Feb 24^{th} – 26^{th}, 2026

Friedrich-Alexander-Universität Erlangen-Nürnberg

PPHPS – GPU Programming with OpenMP Target Offloading

Sebastian Kuckuk, *Erlangen National High Performance Computing Center (NHR@FAU)*

GPUs in HPC – Motivation

- Promises
  - Massive parallelism and performance
  - Good performance in relation to energy (FLOPs per Watt)

- Already widespread in the HPC landscape
  - c.f. Top500 list (https://www.top500.org/lists/top500/2025/11/)
    - 9 out of the top 10 supercomputers are equipped with GPUs
    - 4 NVIDIA (2 x GH200, H100, A100)
    - 4 AMD (MI300A, 3 x MI250X)
    - 1 Intel (GPU Max Series)

Helma is #58
SuperMUC-NG is #91
SuperMUC-NG Ph. 2 is #108
Alex is #322
Fritz is #350

- Where does the performance come from?

GPU Performance

- If GPUs are mainly about performance – what is performance?

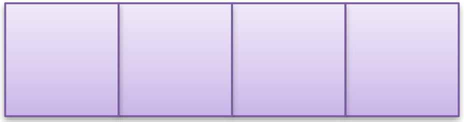
- Two primary factors
  - How fast can the meaningful computation be done?
    Usually given as computational throughput, e.g. FLOP/s

  - How fast can the data be transferred to where the computation is happening (and back)?
    Usually given as sustained bandwidth, e.g. GB/s

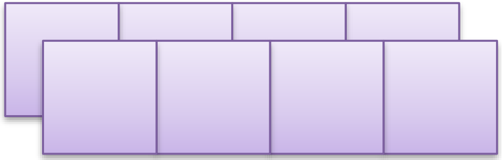
- What are GPUs doing differently?

GPU vs CPU

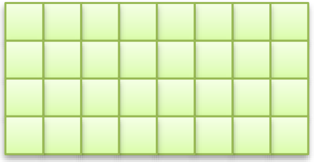
- CPUs have few (~100) powerful cores



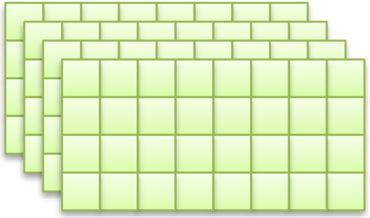
- Organized in sockets



- GPUs have many simplistic 'cores' (10 000s)

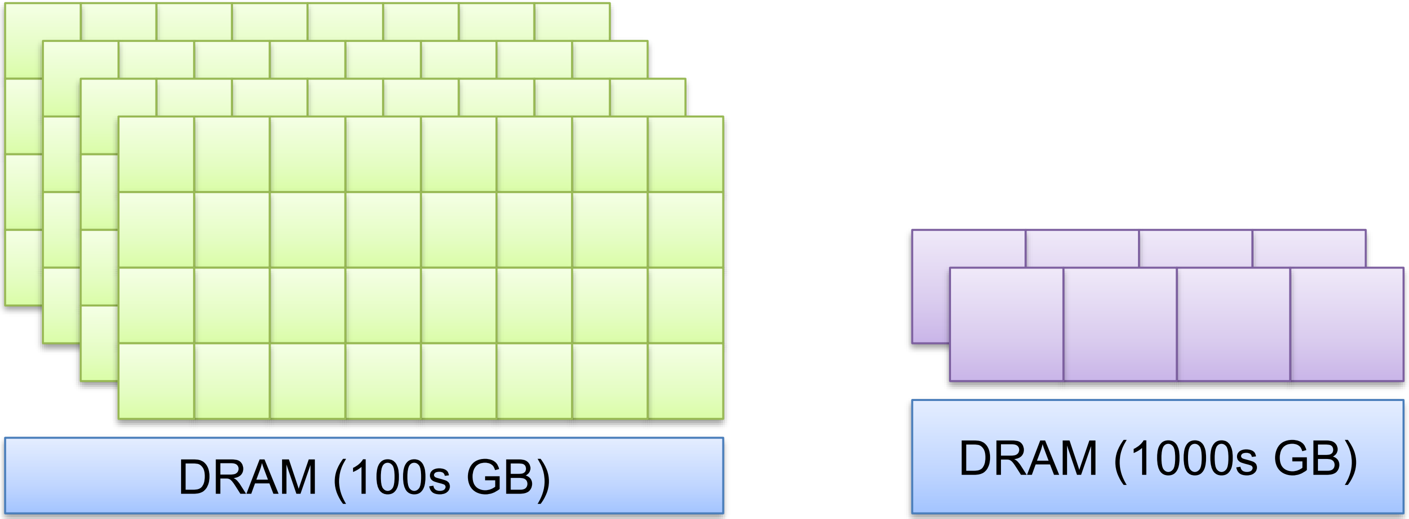


- Organized in 100s of streaming multiprocessors (SMs)



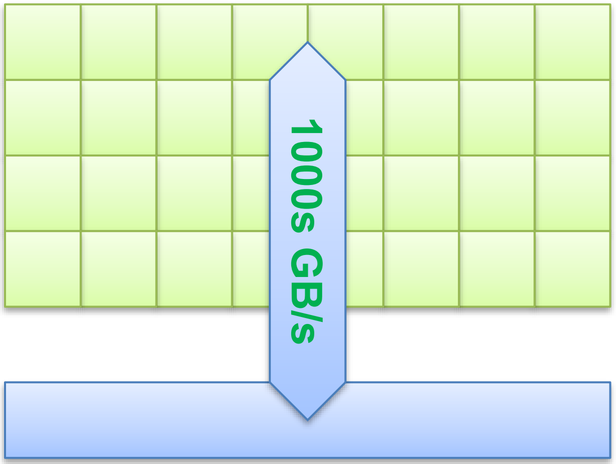
GPU vs CPU

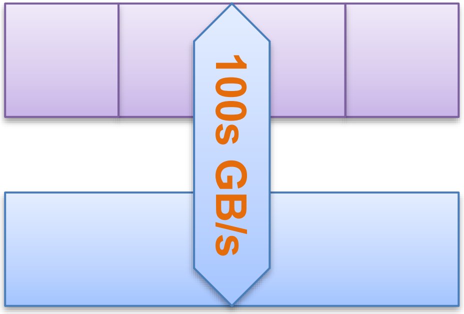
- Both CPU and GPU have a distinct main memory (for most architectures)



GPU vs CPU

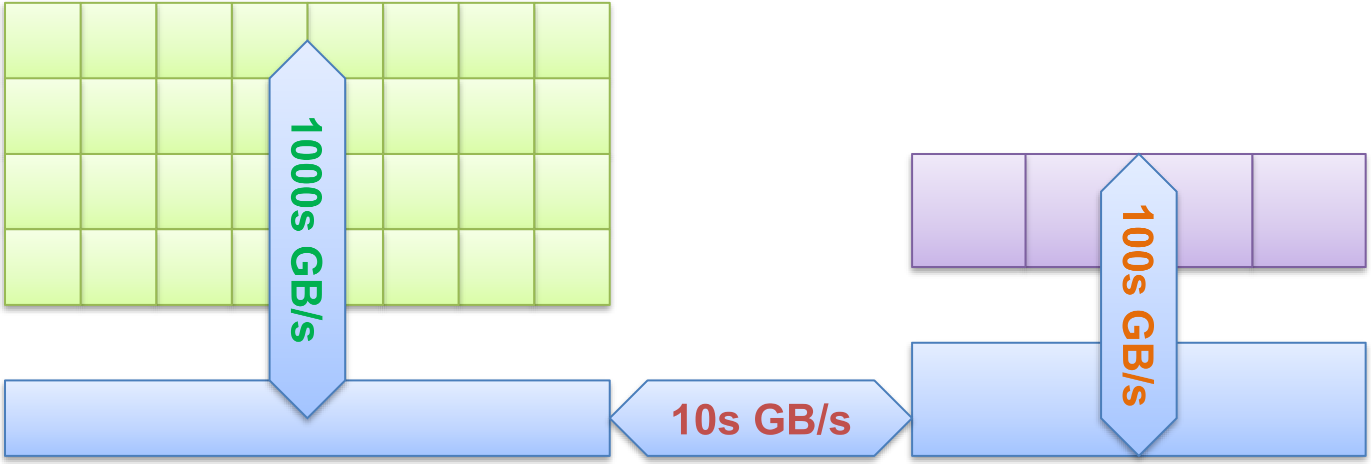
- The memory of GPUs
  - Is optimized for bandwidth
  - Has a high latency
- The memory of CPUs
  - Is optimized for latency
  - Has a high capacity





GPU vs CPU

- Both CPU and GPU have a distinct main memory (for most architectures)
- They communicate via slow interconnects (for most architectures)



Introduction – CPU vs GPU

- Why not use GPUs exclusively?
  - Let’s consider a simple vector copy benchmark to assess sustained bandwidths
- One A100 40GB GPU (Alex) vs one Sapphire Rapids node (Fritz)
  - ~ 1.3 TB/s vs ~ 260 GB/s => 5x faster
- But: One SM of one A100: ~ 90 GB/s => 3x slower
- Serial execution
  - ~ 0.3 GB/s vs ~ 20 GB/s => 67x slower

Introduction – CPU vs GPU

- Why not use GPUs exclusively?

- GPUs deal best with
  - Massive parallelism
    at least 10 000s of threads to saturate computation
    at least 100 000s of threads to saturate memory

  - Structured computations
    Ideally each thread does the same operation but on different parts of a structured data set

- CPUs deal much better with unstructured, low-parallelism computations

GPU Programming Approaches

Options are plentiful

1. Avoid GPU programming all together
   - GPU-accelerated libraries

2. Let the compiler do the job
   - Modern C++
   - Pragma-based approaches (OpenMP, OpenACC)

3. Get your hands dirty and do the technical work
   - CUDA, HIP, oneAPI, …

4. Do all the work – but now with added performance portability
   - Software layers (Kokkos, …)

GPU Programming Approaches

Options are plentiful

1. Avoid GPU programming all together
   - GPU-accelerated libraries

2. Let the compiler do the job
   - Modern C++
   - **Pragma-based approaches** (**OpenMP**, OpenACC)

3. Get your hands dirty and do the technical work
   - CUDA, HIP, oneAPI, …

4. Do all the work – but now with added performance portability
   - Software layers (Kokkos, …)