent computations
- Many similar computations
- Problem size is big enough

The Hardware

NVLINK: Point-to-point connection, GPU-to-GPU and CPU-GPU connections

A Glimpse at a GPU:

Streaming Multiprocessor (SM)

Streaming Processor (SP), or CUDA core

- SPs within SM share control logic and instruction cache. That is, SPs are just execute units.

Quick Glimpse At Programming Models

Application can include multiple kernels

Threads of the same block run on the same SM

- So threads in SM can operate and share memory
- Block in an SM is divided into warps of 32 threads each
- A warp is the fundamental unit of dispatch in an SM

Blocks in a grid can coordinate using global shared memory

Each grid executes a kernel

Scheduling In NVIDIA GPUs

Two-level, distributed thread scheduler

- At the chip level: a global work distribution engine schedules thread blocks to various SMs
- At the SM level, each warp scheduler distributes warps of 32 threads to its execution units.

Modern GPUs can simultaneously execute multiple kernels of the same application

Two warps from different blocks (or even different kernels) can be issued and executed simultaneously

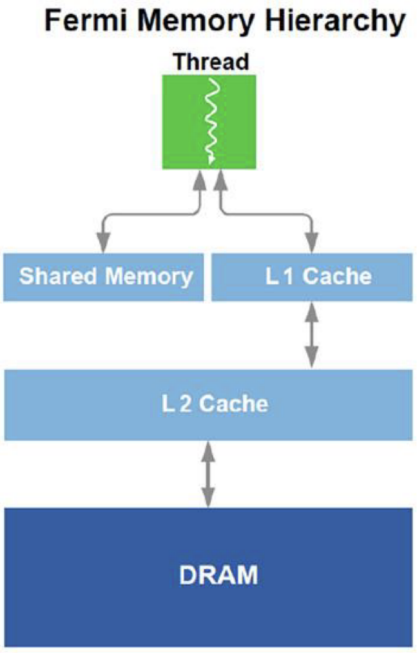
The Memory Hierarchy

Local memory in each SM

- first-level (L1) cache
- shared memory

GPUs are also equipped with an L2 cache, shared among all SMs.

The last level is the global memory.



Conclusions

The main keywords:

- data parallelism

- kernel, grid, block, and thread
- SP (or cuda cores), SMs, and warps

Some applications are better run on CPU while others on GPU.

Main limitations

- The parallelizable portion of the code
- The communication overhead between CPU and GPU
- Memory bandwidth saturation