Introduction to GPU Programming

Wim R.M. Cardoen, PhD

Center of High-Performance Computing (CHPC)
University of Utah

October 14, 2024

Outline

Theoretical GFLOP/s: GPU vs. CPU

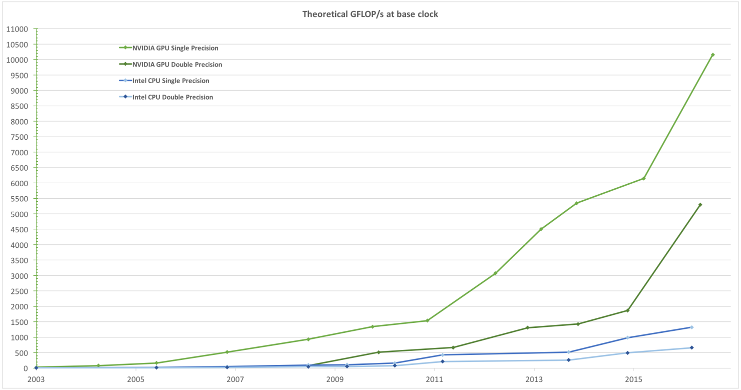
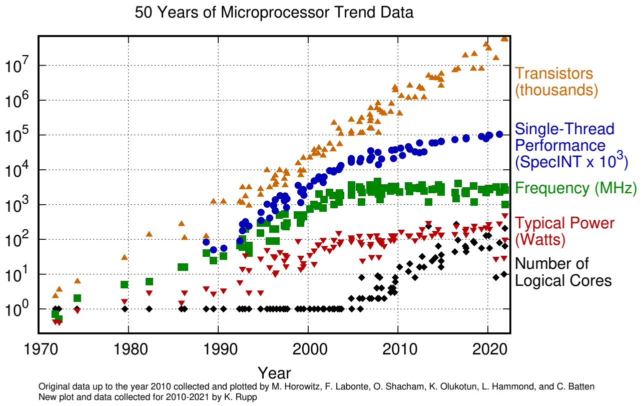


Figure: Theoretical GFLOP/s: GPUs vs. CPUs.[a]

[a]https://docs.nvidia.com/cuda/archive/9.1/pdf/CUDA_C_Programming_Guide.pdf

CPU processor trend (last 50 years)



- After the year 2000, `freq./power` for a single CPU core reaches a max. (**Heat dissipation!**).

Energy efficiency per job: GPU vs. CPU

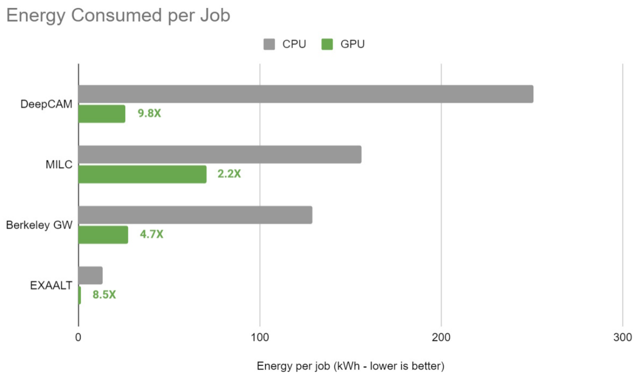


Figure: Energy efficiency per job (NERSC).[a]

[a] https://blogs.nvidia.com/blog/gpu-energy-efficiency-nersc/ (05/21/2023)

Streaming Multiprocessor (SM) & Thread Blocks

An understanding of the GPU hardware is fundamental for CUDA programming.

- Each GPU device contains a set of Streaming Multiprocessors (**SM**). (newer generation of GPU devices: #SMs increase (**scalability**))
- Work i.e. **blocks of threads** are distributed on the different SMs (**load balancing**) until SMs are **full**[1].
- If the work of a block of threads is completed on an SM, a new block is provided (until all work is done).
- Threads with the same thread block run on the **same SM**. (allows them to communicate using their shared memory & synchronize).
- An SM may run several threads blocks at the same time.
- Threads within the same thread block are alive simultaneously.

[1] i.e. resources (registers, shared memory, . . .) are exhausted

Examples of hardware currently available at CHPC: Part 1

- `NVIDIA A100-PCIE-40GB` (notch293)
  - 108 SMs, 64 cores/SM, 4 third-gen tensor cores/SM.
  - GPU max. clock rate: 1.41 GHz
  - max. #threads/SM: 2048
  - max. #blocks/SM : 32
  - FP32 cores/SM: 64
  - FP64 cores/SM: 32
- `NVIDIA H100 SXM5 NVL` (grn008)
  - 132 SMs, 128 cores/SM, 4 fourth-gen tensor cores/SM.
  - GPU max. clock rate: 1.78 GHz
  - max. #threads/SM: 2048
  - max. #blocks/SM : 32
  - FP32 cores/SM: 128
  - FP64 cores/SM: 64

Types of GPU memory

GPU devices have different types of memory:

- **global** memory: available to all threads.
- **shared** memory: common to all threads in a block.
  The shared memory available on an SM is divided among the blocks on the SM.
- 32-bit **registers**: fast, on-chip memory (exclusive to each thread).
  $\Rightarrow$ registers per block: (threads per block) $\times$ (registers per thread).
- **constant** memory: cached, read-only

Advice:

- Try to max. use shared memory & registers.
- Due to their limited availabity, their overuse in a block may lead to partially occupied/underutilized SMs.

Examples of hardware currently available at CHPC (Part 2)

- `NVIDIA A100-PCIE-40GB` (notch293)
  - global memory: 40326 MB HBM2
  - global memory bandwidth: 1555 GB/s
  - shared memory/SM: 164 kB
  - #registers/SM: 65536
  - constant memory: 64 kB
  - L1 cache size/SM: 192 kB
  - L2 cache size: 40960 kB
- `NVIDIA H100 SXM5 NVL` (grn008)
  - global memory: 95357 MB HBM3
  - global memory bandwidth: 3000 GB/s
  - shared memory/SM: 228 kB
  - #registers/SM: 65536
  - constant memory: 64 kB
  - L1 cache size/SM: 192 kB
  - L2 cache size: 61440 kB

NVIDIA GH100 SMP

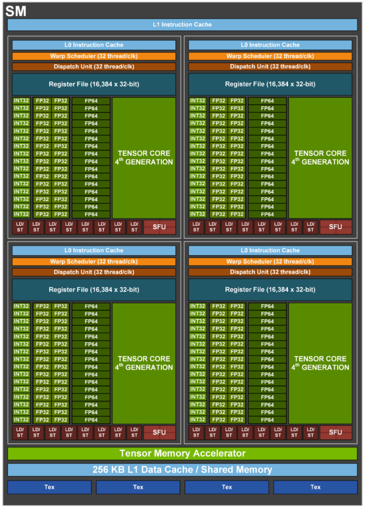

SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

Figure: GH100 SMP.

NVIDIA GH100 Full Device

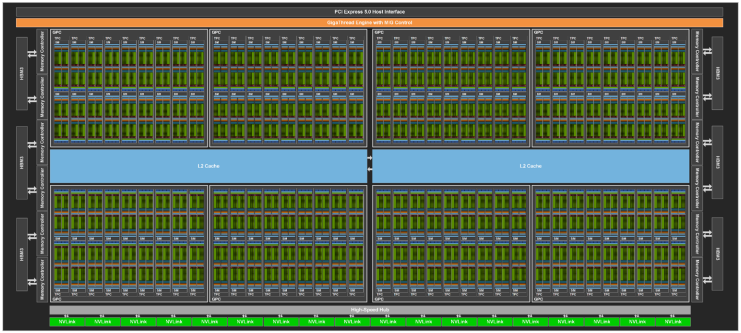


Figure: NVIDIA GH100 Full Device (144 SMPs).

GPGPU & CUDA

- GPU (Graphic Processing Unit):
  orginally developed for graphical applications.
- GP-GPU: General-Purpose GPU, i.e.
  the use of GPUs beyond graphical applications.
  CAVEAT: problem to be reformulated in terms of the graphics API.
- **2006**: NVIDIA introduces the **CUDA**[2] framework
  (**C**ompute **U**nified **D**evice **A**rchitecture)
  - CUDA API: extension of the `C` language.
  - handles the GPU thread level parallelism
  - deals with moving data between CPU and GPU.
  - also support for `C++` , `Fortran`.

[2]The CUDA Toolkit consists of 2 parts:

- CUDA Driver
- CUDA Toolkit (`nvcc`,`nvprof`, . . . , libraries, header files).

GPU Threads - Warps

- Each SMP:
  - generates, schedules, executes threads in batches of 32 threads.
  - **WARP**: a batch of 32 threads
- each thread in a WARP executes the same instructions but runs its own "path".
- if threads within a WARP diverge, the threads become inactive/disabled.

Links

- CUDA Toolkit Documentation
- CUDA C++ Programming Guide
- CUDA C++ Best Practices Guide

GPU devices on lp/kp/np/grn

GPU device type	compute capability
NVIDIA GeForce GTX TITAN X	5.2
Tesla P100-PCIE-16GB	6.0
Tesla P40	6.1
NVIDIA GeForce GTX 1080 Ti	6.1
NVIDIA Titan V	7.0
NVIDIA Tesla V100-PCIE-16GB	7.0
Tesla T4	7.5
NVIDIA GeForce RTX 2080 Ti	7.5
NVIDIA A100-PCIe-40GB	8.0
NVIDIA A100-SXM4-80GB	8.0
NVIDIA A800 40GB Active	8.0

Table: GPU devices on lp/kp/np/grn (10/01/2024)

GPU devices on lp/kp/np/grn (cont.)

GPU device type	compute capability
NVIDIA GeForce RTX 3090	8.6
NVIDIA A40	8.6
NVIDIA RTX A5500	8.6
NVIDIA RTX A6000	8.6
NVIDIA RTX 6000 Ada Generation	8.9
NVIDIA L40	8.9
NVIDIA L40S	8.9
NVIDIA H100 NVL/Deep Dive	9.0

Table: GPU devices on lp/kp/np/grn (10/01/2024)

GPU devices on redwood

GPU device type	compute capability
`NVIDIA GeForce GTX 1080 Ti`	6.1
`NVIDIA A100-SXM4-40GB`	8.0
`NVIDIA A100 80GB PCIe`	8.0
`NVIDIA A30`	8.0
`NVIDIA A40`	8.6
`NVIDIA RTX 6000 Ada Generation`	8.9
`NVIDIA H100 NVL/Deep Dive`	9.0

Table: GPU devices on redwood (10/01/2024)

Questions?

Thank you!
Any questions?