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

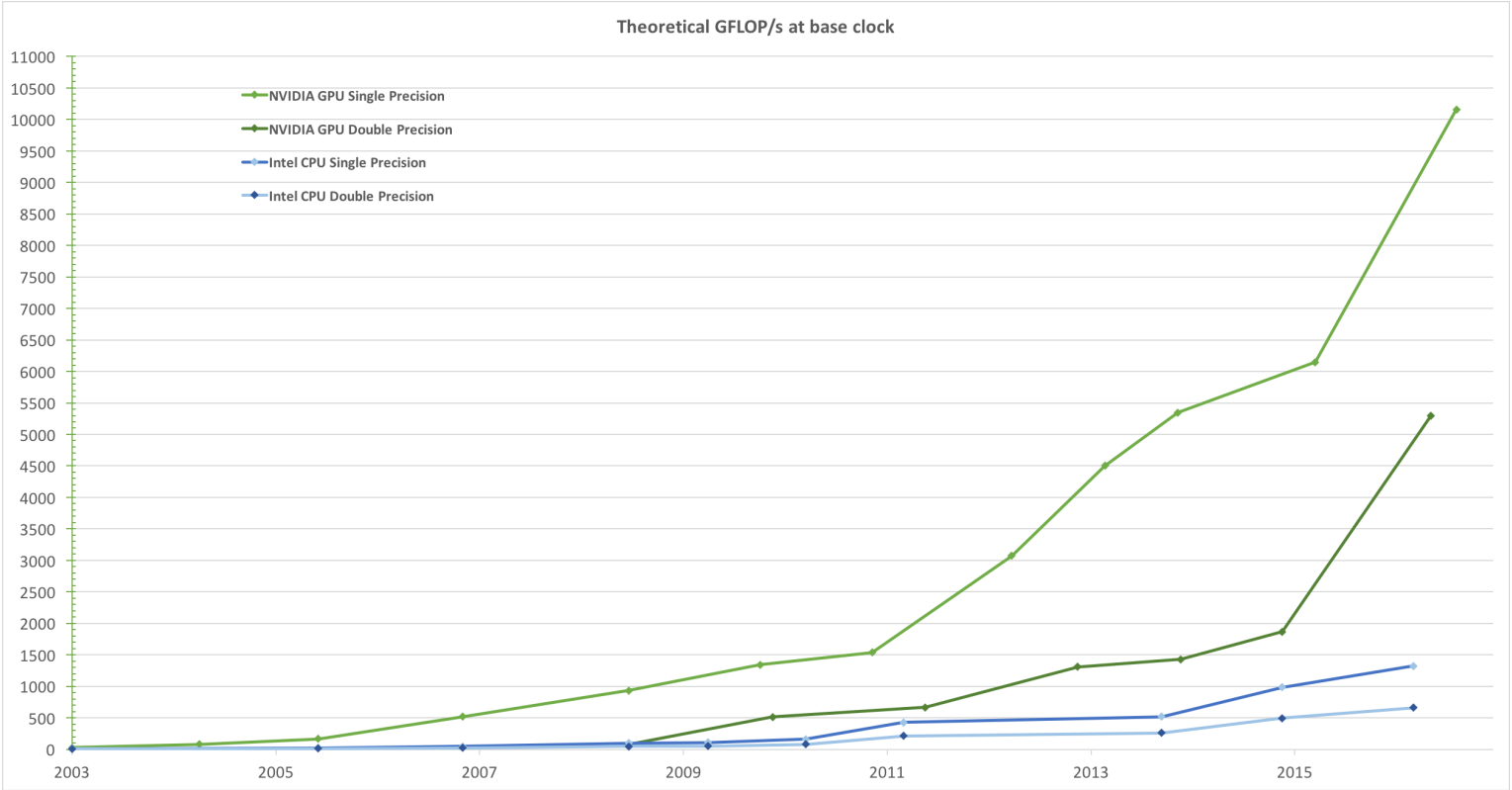
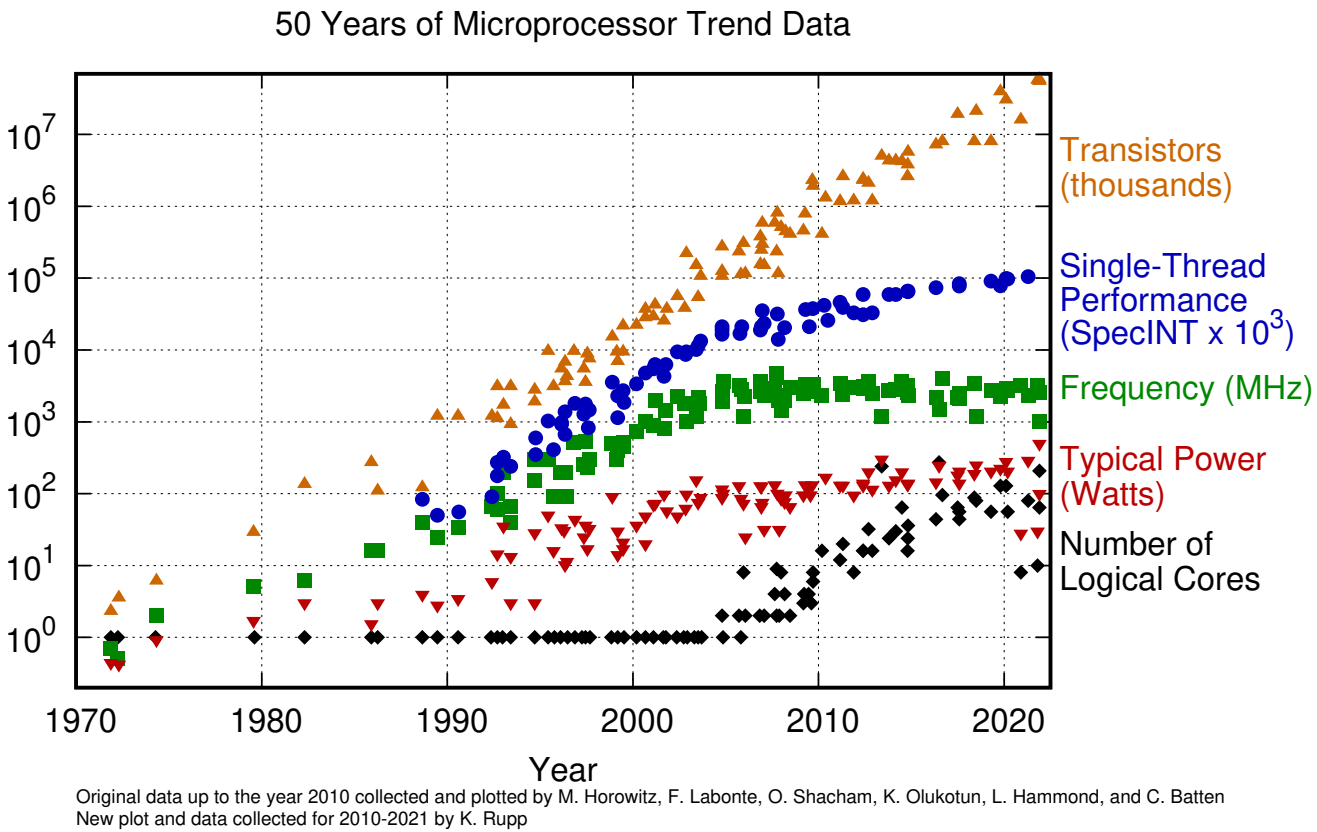


Figure: Theoretical GFLOP/s: GPUs vs. CPUs.[a]

[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

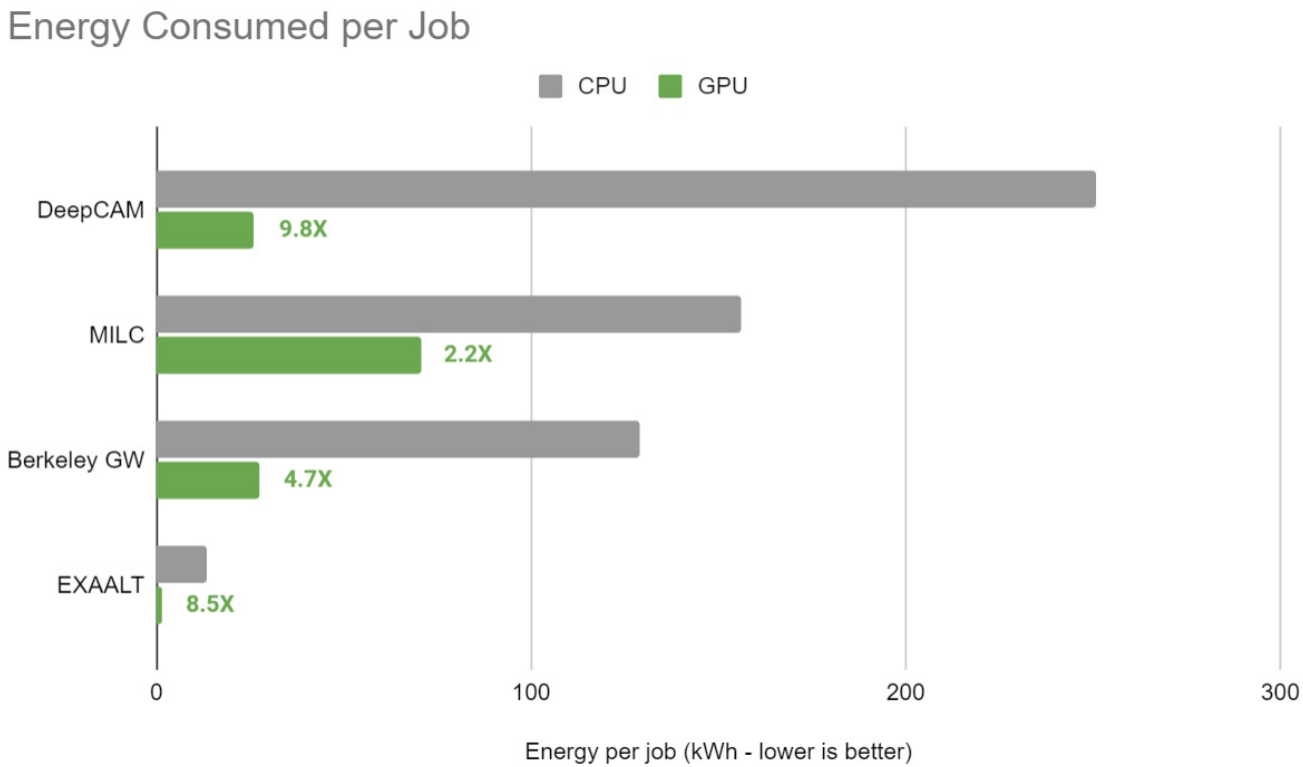


Figure: Energy efficiency per job (NERSC).[a]

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

Streaming Multiprocessor (SMP)

- GPU device connected to the CPU by a PCIe bus.
- each GPU device contains an array (**x**) of Streaming Multiprocessors **(SMP)**.
- each SMP has:
  - a Single-Instruction Multiple-Thread (`SIMT`) Architecture.
  - contains **y** regular cores and [**z** tensor cores].
- scalable: newer generations: increase of **x**, **y** and [**z**], e.g.:
  - `NVIDIA A100-PCIE-40GB` (*notch293*)
    - global memory: 40 GB.
    - 108 SMPs, 64 Cores/SMP, 4 Tensor Cores/SMP.
    - GPU Max. Clock Rate: 1.41 GHz.
  - `NVIDIA H100 SXM5 NVL` (*grn008*)
    - global memory: 93 GB.
    - 132 SMPs, 128 Cores/SMP, 4 Tensor Cores/SMP.
    - GPU Max. Clock Rate: 1.78 GHz.

NVIDIA GH100 SMP

Figure: GH100 SMP.

NVIDIA GH100 Full Device

Figure: NVIDIA GH100 Full Device (144 SMPs).

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.

Types of GPU memory

- global memory (largest and slowest)
- shared memory: allocated per thread block & low latency
- constant memory: cached, read-only
- registers: fast, on-chip memory (exclusive to each thread).

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**[1] 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`.

[1] The CUDA Toolkit consists of 2 parts:

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

Links

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

Questions?

Thank you!
Any questions?