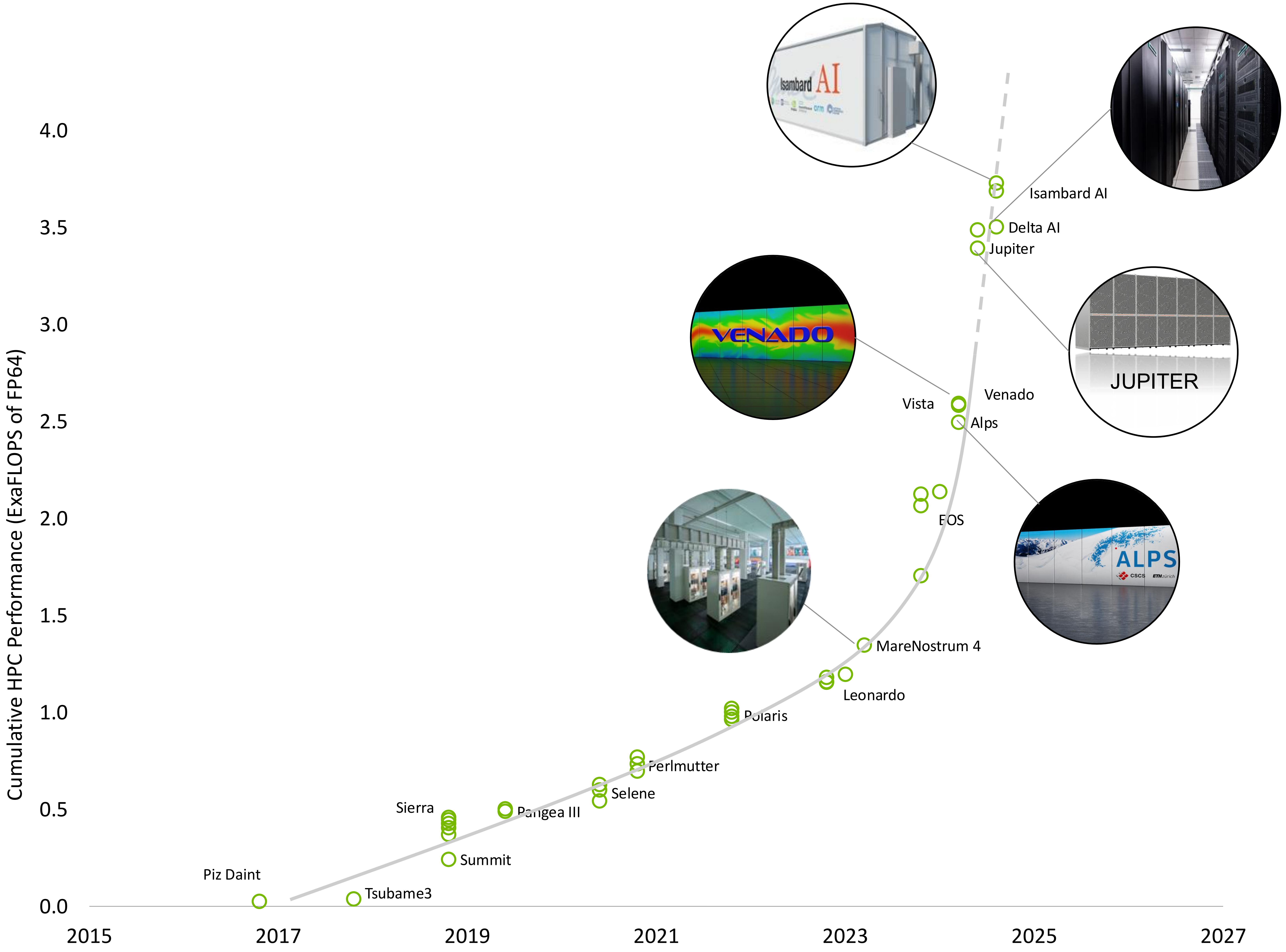
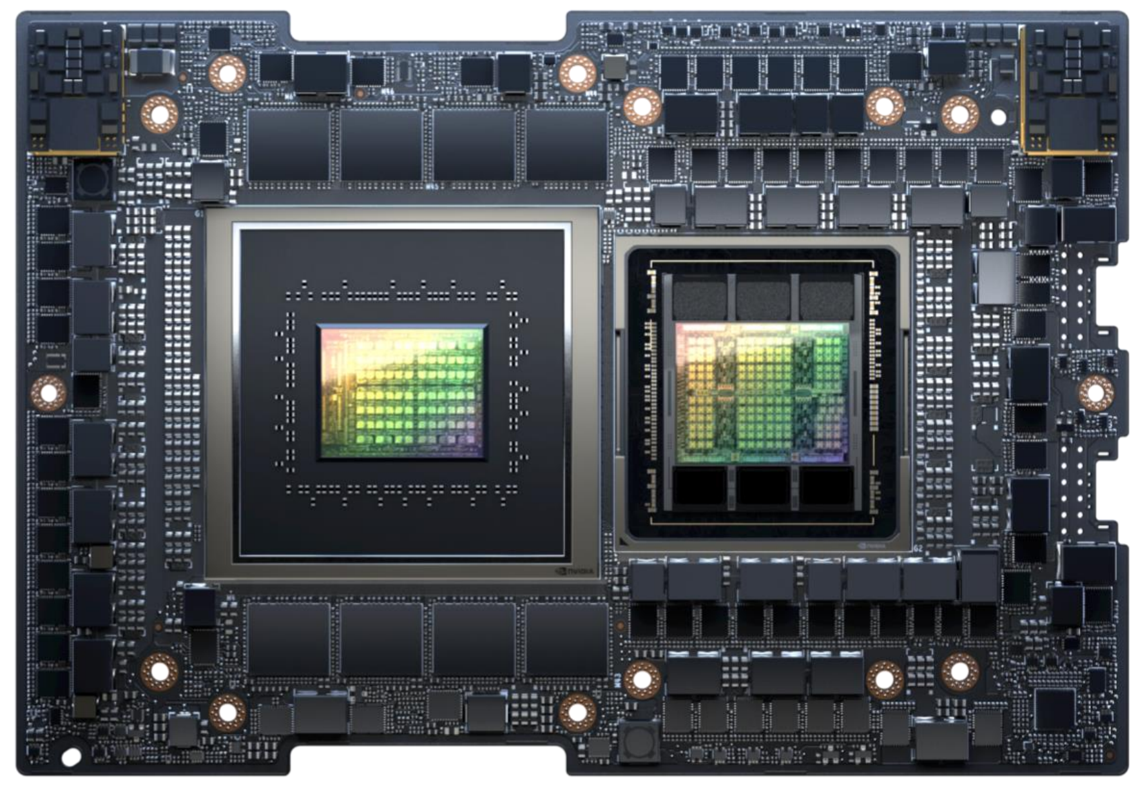
Lattice QCD (and more) on the NVIDIA Grace Hopper Platform

Mathias Wagner, Lattice 2024

Next-Gen Supercomputing Datacenter

4 ExaFLOPs of HPC Performance Driving Scientific Innovation





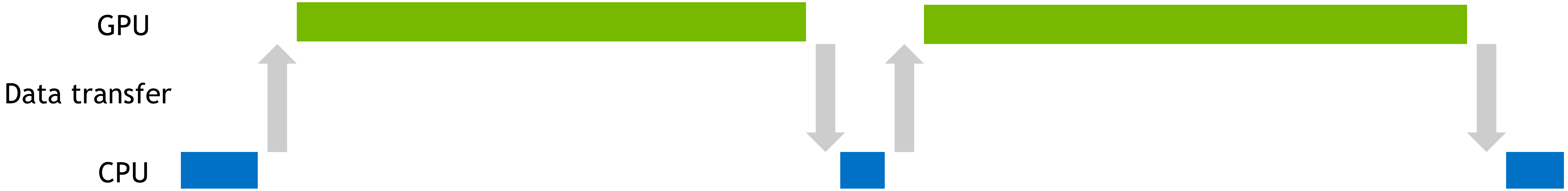
1.7 Exaflops Grace Hopper
Coming online 2024

How much of your WORKLOAD is running on GPUs?

Application on Accelerated Systems

Fully GPU Accelerated

- Compute almost fully on the GPU with data in GPU memory

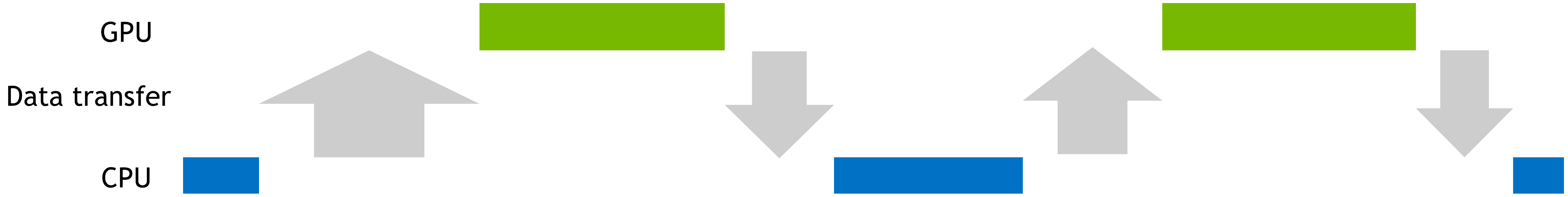


- Little to no limitation from CPU and data transfers

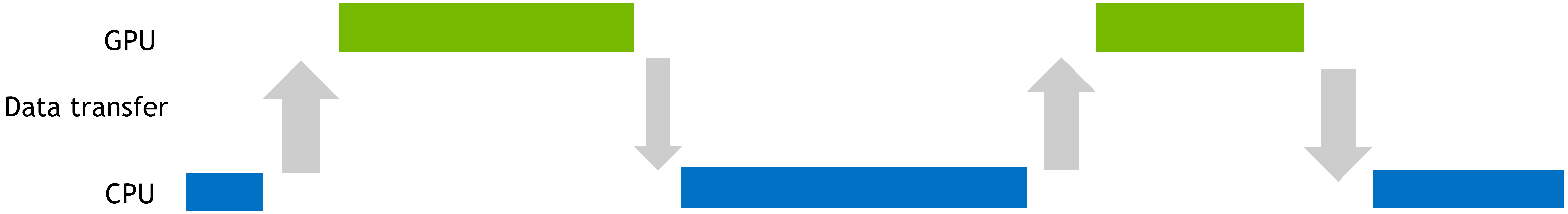
Application on Accelerated Systems

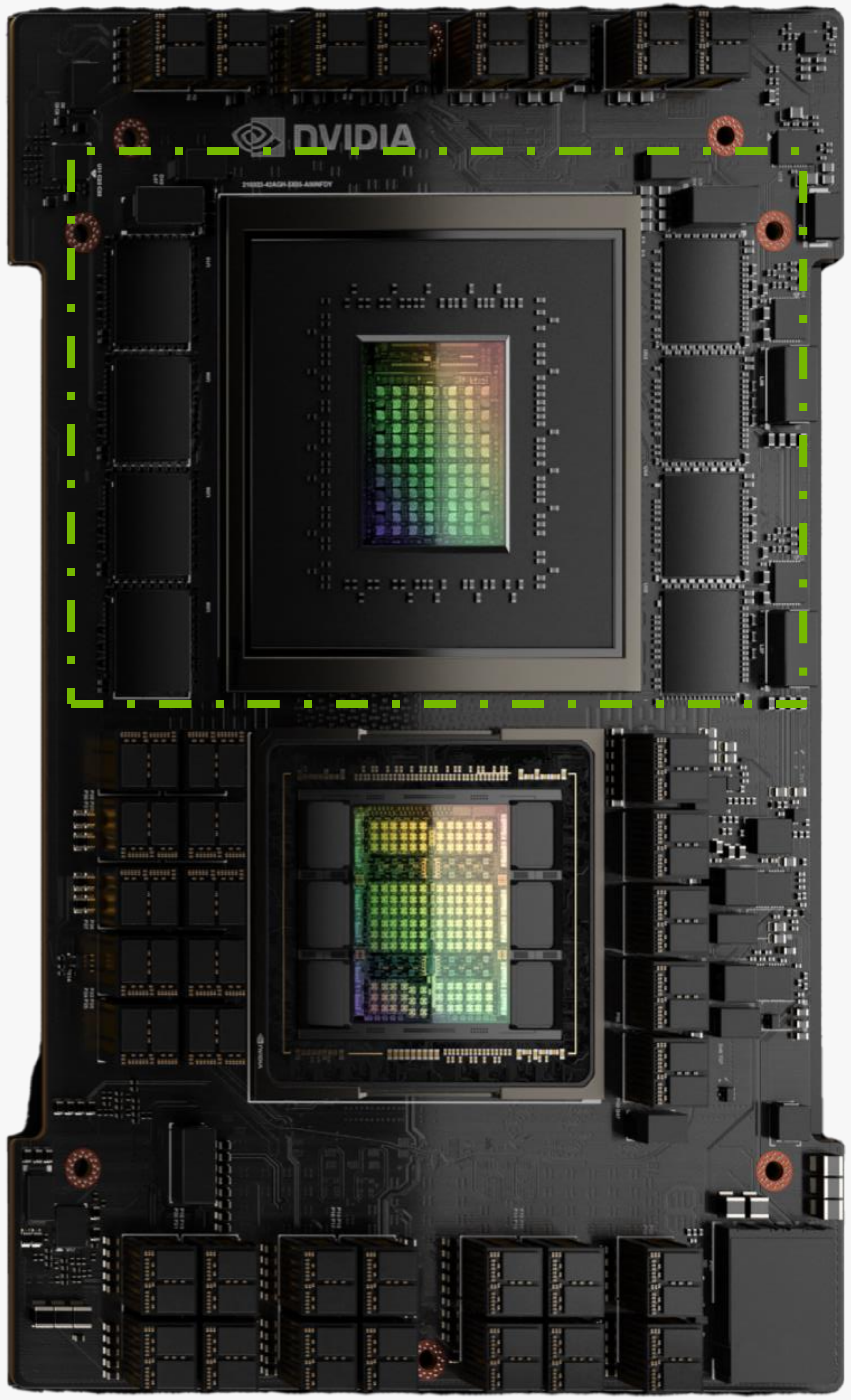
Partially GPU Accelerated

- As GPUs become faster applications become increasingly limited by non-GPU factors, e.g.
- mostly data transfer (PCIe) limited



- mostly CPU limited

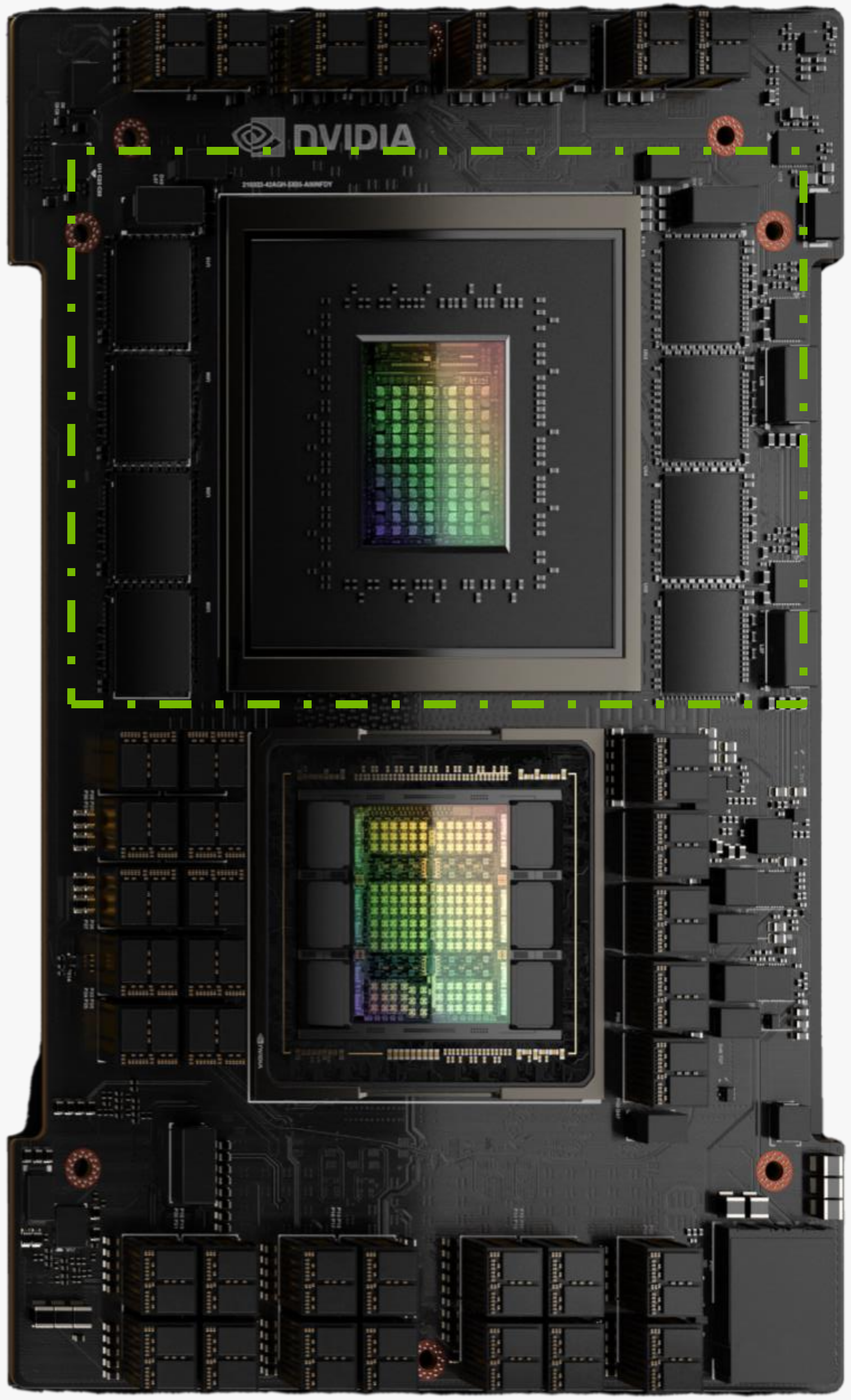




NVIDIA Grace Hopper Superchip

“super” - more than a “chip”

NVIDIA CPU + NVIDIA GPU w/o compromises

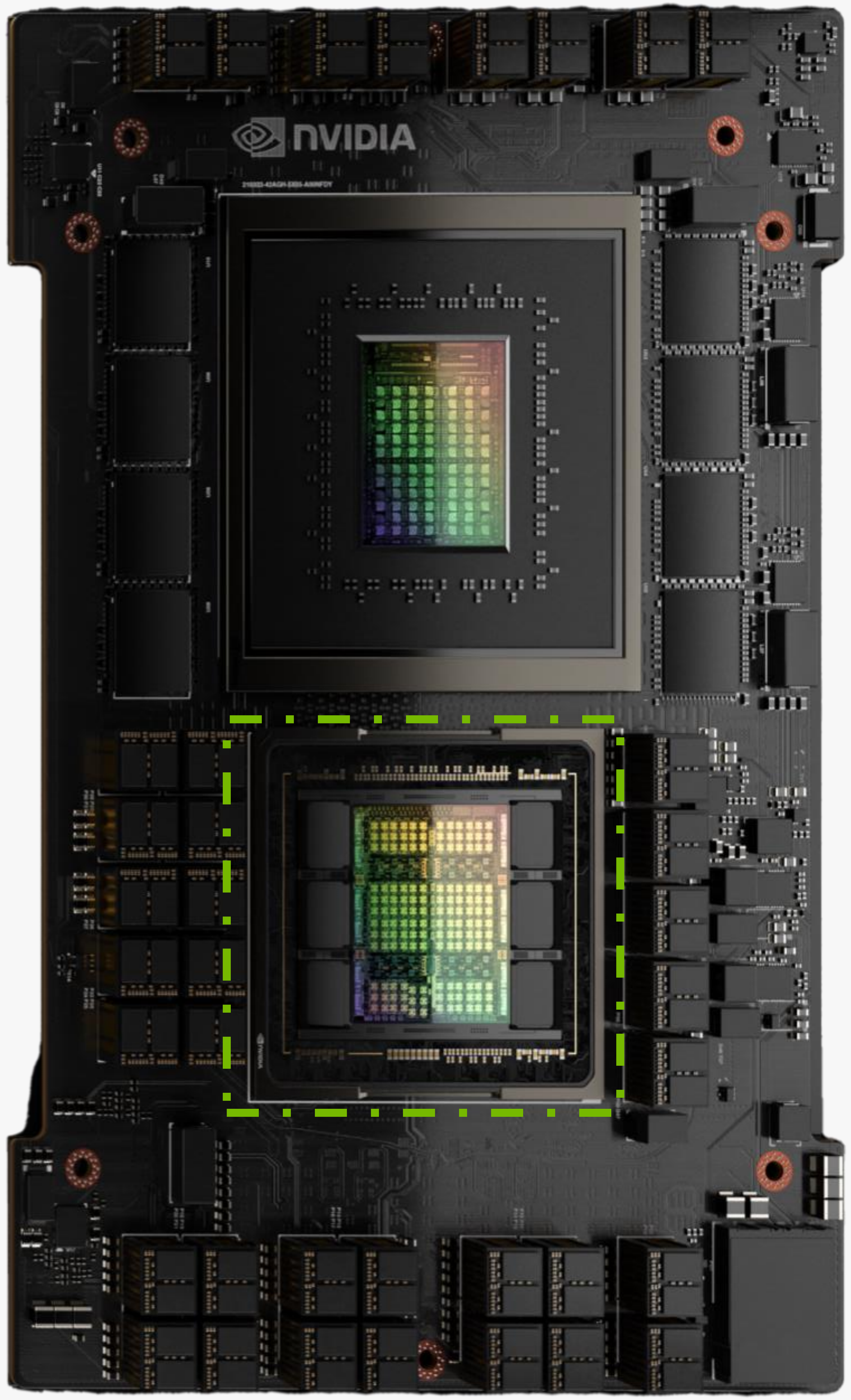


NVIDIA Grace Hopper Superchip

"super" - more than a "chip"

NVIDIA CPU + NVIDIA GPU w/o compromises

- **NVIDIA Grace CPU**
  - 72 Arm-v9 Neoverse V2 CPU cores with SVE2.
    - → Throughput: 3.6 TFLOP/s
  - Memory:
    - →High capacity: ≤ 480 GB LPDDR5X
    - →High System Memory bandwidth: ≤ 500 GB/s



NVIDIA Grace Hopper Superchip

“super” - more than a “chip”

NVIDIA CPU + NVIDIA GPU w/o compromises

- **NVIDIA Grace CPU**
  - 72 Arm-v9 Neoverse V2 CPU cores with SVE2.
    - → Throughput: 3.6 TFLOP/s
  - Memory:
    - →High capacity: ≤ 480 GB LPDDR5X
    - →High System Memory bandwidth: ≤ 500 GB/s

- **NVIDIA Hopper GPU**
  - →High throughput: 60 TFLOP/s
  - Memory:
    - → Capacity: 96 GB HBM3 / 144 GB HBM3e
    - → Extreme bandwidth ≤ 4000 GB/s / 5000 GB/s
  - ≤ 18x NVLink 4 → 900 GB/s
  - → Threads are threads

NVIDIA Grace Hopper Superchip

Soul is the new **NVLink-C2C** CPU ←→ GPU interconnect

- **Memory consistency**: ease of use
  - → *All* threads – GPU and CPU – access system memory: C++ `new`, `malloc`, `mmap`'ed files, atomics, …
  - → Fast automatic page migrations
  - → Threads cache peer memory → Less migrations

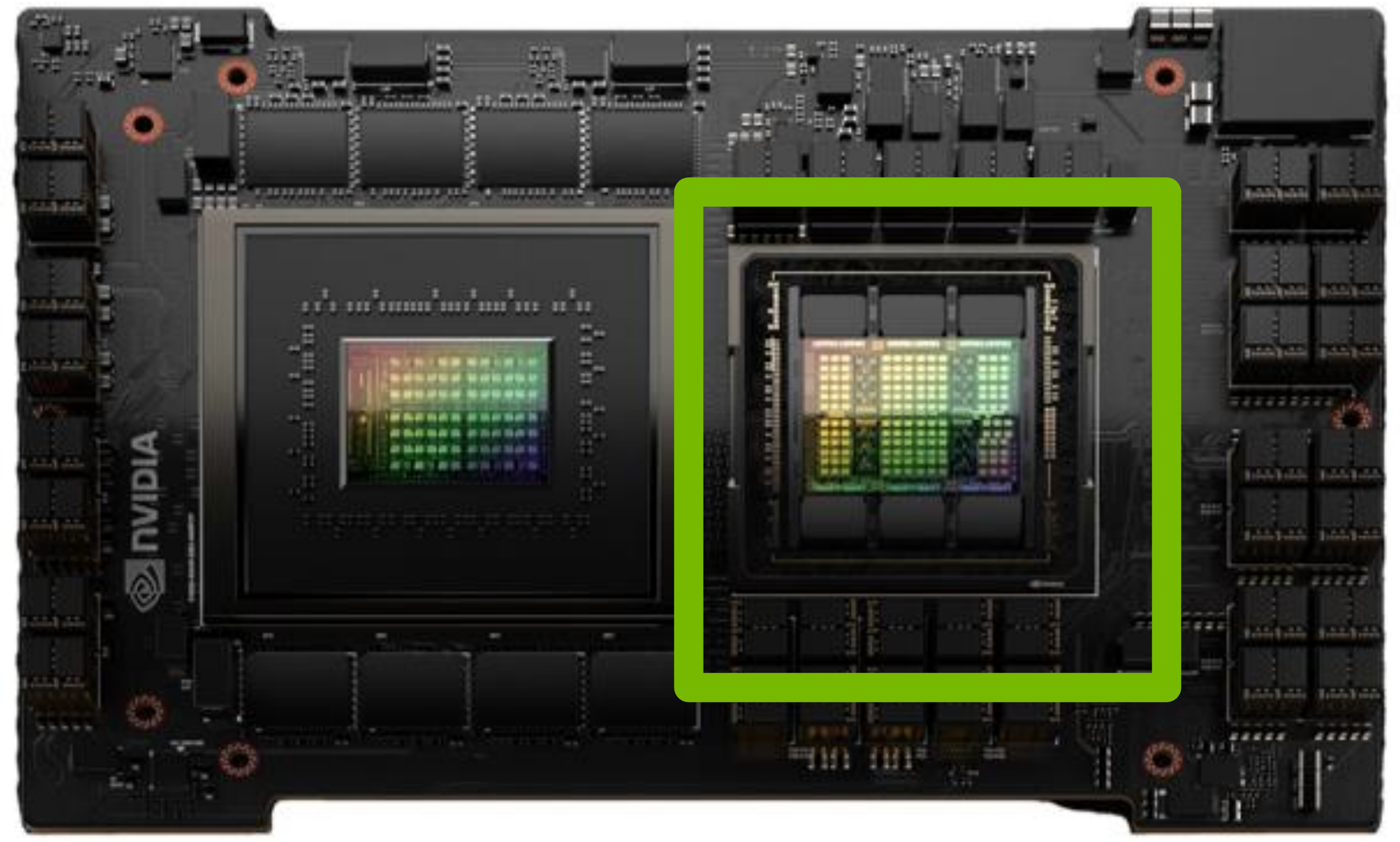
- **High-bandwidth**: 900 GB/s (same as peer NVLink 4)
  - → GPU reads or writes local/peer LPDDR5X at ~peak BW

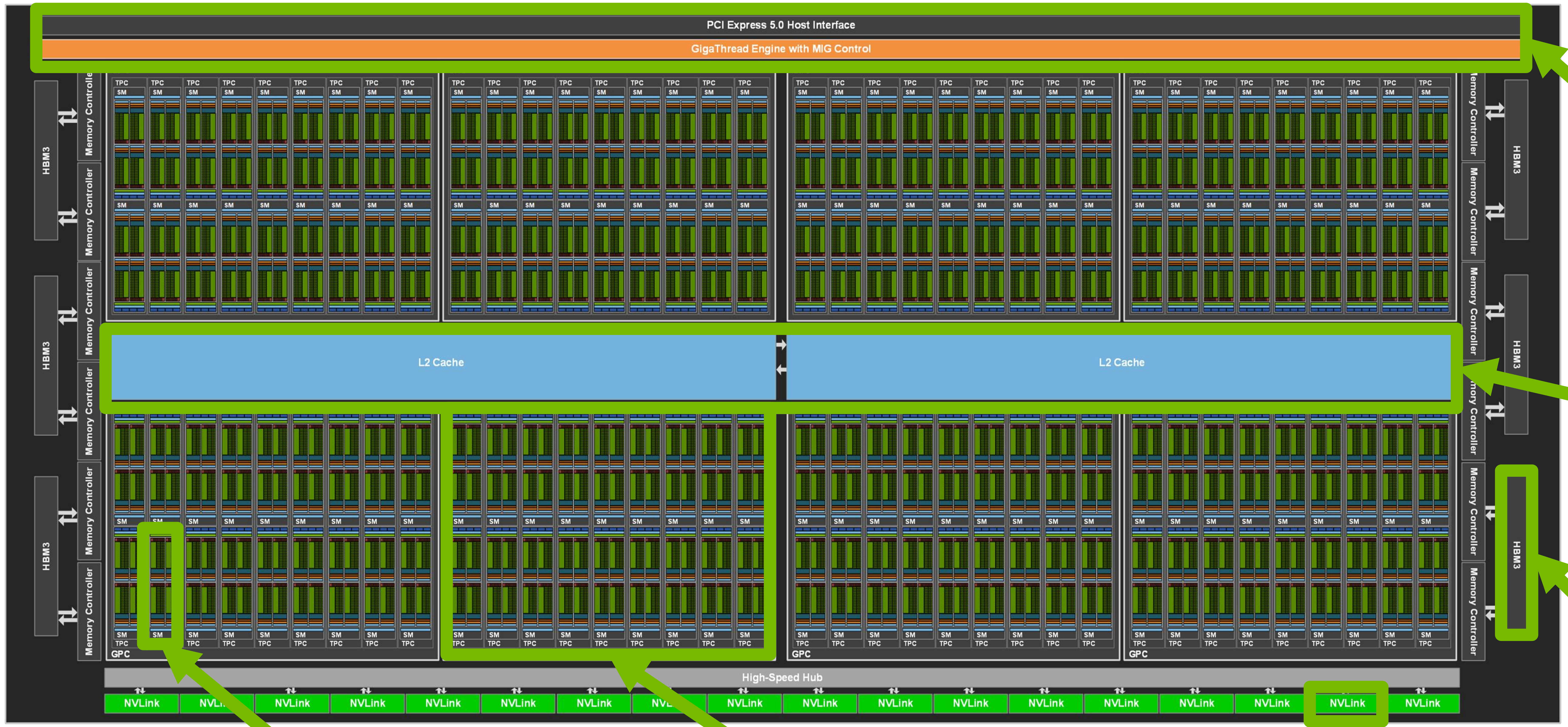
- **Low-latency**: GPU→HBM latency
  - →GPU reads or writes LPDDR5X at ~HBM3 latency

For all threads in the system
memory tastes like memory
expected behavior + latency + bandwidth.

Hopper Architecture

H100 GPU Key features




PCI Express 5.0 Host Interface
GigaThread Engine with MIG Control
HBM3
Memory Controller
TPC
SM
GPC
L2 Cache
High-Speed Hub
NVLink

2nd Gen Multi-Instance GPU
Confidential Computing
PCIe Gen5

Larger 60 MB L2

96GB HBM3, 4 TB/s bandwidth

132 SMs
4th Gen Tensor Core

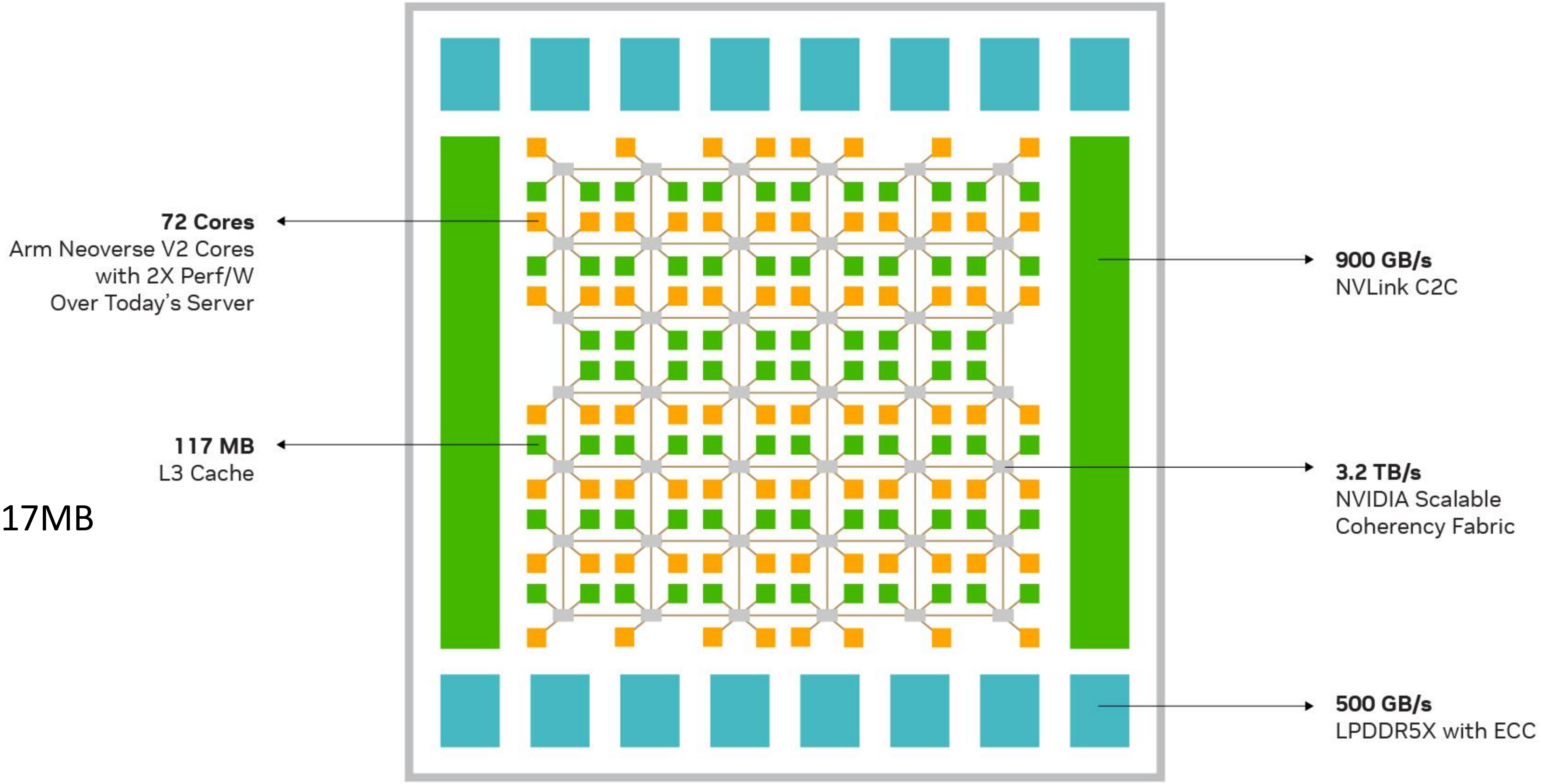
Thread Block Clusters

4th Gen NVLink
900 GB/s total bandwidth

NVIDIA GRACE CPU

NVIDIA Scalable Coherency Fabric and distributed cache design

- **72 Arm Neoverse V2 Cores**
  - 4x128b SVE2 SIMD units per core
- **Single Die:** single NUMA
- **3.16 Ghz Base Clock**
  - **2.7 GHz Vector Clock**
- **3,225.6 GB/s Bisection Bandwidth**
- **Scalable Coherency Fabric:** Shared, uniform 117MB of L3 cache for entire chip.
- **LPDDR5x**: up to 500GB/s memory bandwidth. Local caching of remote die memory.

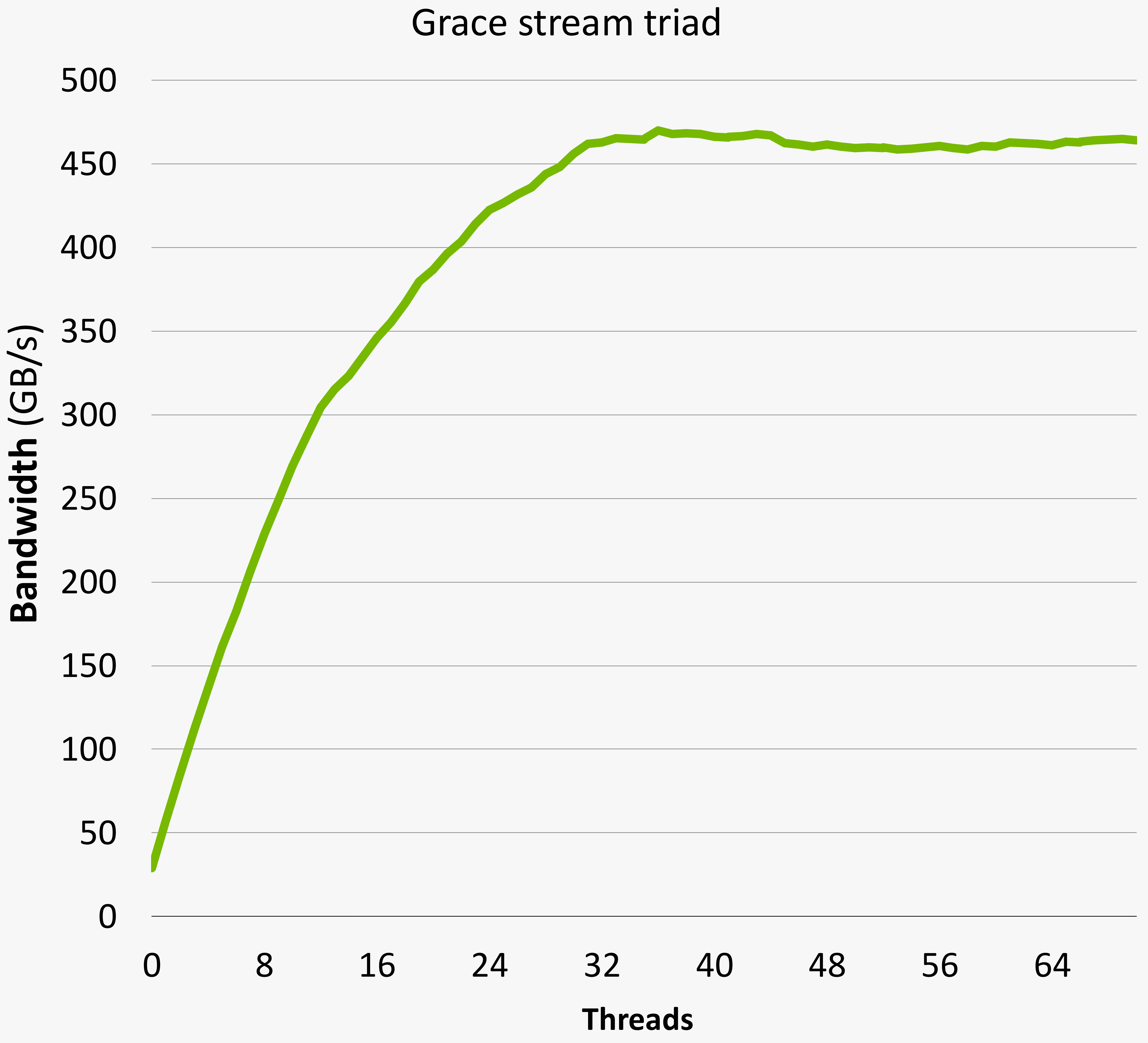


Example possible fabric topology for illustrative purposes

Grace Memory Subsystem

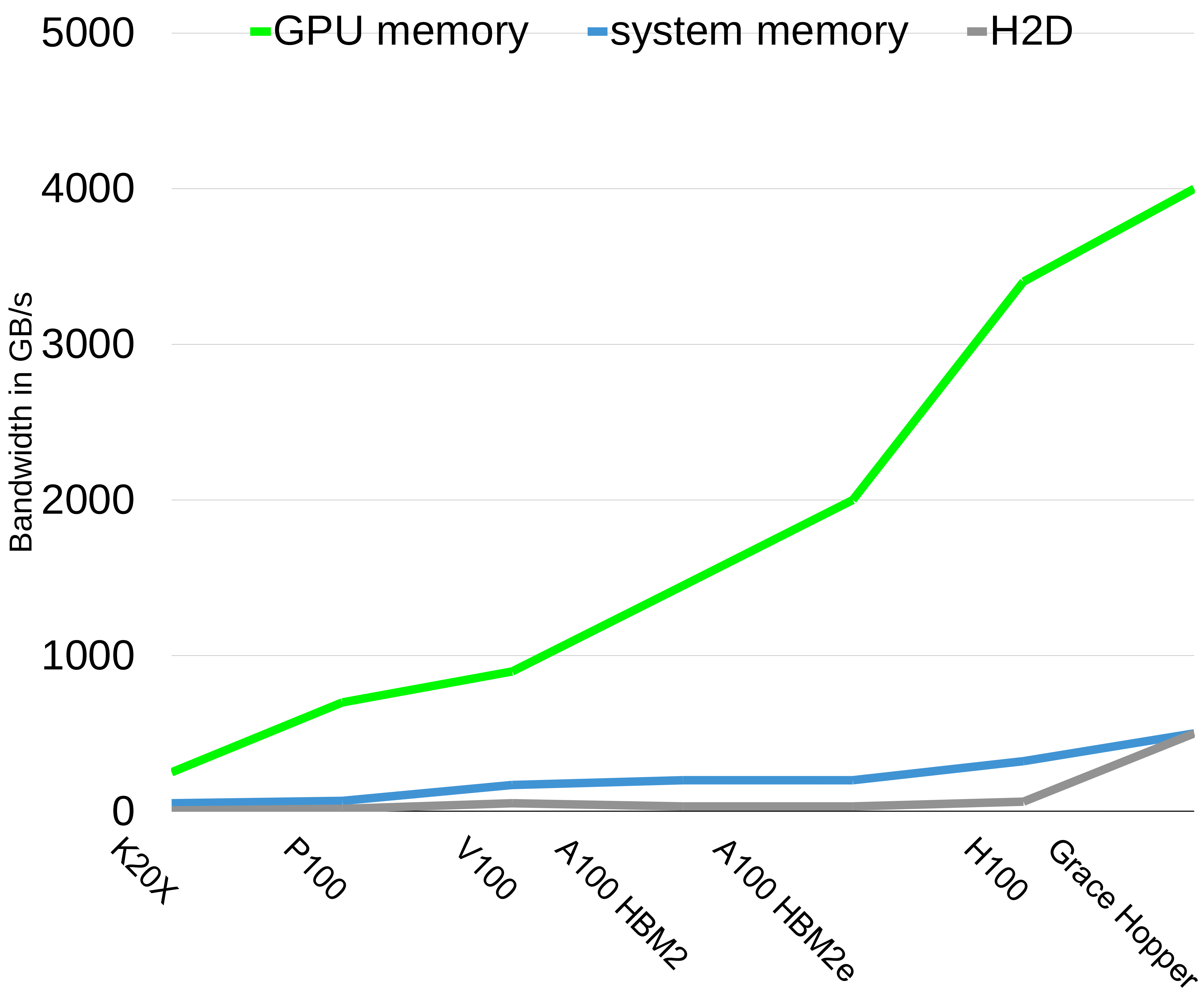
Per 72C Grace SoC

- Separate L1 data and instruction caches per core
  - L1 instruction memory system
    - 64KB, 4-way set associative, 64B cache line
  - L1 data memory system
    - 64KB, 4-way set associative, 64B cache line
- Private, unified data and instruction L2 cache per core
  - 1MB , 8-way set associative
- Scalable Coherency Fabric:
  Shared, uniform 117MB of L3 cache for entire chip.
- LPDDR5x: up to 500GB/s memory bandwidth
  - 120GB / 240GB capacity: 500 GB/s
  - 480GB capacity: 375 GB/s

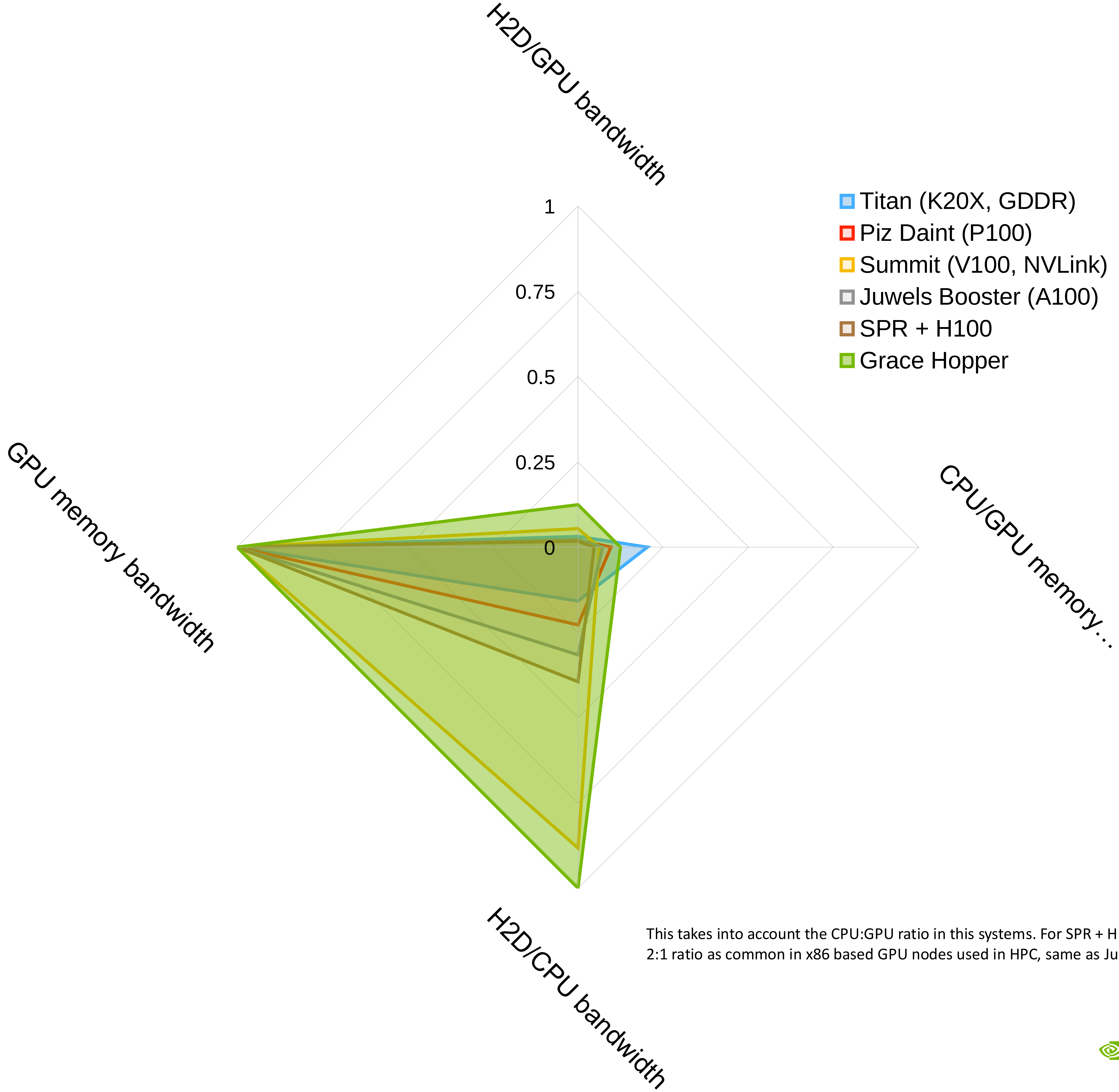


Widening the bottlenecks

How much do transfer and system memory bandwidth limit your application?



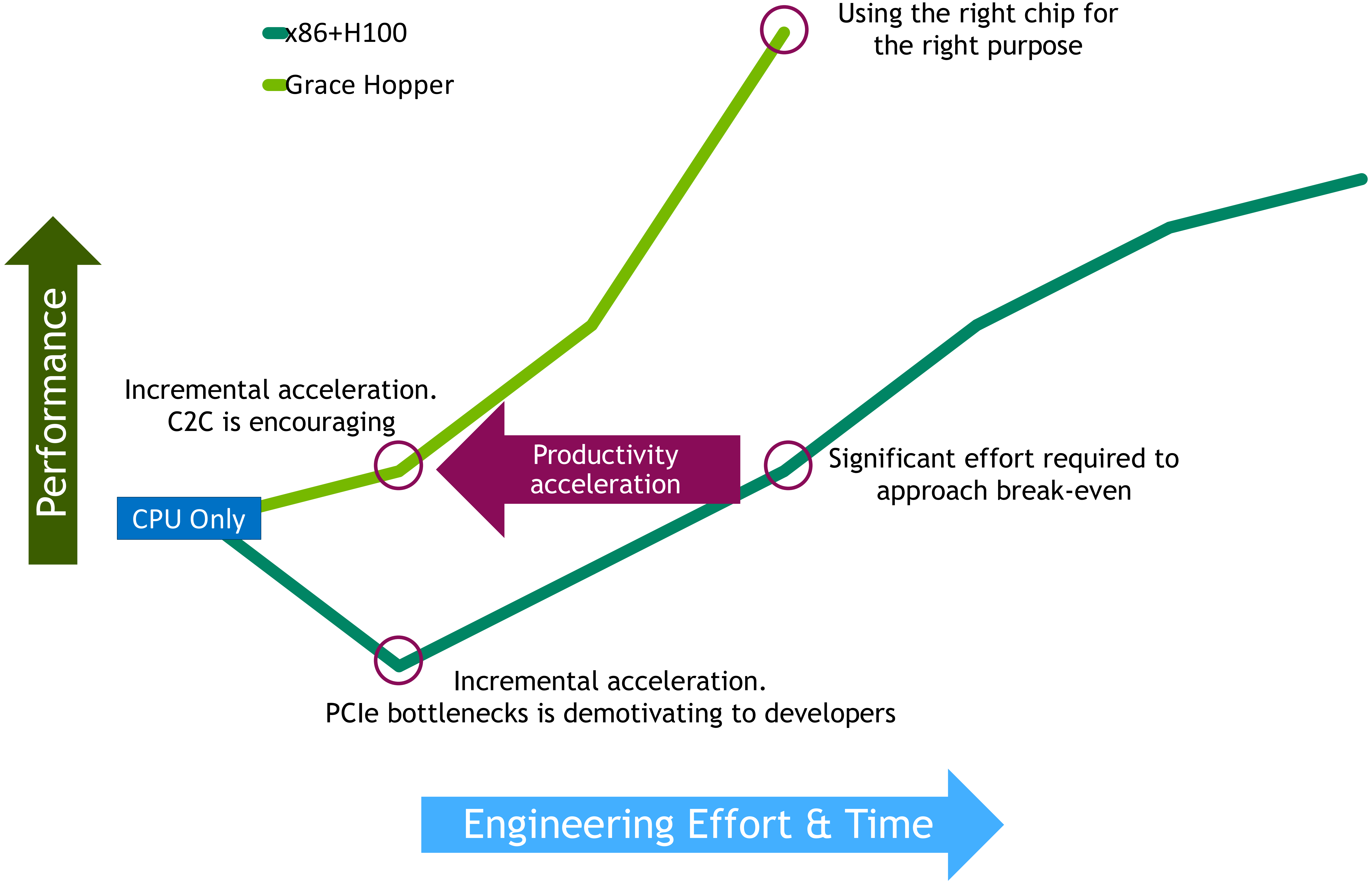
Assumes a typical CPU used in the timeframe.



This takes into account the CPU:GPU ratio in this systems. For SPR + H100 we assume a 2:1 ratio as common in x86 based GPU nodes used in HPC, same as Juwels Booster..

Developer Velocity with Grace Hopper

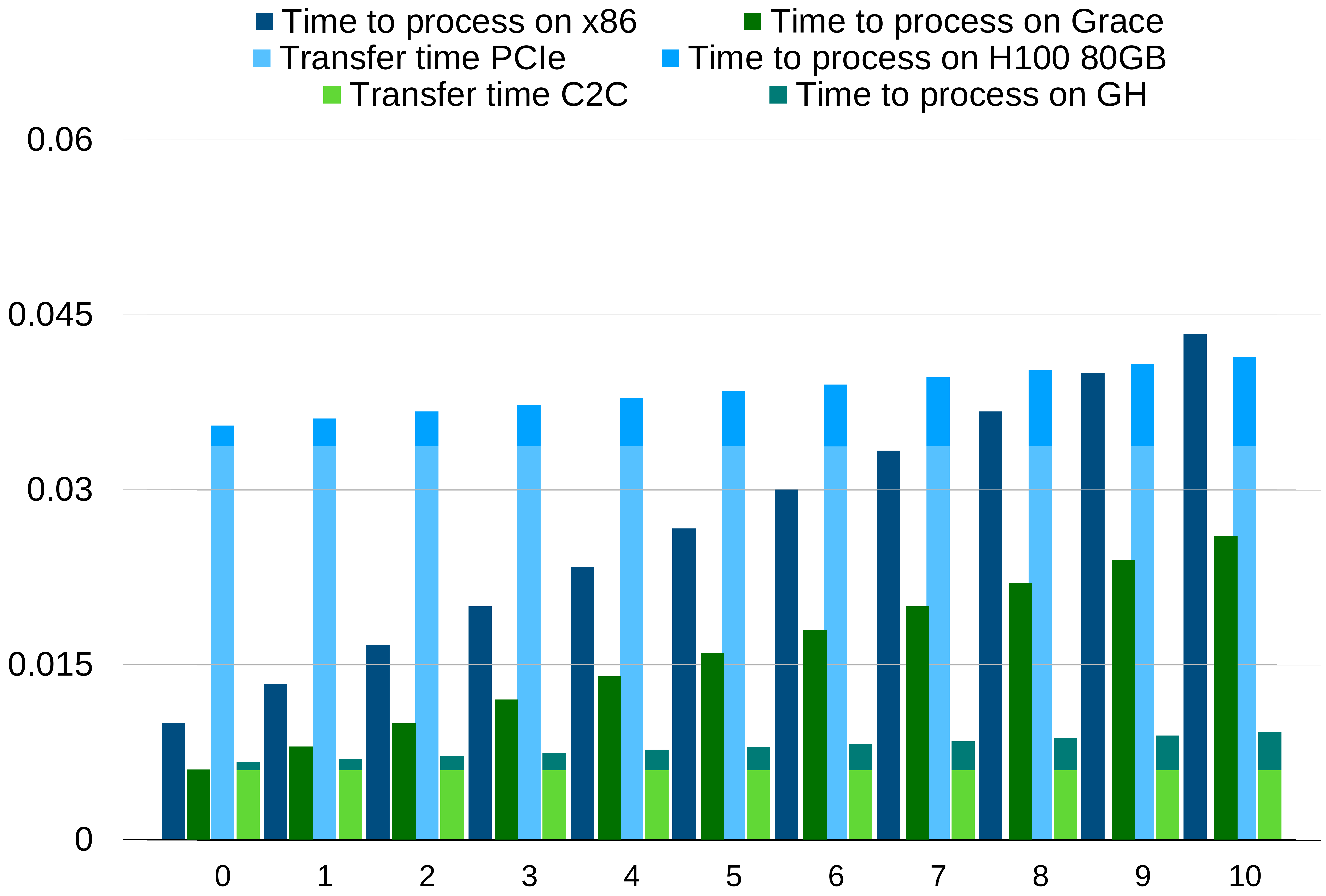
Accelerating the path to accelerated computing



Shifting the break-even point

Further lowering the barrier to GPU acceleration with C2C

- Assume a memory-bandwidth bound workload
  - Idealized: time ~ data size / bandwidth
- Process 3+x GB of data on the CPU
- For GPU processing
  - Transfer 3 GB from / to GPU
  - Process 3+x GB of data on the GPU



Getting ready for Grace-Hopper

Recompile and Run

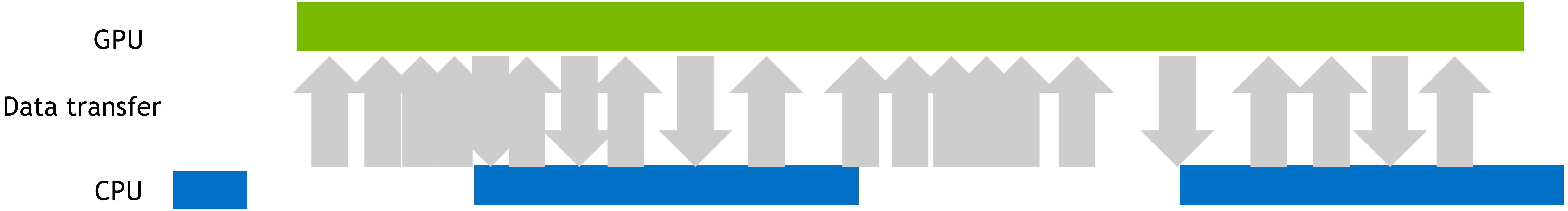
- Currently existing applications do not need to be changed
  - Recompile the application for ARM Neoverse-V2 (Grace) and sm_90 (Hopper)*
  - Benefit from more bandwidth everywhere

- Accelerate existing applications
  - Easier to port than ever
  - Large selection of programming models and language available
  - Hardware coherency
  - Obtain overall speedup even for partially ported applications with the Grace CPU and C2C
  - Large selection of tools (NVIDIA tools and 3rd party) available
  - Balanced architecture results in fewer Amdahl’s limiters

Application on Accelerated Systems

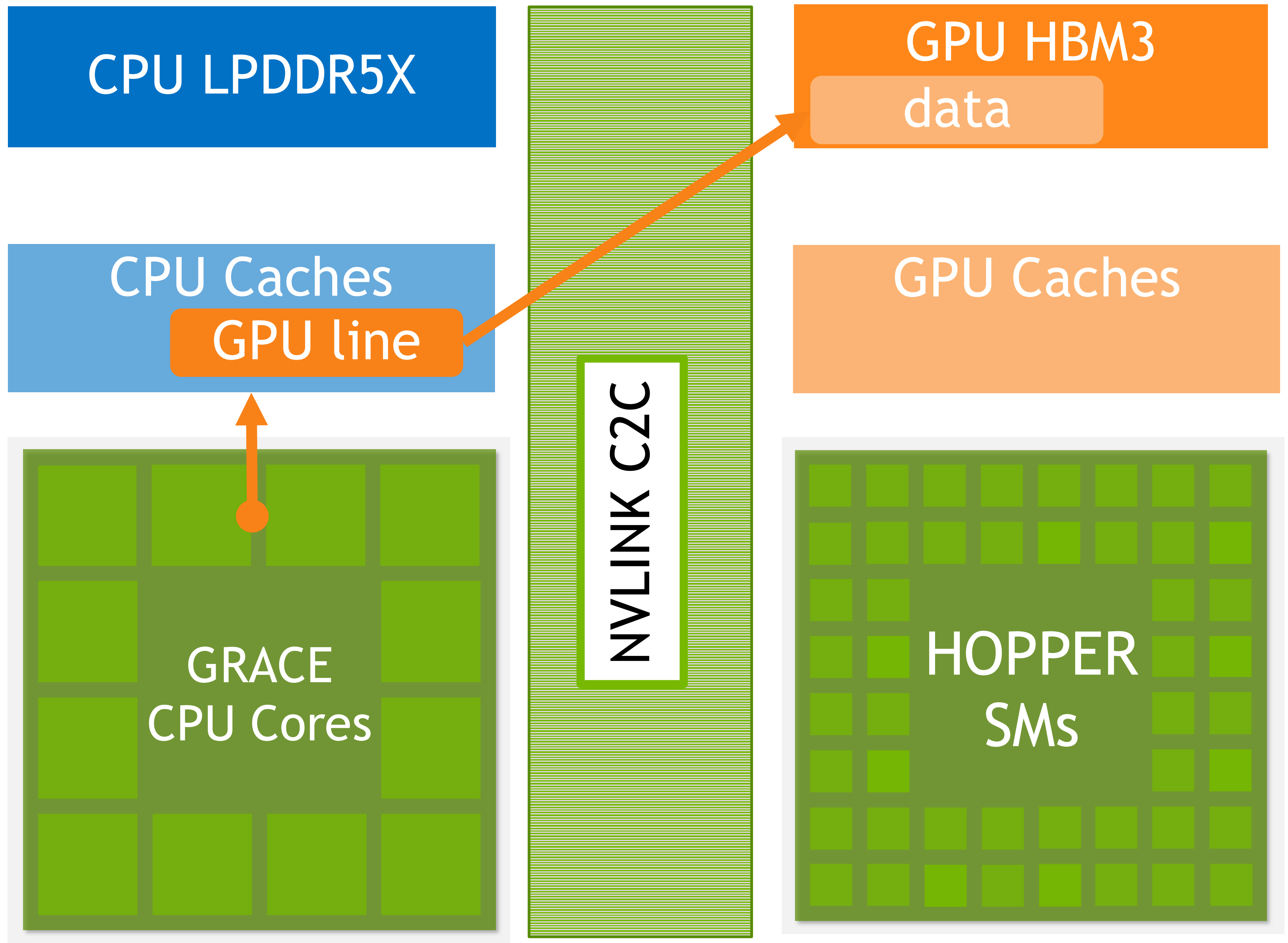
Coherently GPU Accelerated

- Exploit GPU / CPU coherency
- Use all available system features
  - not necessarily clean distinction between phases



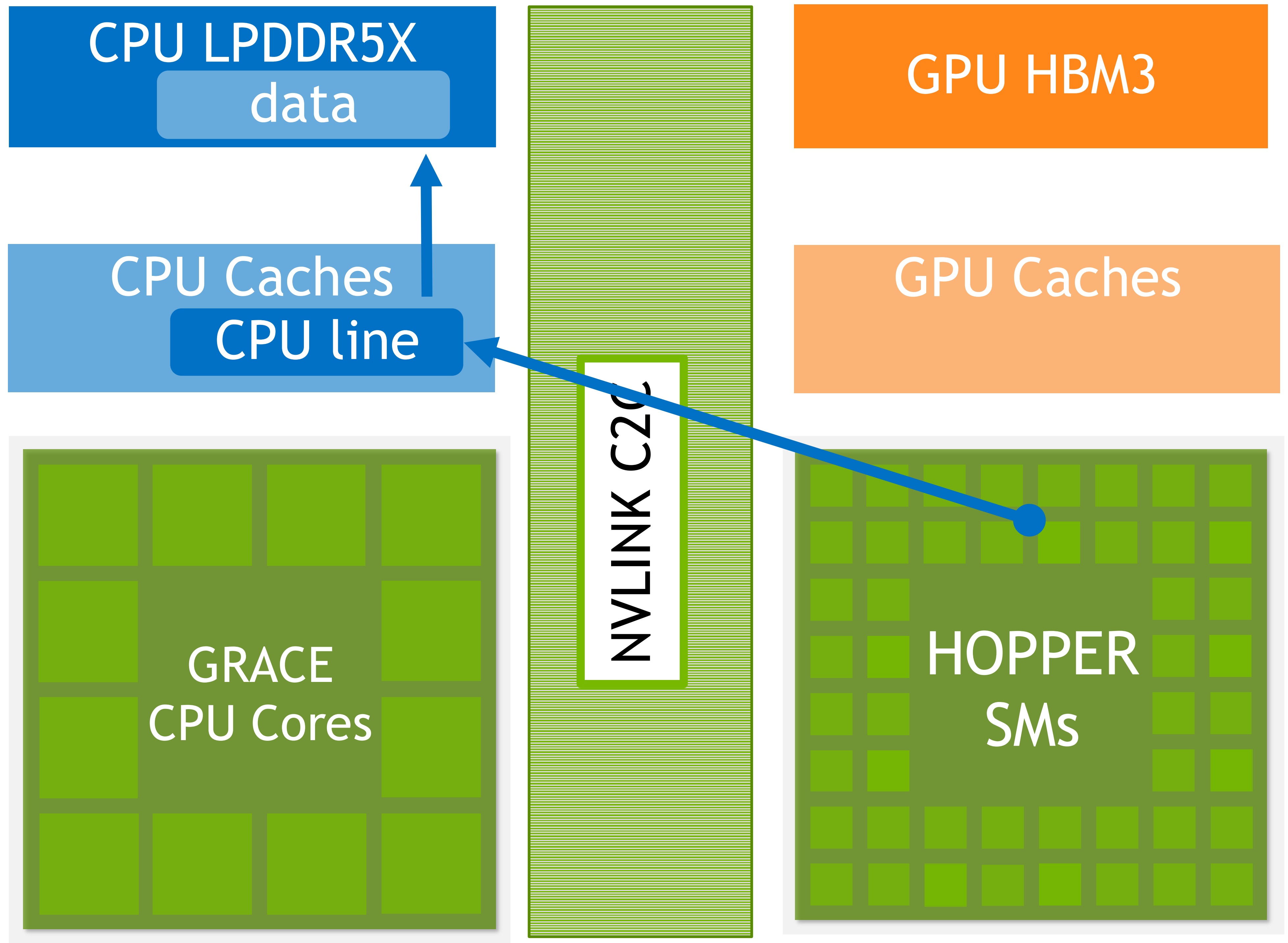
Global Access to All Data

Cache-coherent access via NVLink C2C from either processor to either physical memory



Grace directly reading Hopper's memory

CPU fetches GPU data into CPU L3 cache
Cache remains coherent with GPU memory
Changes to GPU memory evict cache line

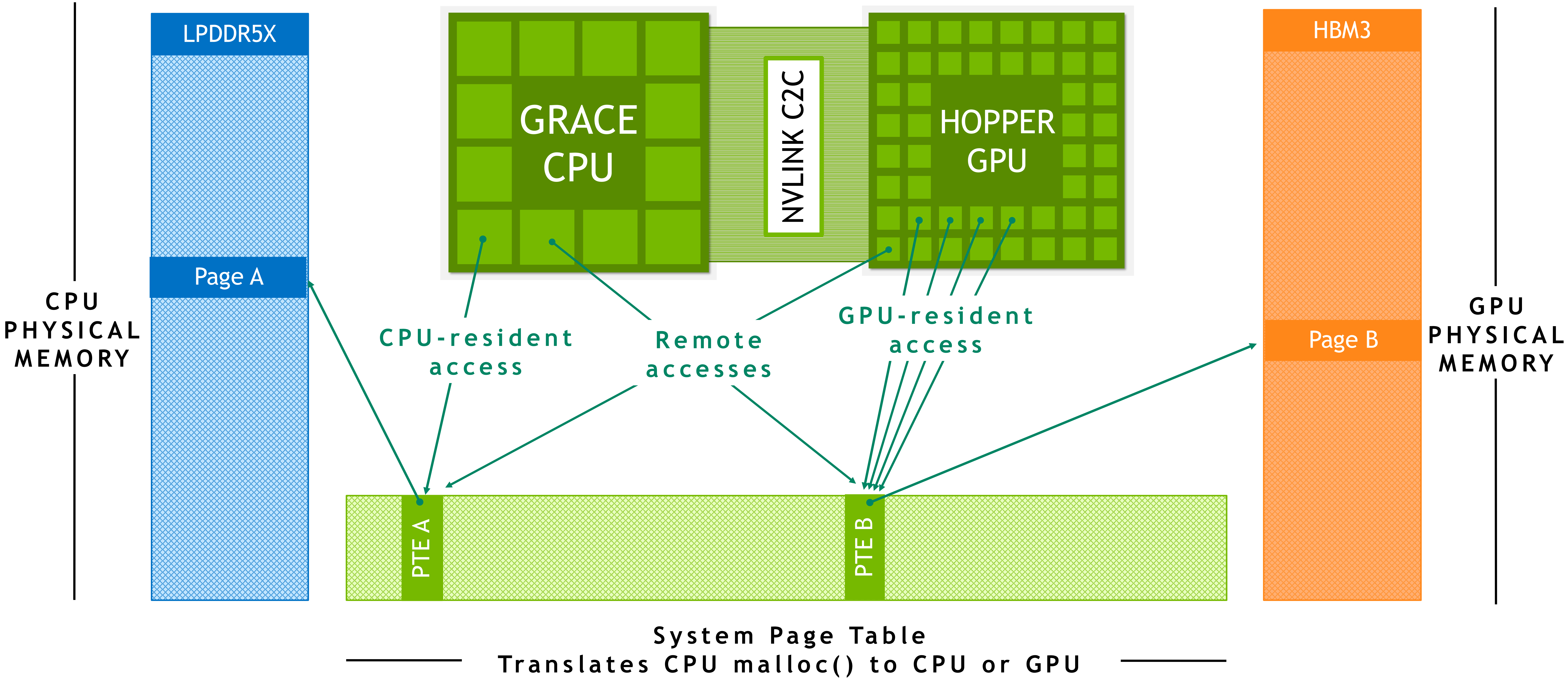


Hopper directly reading Grace's memory

GPU loads CPU data via CPU L3 cache
CPU and GPU can both hit on cached data
Changes to CPU memory update cache line

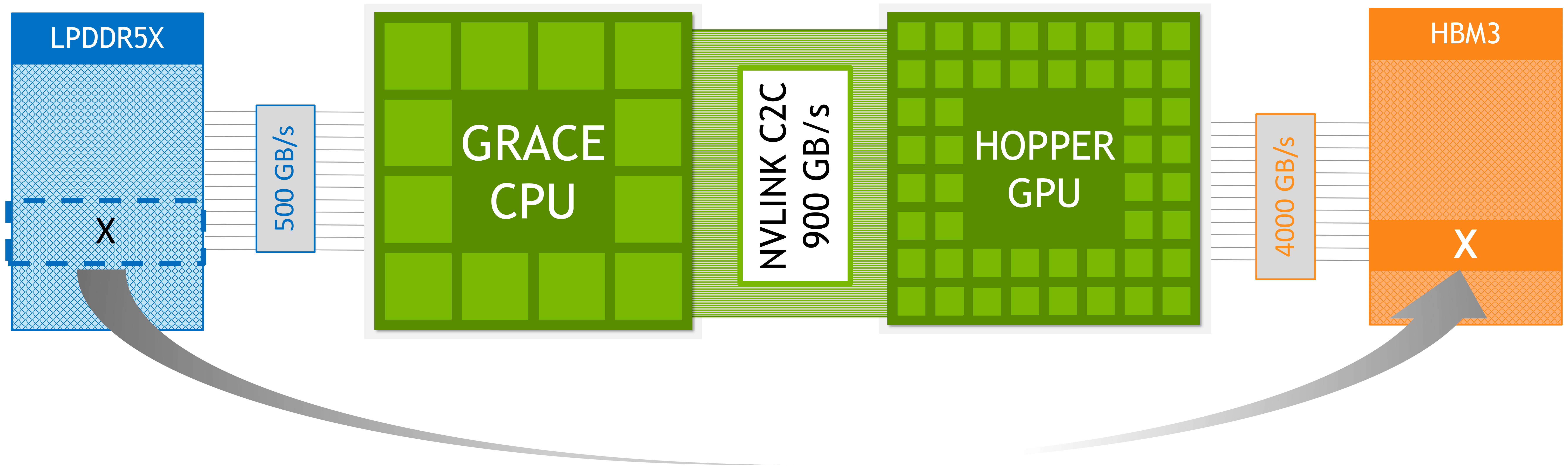
Grace Hopper

Address Translation Service (ATS) enables full access to all CPU & GPU allocations
Migrations are not required: Fewer Migrations



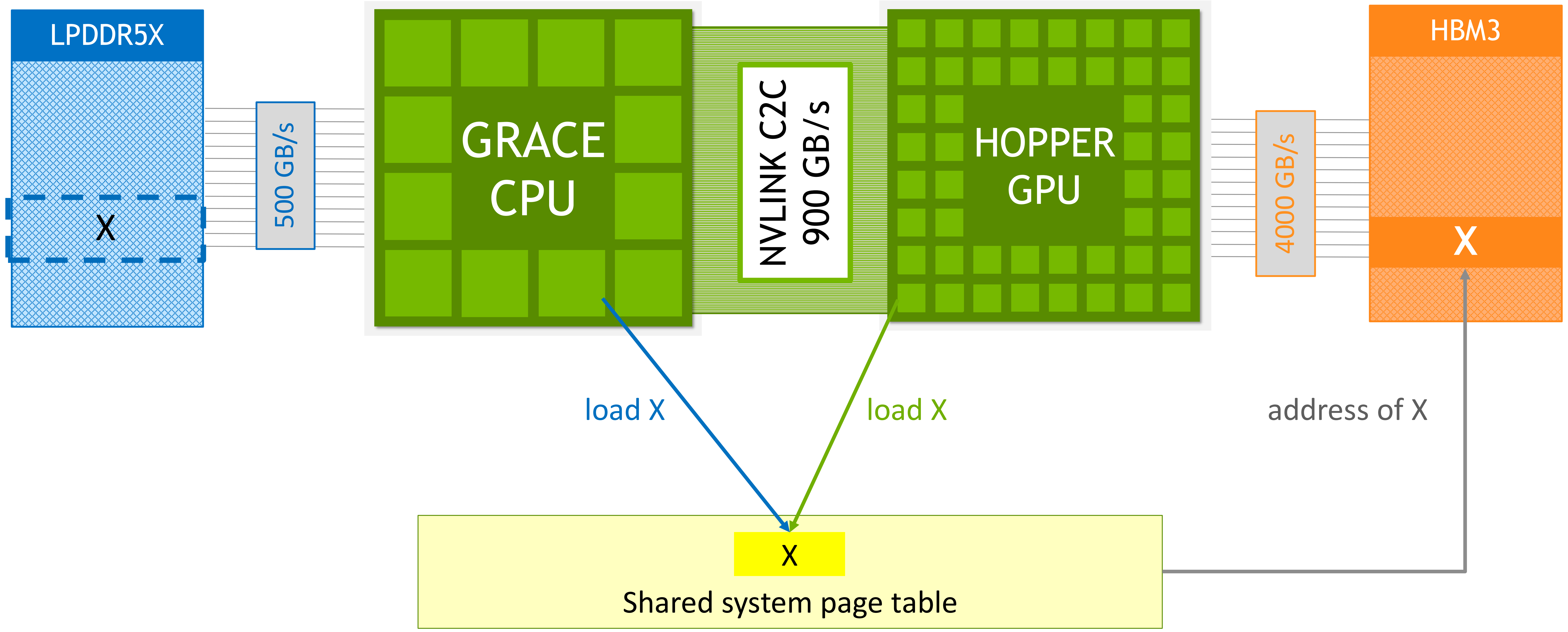
ATS creates a single page table for the whole system
NVLink C2C allows access to all physical memory without migration

High Bandwidth Memory Access & Automatic Data Migration



The system can automatically migrate
both managed and CPU-allocated memory
in order to optimize access speed

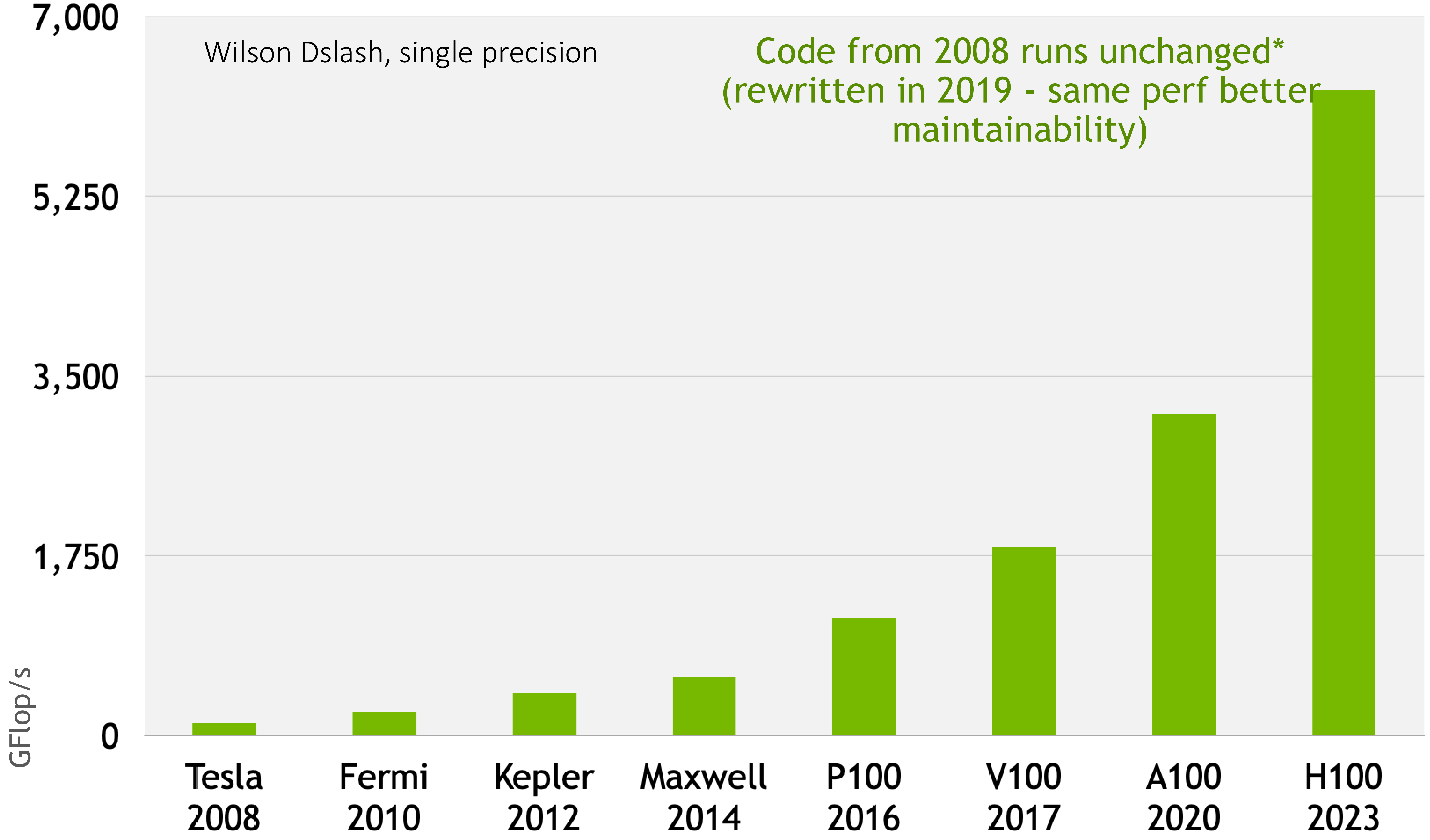
High Bandwidth Memory Access & Automatic Data Migration



ATS shared page table means that both CPU and GPU automatically access X in its new location after migration

SINGLE GPU PERFORMANCE

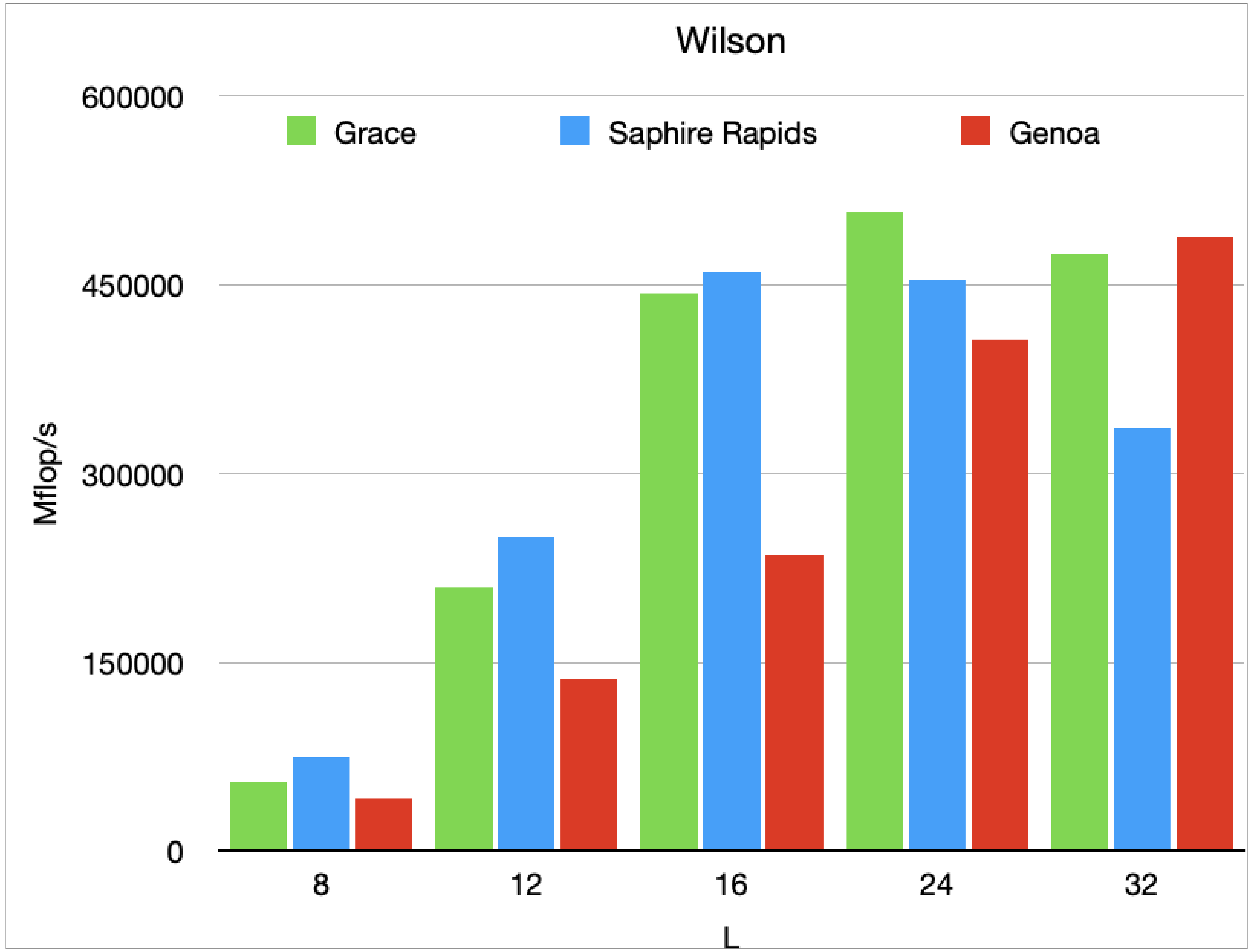
QUDA Wilson Dslash Kernel

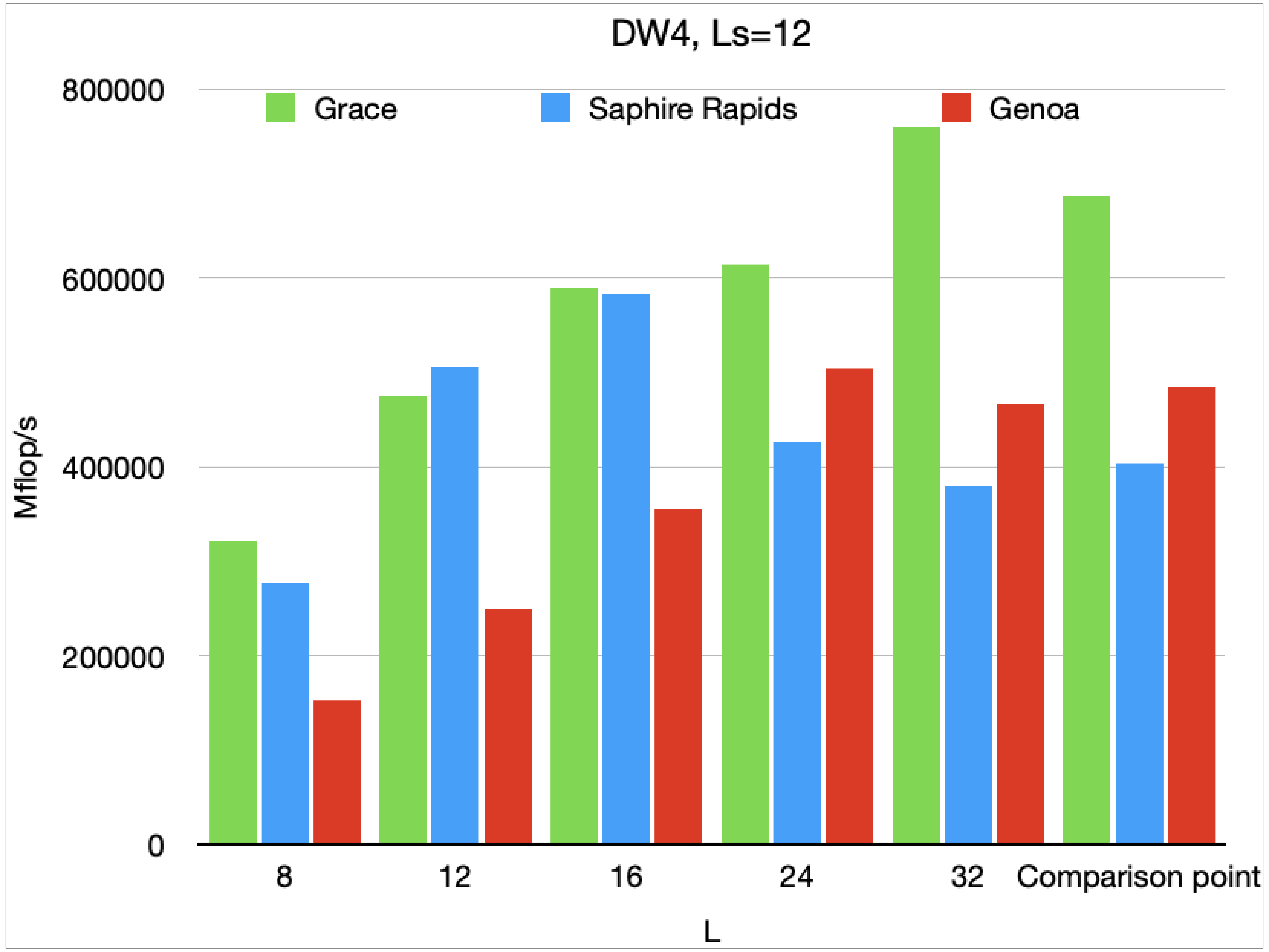


Code from 2008 runs unchanged*
(rewritten in 2019 - same perf better maintainability)

Grid CPU performance

Dirac Operator (Benchmark_ITT)



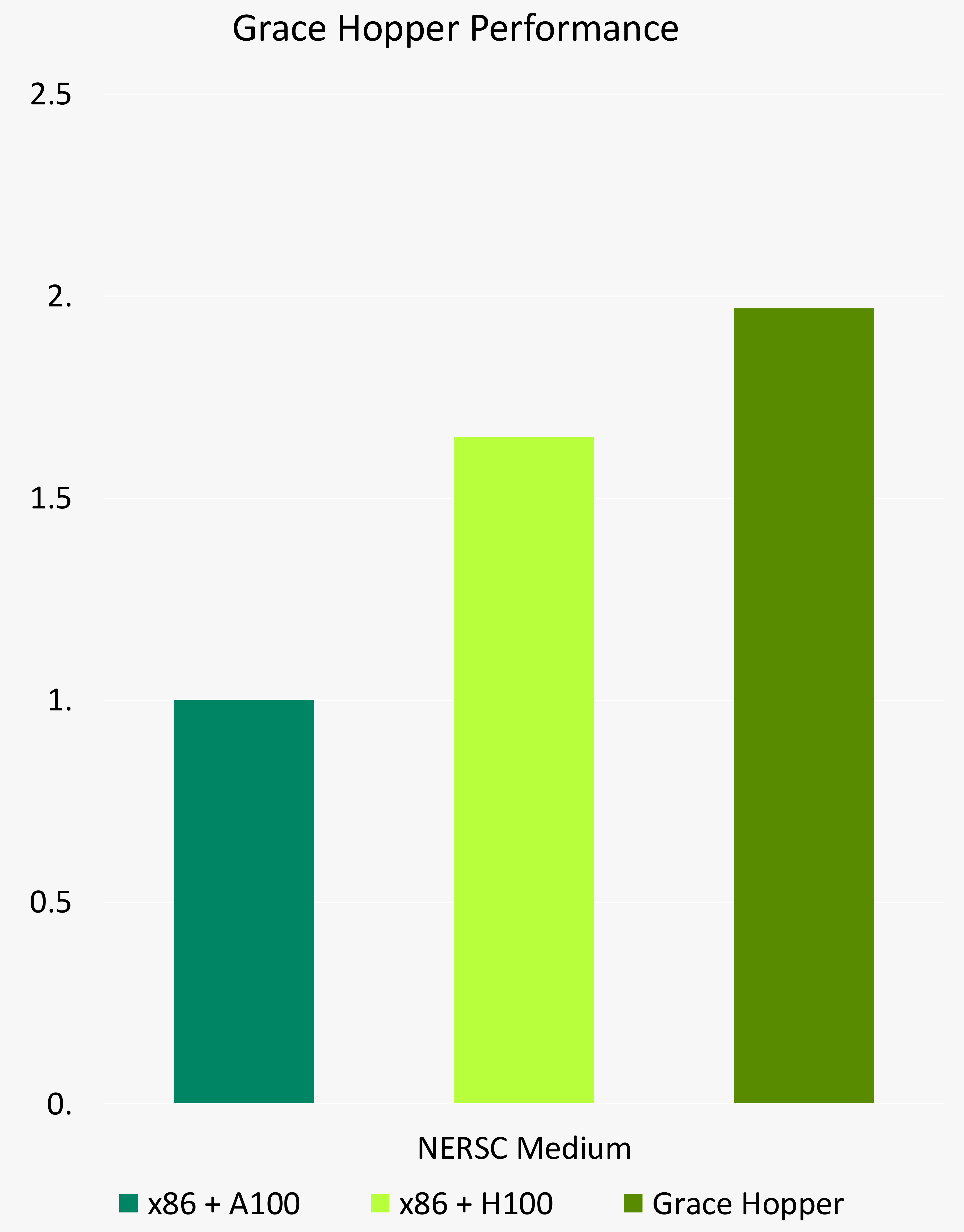


NVIDIA Grace Superchip vs x86 (AMD Epyc 9654 and Intel Xeon 8480+).
Grid development version as of July 2023 with GCC 12, results measured in July 2023

MILC RHMD Benchmark

Fully accelerated using QUDA

- GPU offload acceleration through QUDA with partial GPU data residency
- QUDA accelerated solvers
  - Mixed-precision multishift inverter
  - Gauge force
  - Fermion force
- **NERSC Medium benchmark**
- Performance on Grace-Hopper ensures 2x scaling over x86 +A100
  - C2C drastically reduces data-transfer time
  - Grace CPU memory bandwidth accelerates remaining CPU parts
  - Both combined restore scaling between generations

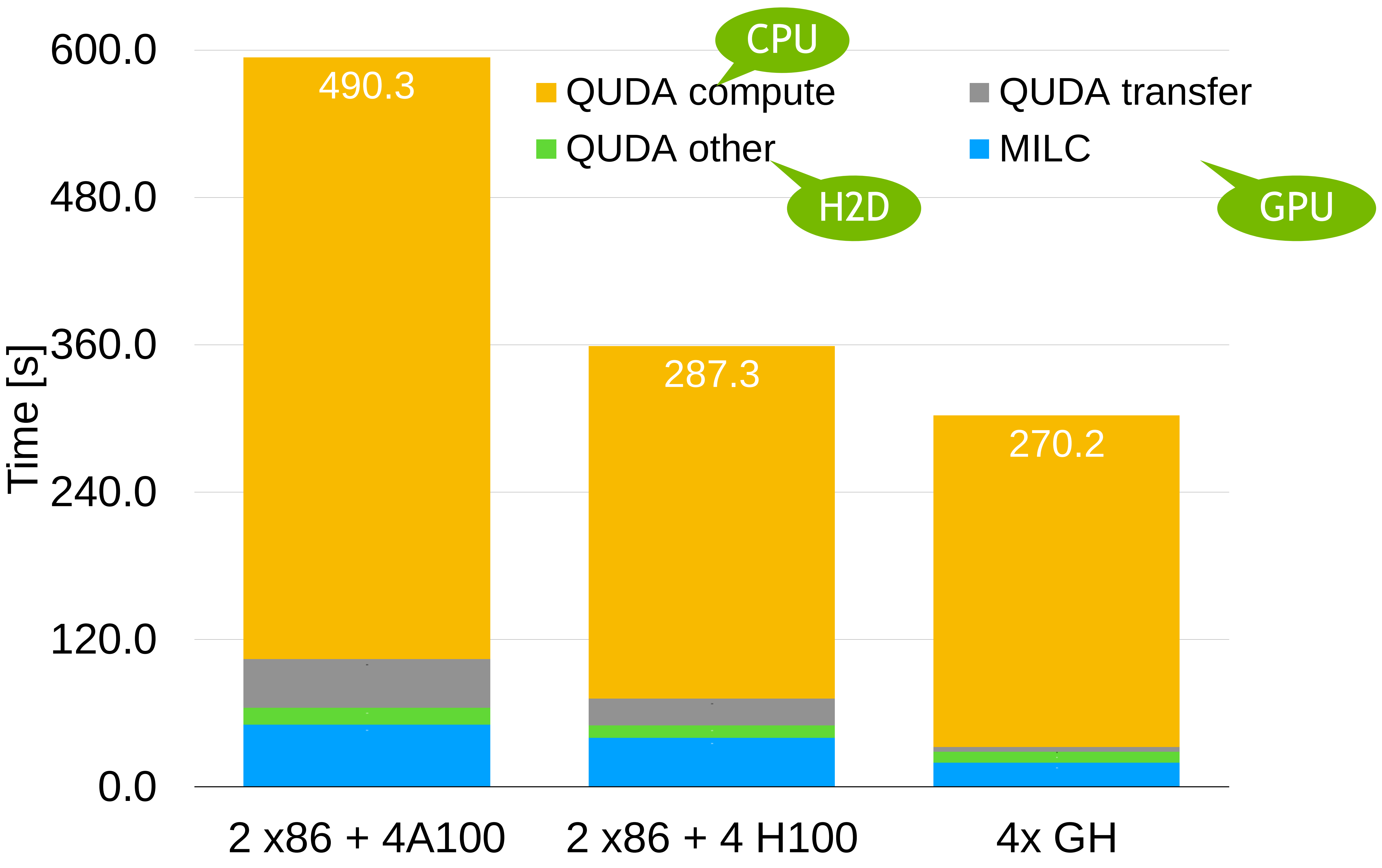


1 node with 4 GPUs

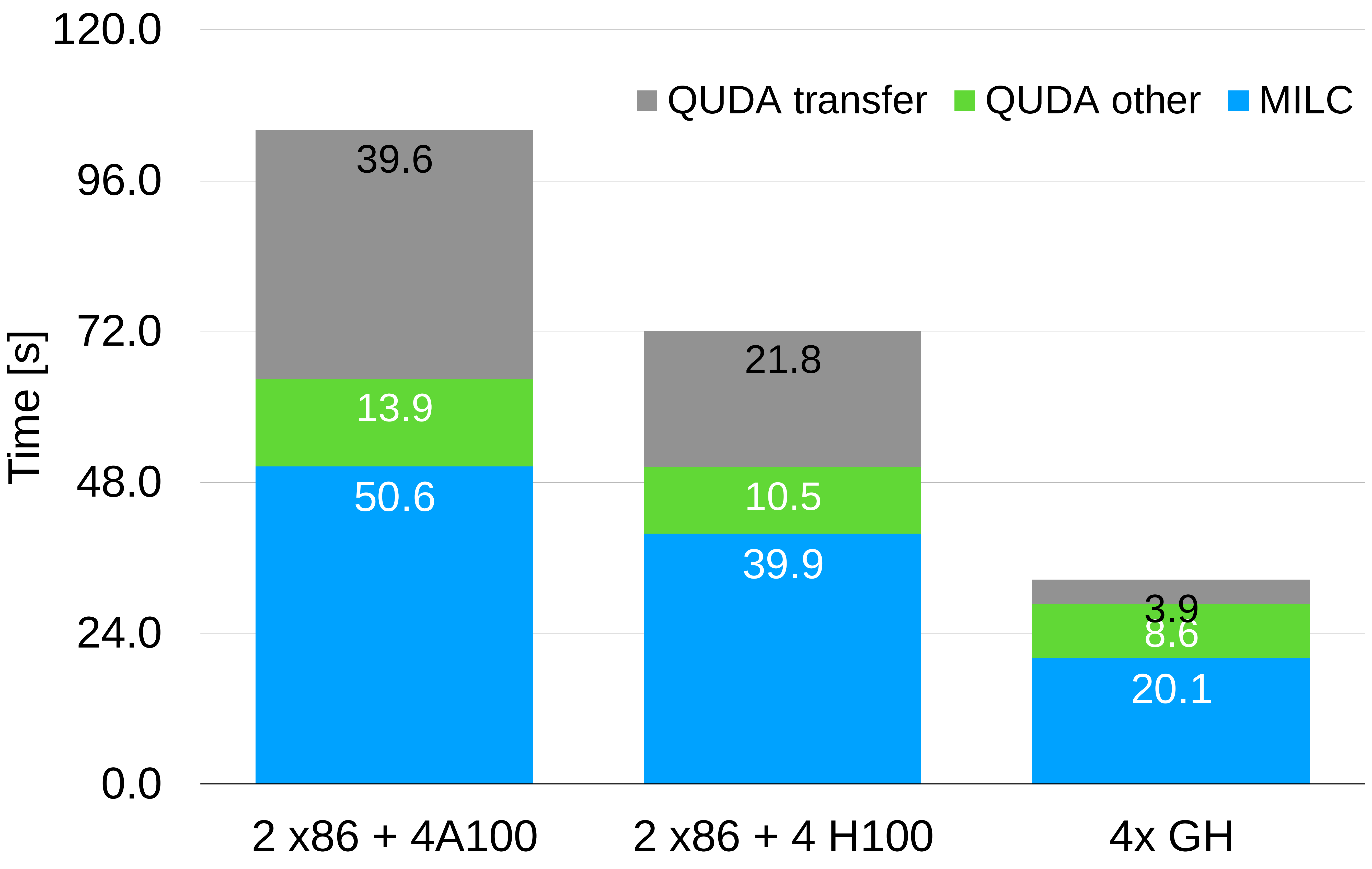
A100 runs were done using AMD EPYC (Rome) CPUs.
H100 runs were done using Intel Xeon (SPR) CPUs.

MILC

Breakdown



Full breakdown

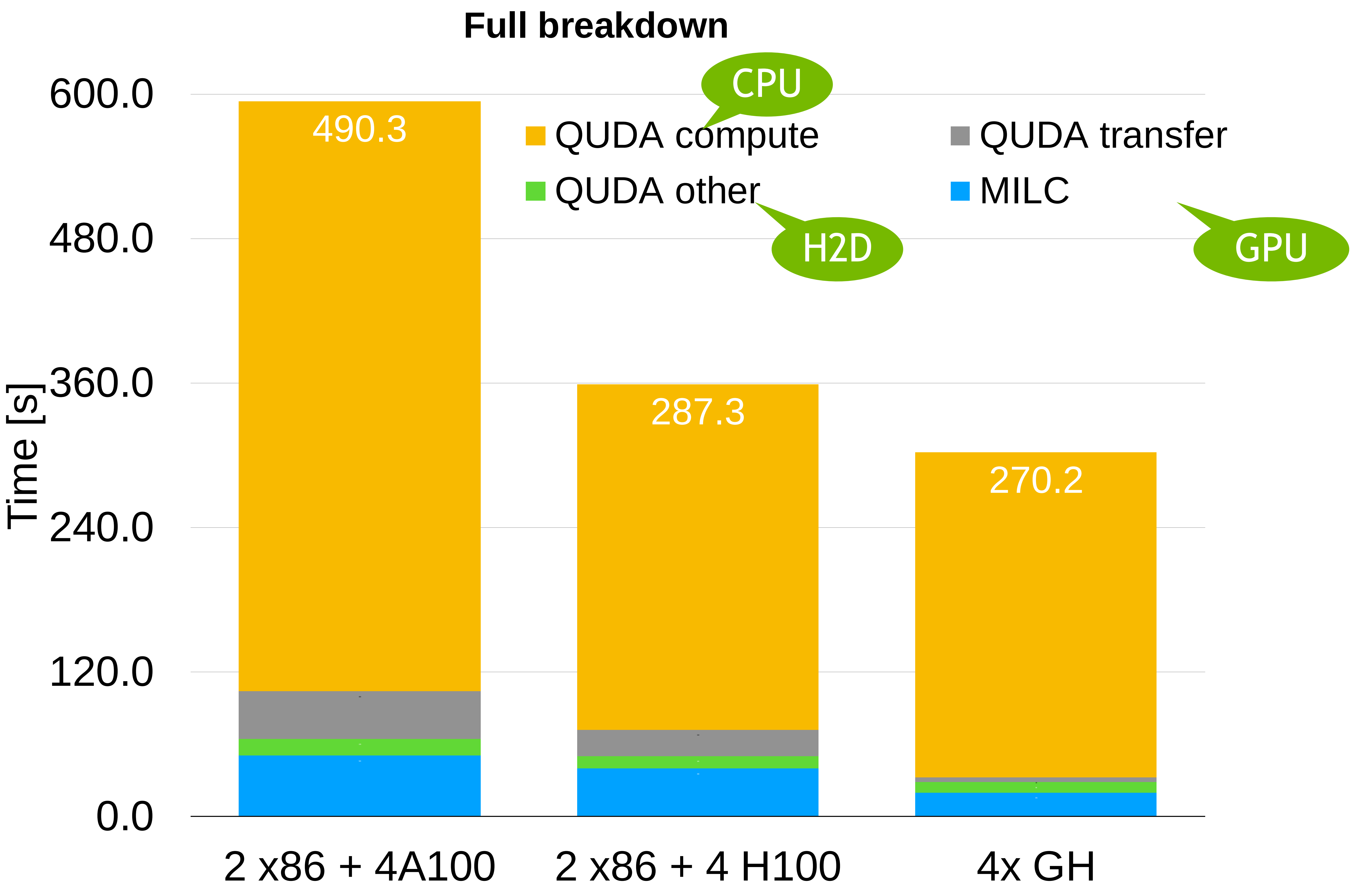


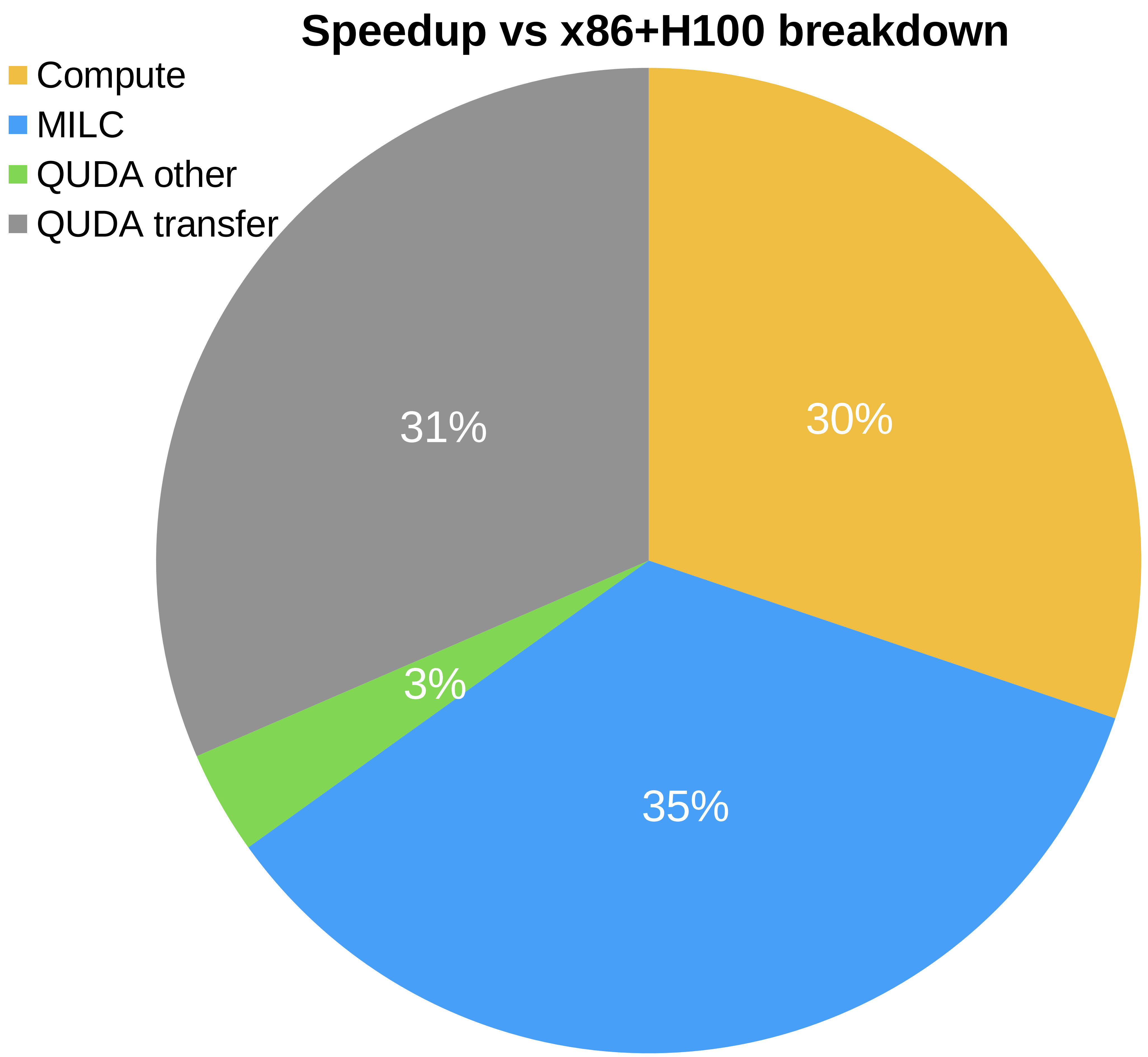
Non GPU breakdown

A100 runs were done using AMD EPYC (Rome) CPUs.
H100 runs were done using Intel Xeon (SPR) CPUs.

MILC

Breakdown



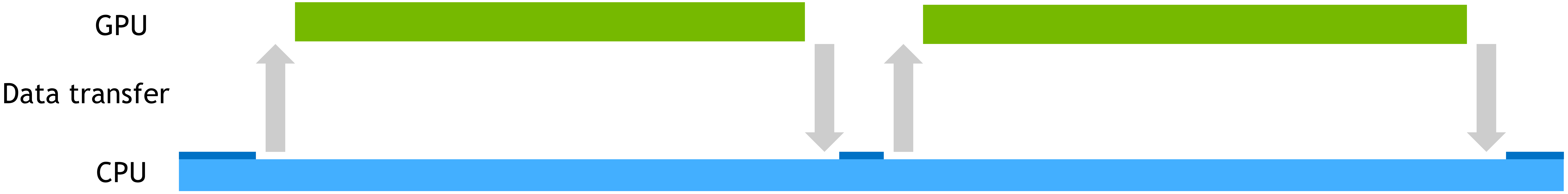


A100 runs were done using AMD EPYC (Rome) CPUs.
H100 runs were done using Intel Xeon (SPR) CPUs.

Backfill free CPU resources

Run highly demanding phase on GPU and overlap another with another phase on the CPU

- Accelerated jobs mostly uses the GPU and only fraction of CPU

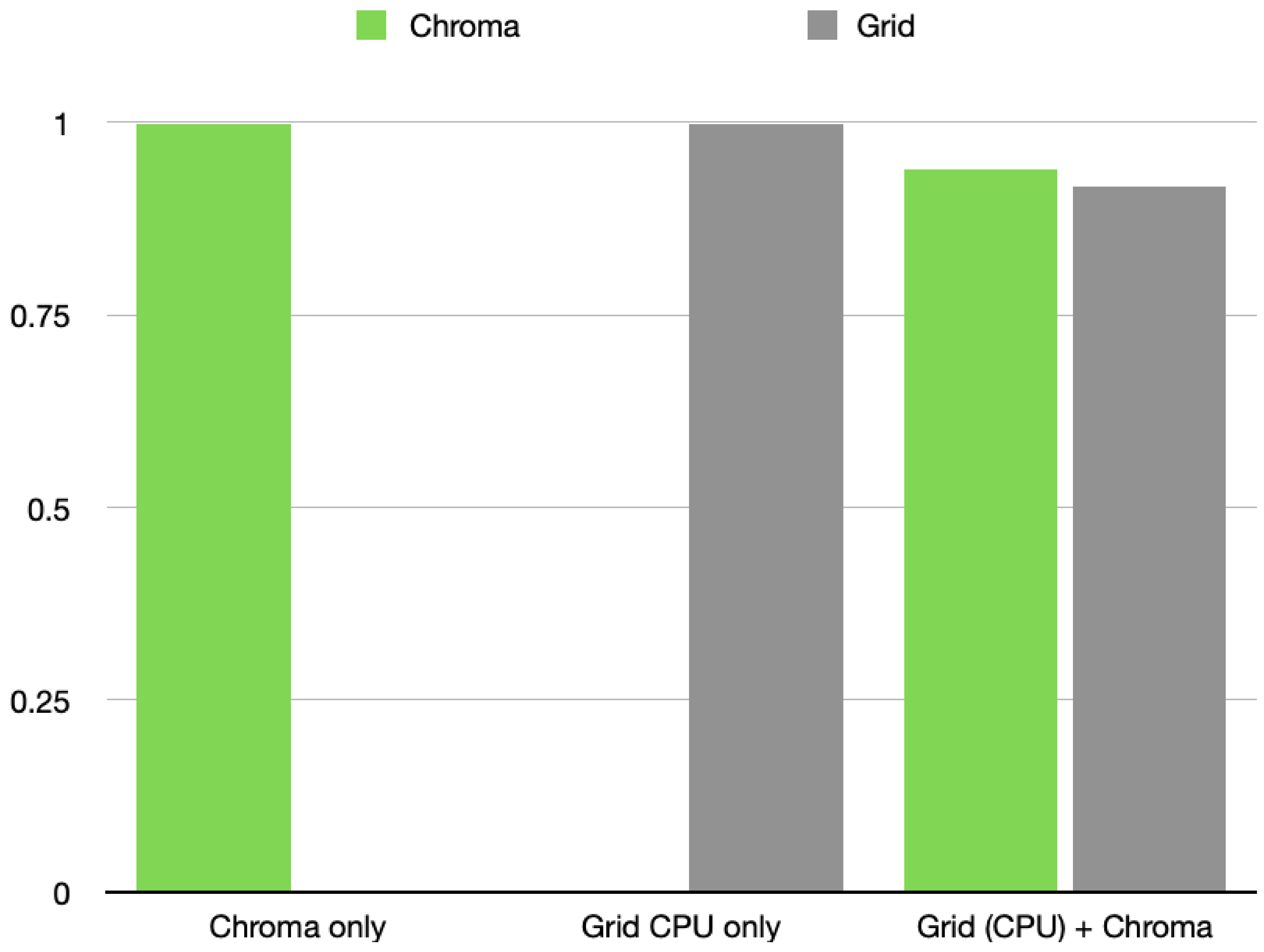


- Backfill idle CPU resources

Chroma HMC + Grid Dslash

Co-scheduled on a 4 GH node

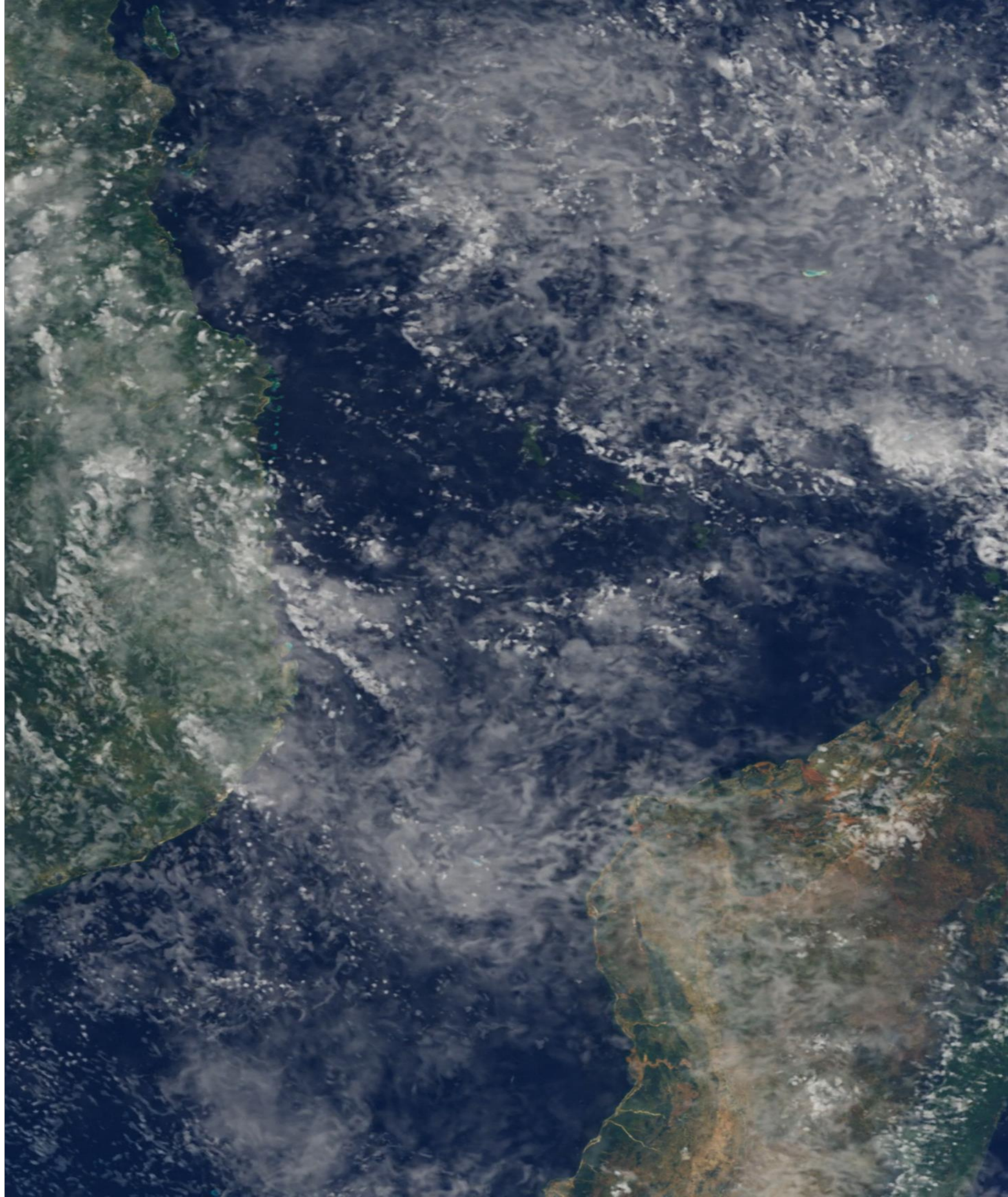
- Chroma HMC with QUDA + QDPJIT
  - Fully device resident
  - Requires just a single core to drive GPU
- Grid Dirac Wilson benchmark Proxy to simulate running a CPU heavy workload
  - Analysis job, …?
  - Can use 64 cores per Grace CPU
- Combine both to fully exploit the node
  - Performance impact ~ 5%
- More throughput
- Increase energy efficiency
  - Additional CPU job just consumes increase in CPU power
  - System socket power in consumed anyway



Examples from other domains

ICON Coupled Ocean

- ICON is a unified next-generation global numerical weather prediction and climate modelling framework
- Developed by DWD (German weather prediction center), MPI-M (German Max Planck climate research institute) and MeteoSwiss with help from CSCS
- Currently used for operational forecast at DWD, soon to be in production in Switzerland on GPUs. Used by many institutes for climate simulations
- Typical scales from 1 to 1000s GPUs
- Atmospheric simulation is fully GPU-ported with OpenACC. Ocean part is not fully ported yet and can only be run on the CPU
- Coupled atmosphere-ocean simulations are very important for understanding long-term climate change and multiple institutions are currently working on such setups



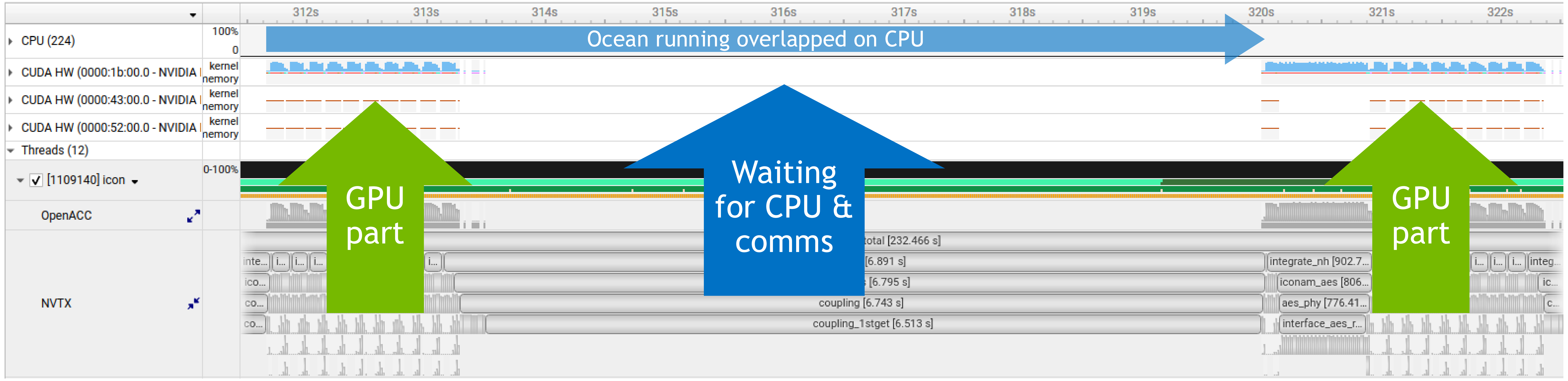
ICON Coupled Ocean

Profile

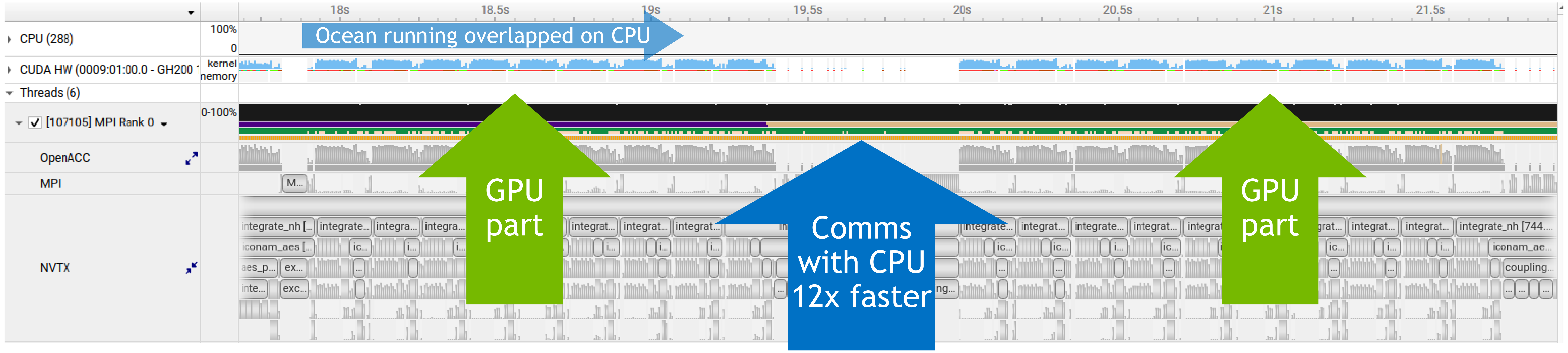
- Full globe coupled simulation at 10 km atmosphere resolution and 5 km ocean resolution. 90 vertical atmosphere layers, 72 vertical ocean layers. Atmosphere time-step is 90s, ocean time-step is 5 min and coupling time-step is 15 min. Atmosphere and ocean run in different ranks within the same MPI job. 64 GPUs and 512 (Eos) or 3008 (Alps) CPU ranks

Profiling atmosphere GPU process

Eos



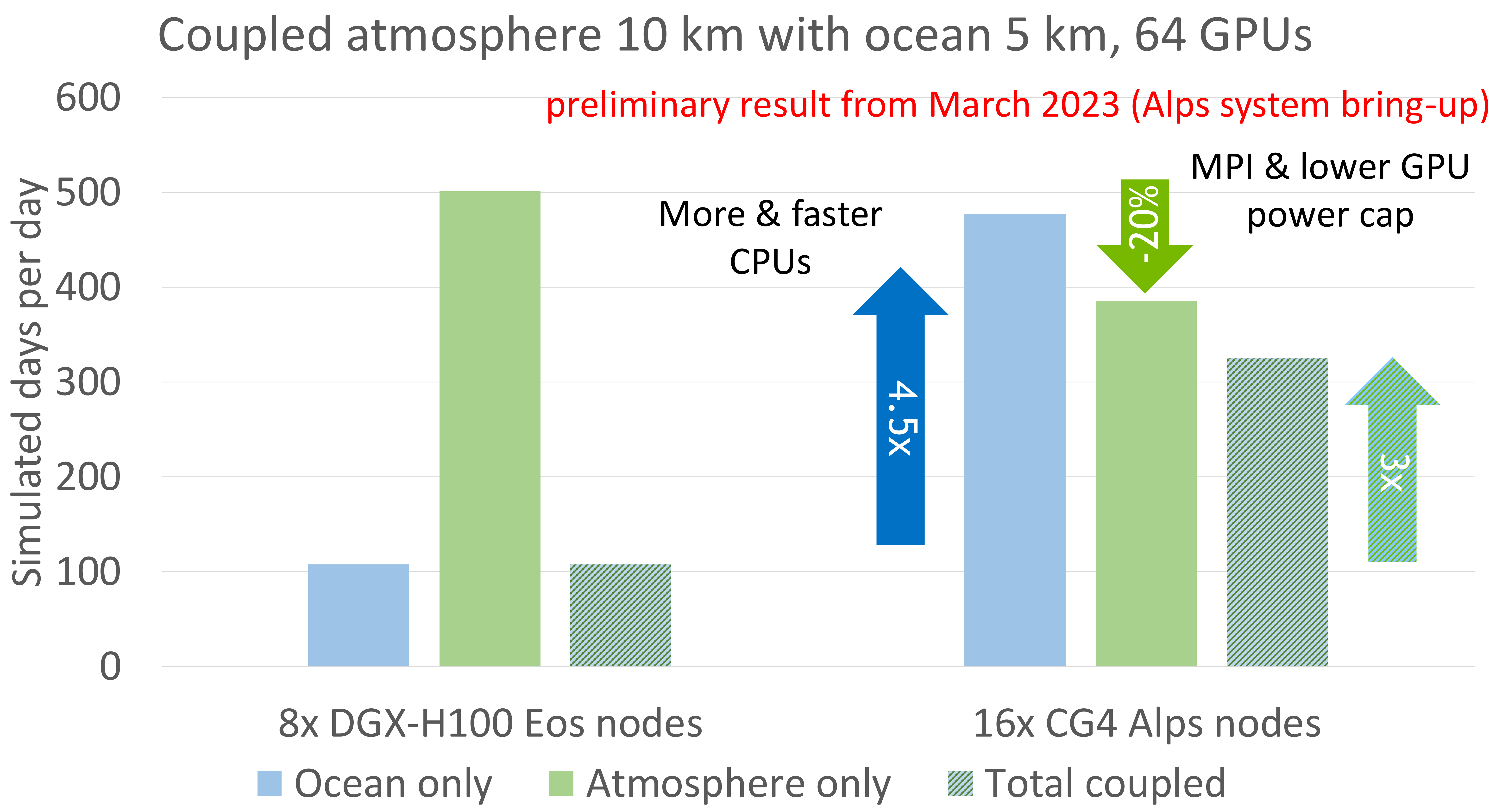
Alps



ICON Coupled Ocean

Grace-Hopper: 3x speedup

- On EOS:
  - Performance limited by Ocean running on the CPU
- On Alps:
  - Unleash full performance of Hopper GPUs
  - Grace is powerful enough to run the ocean in the background
  - Alps network is still in bring-up phase, which introduces some atmosphere-only and coupling overhead
  - 3x end-to-end performance



NEMO Ocean Model

A partially accelerated case utilizing unified memory on Grace-Hopper

The "**N**ucleus for **E**uropean **M**odelling of the **O**cean" (**NEMO**) is a state-of-the-art modelling framework, used for research activities and forecasting services in ocean and climate sciences.

- **Setup (NEMO v4.2.0)**
  - **GYRE_PISCES** benchmark
    - Scaling factor for grid resolution: **nn_GYRE = 25**
      - ~ORCA ½ grid
      - ~80 GB RAM, fits on single GPU
  - **MPI-only**, single core to every MPI process for CPU runs

- **Incremental porting** on Grace-Hopper **(480GB)** using unified memory and access-counter based migrations
  - Memory management left to runtime – **system-allocated memory with automatic migrations**
    - compile with `-gpu=unified,nomanaged`
  - Simply offloading loops to GPU using **OpenACC**, in 3 steps:
    - **Horizontal (lateral) diffusion,**
    - **Advection,**
    - **Vertical diffusion and time-filtering,**

    for both “active” (**TRA**) and “passive” (**TRC**) tracer transport

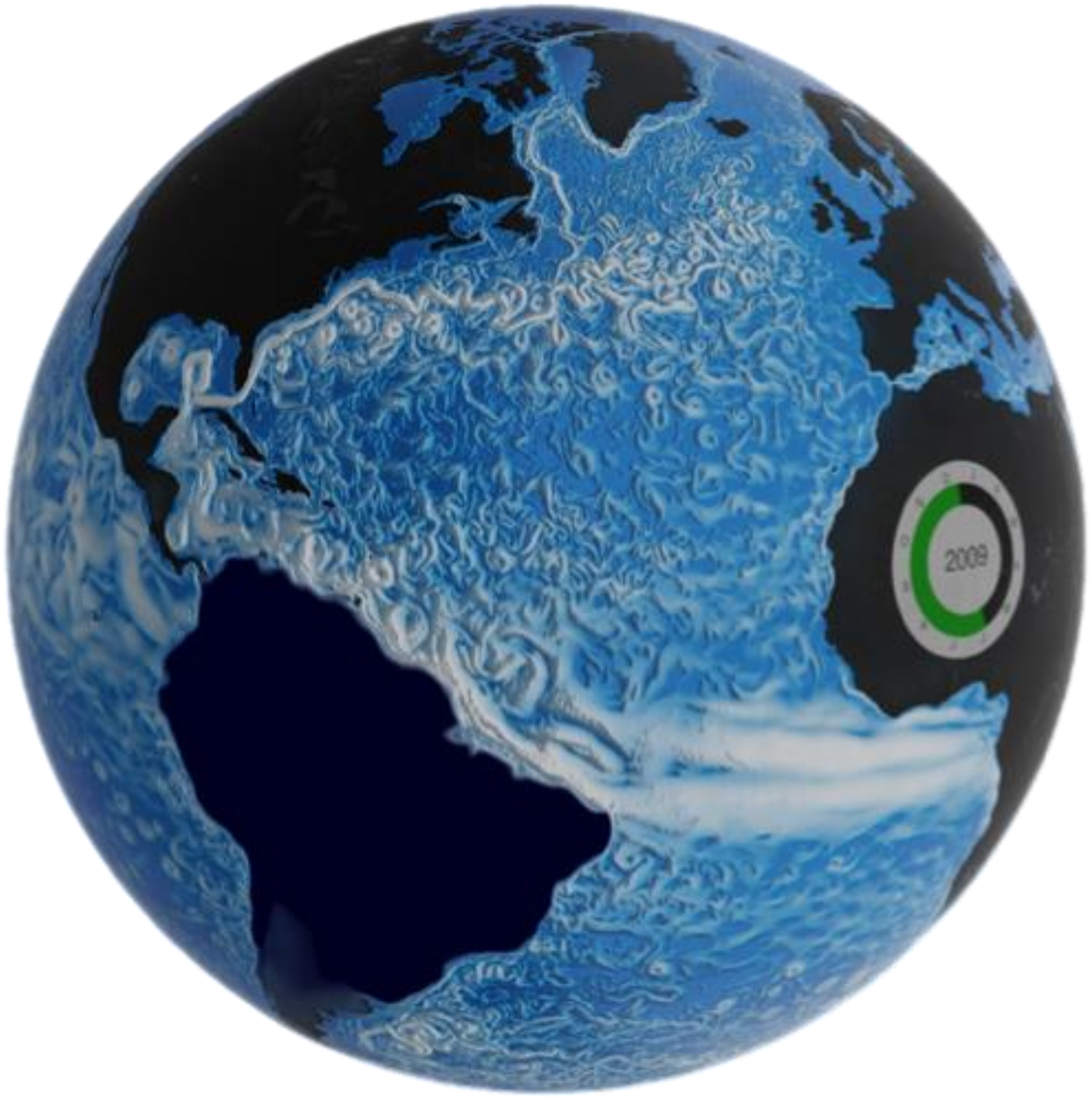
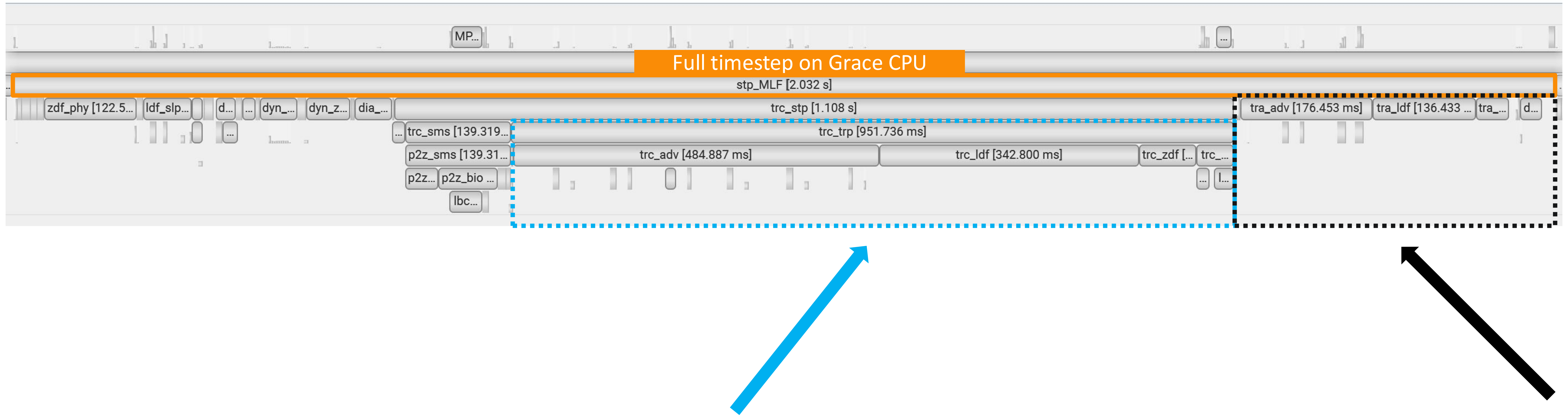


Image source:
NEMO User Guide — NEMO release-4.2.2 documentation (nemo-ocean.io)

Porting NEMO to Grace-Hopper using Unified Memory

Incremental porting, zooming in to a single timestep …



"Passive" tracer transport (TRC)

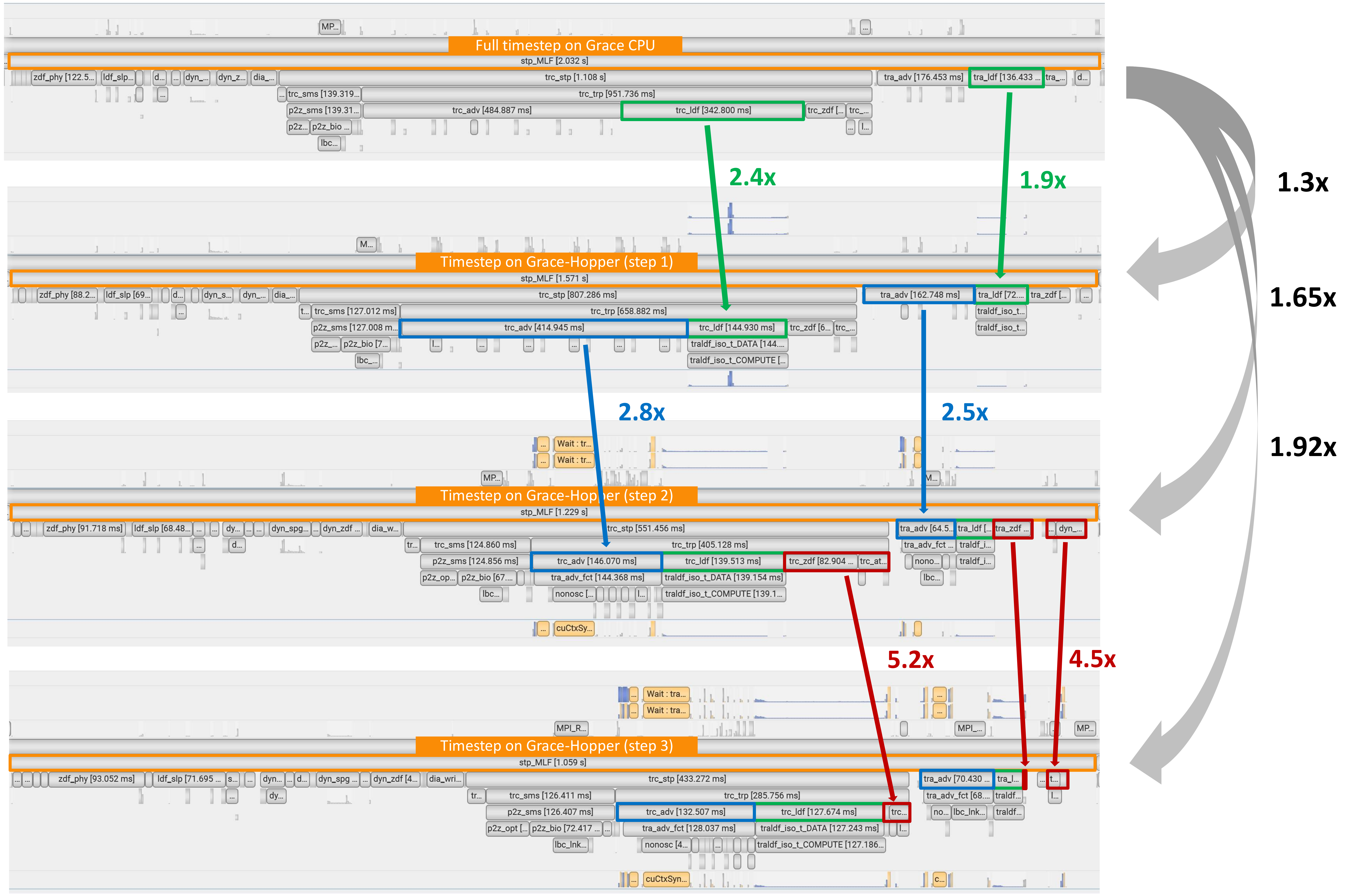
"Active" tracer transport (TRA)

Porting NEMO to Grace-Hopper using Unified Memory

Incremental porting, zooming in to a single timestep …

Ported to GPU:

- **Horizontal diffusion**
- **Advection**
- **Vertical diffusion and time-filtering**



We run multiple (i.e. 40) MPI processes on **CPU and GPU** using **MPS**, and use **“migratable”** system allocated memory

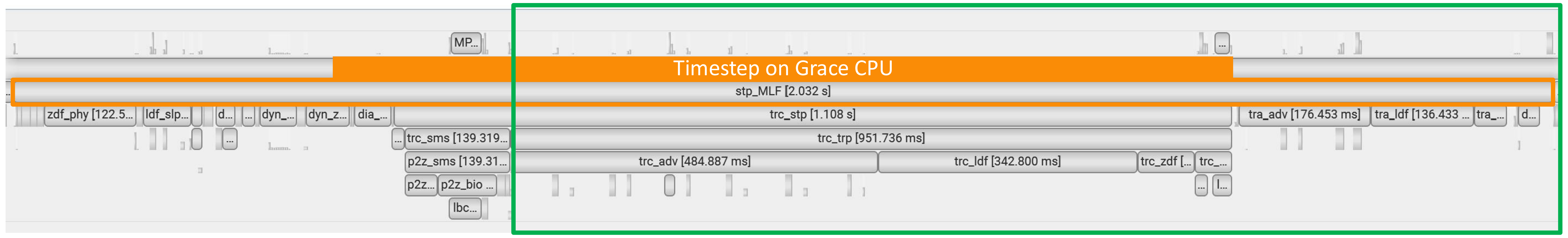
Porting NEMO to Grace-Hopper using Unified Memory

A deeper look into the effect of access-counter based migrations on the partially accelerated port

CPU only run

0.656 ms

1.376 ms



1.46x

1.33x

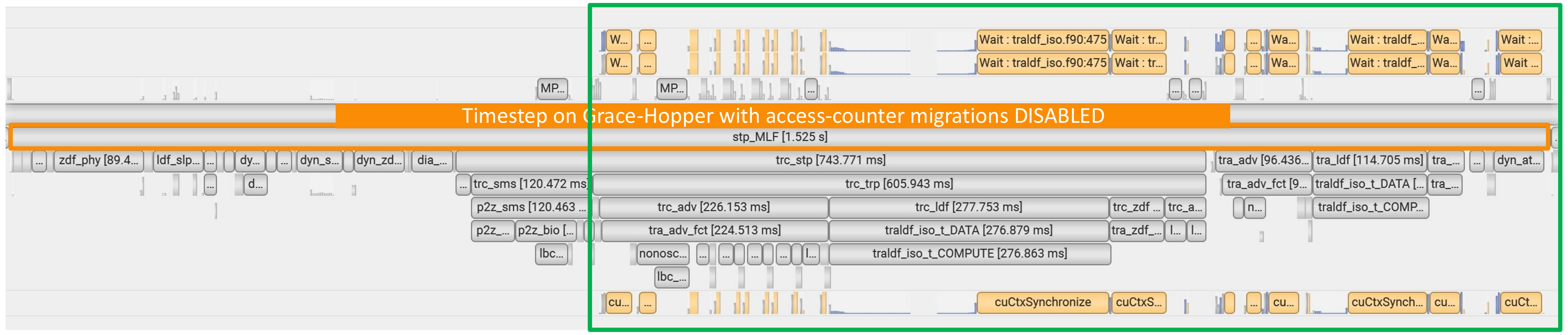
GPU kernels pull data first-touched by CPU directly from **CPU memory**

Tracer transport on GPU with migrations disabled

(buffers with first-touch on CPU will never migrate to GPU memory)

0.580 ms

0.945 ms



2.02x

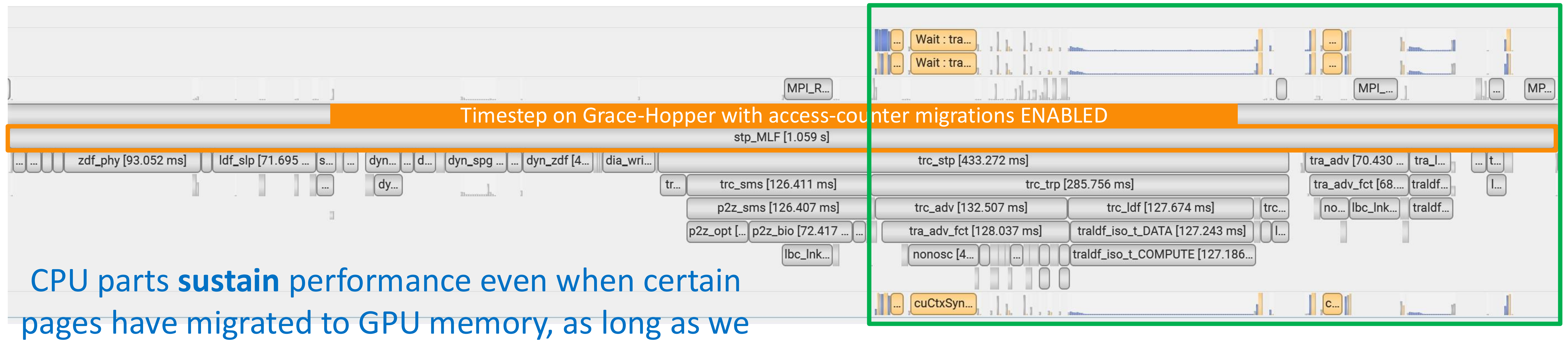
1.44x

Enabling automatic page migrations from CPU to GPU

(“hot” pages migrate to GPU)

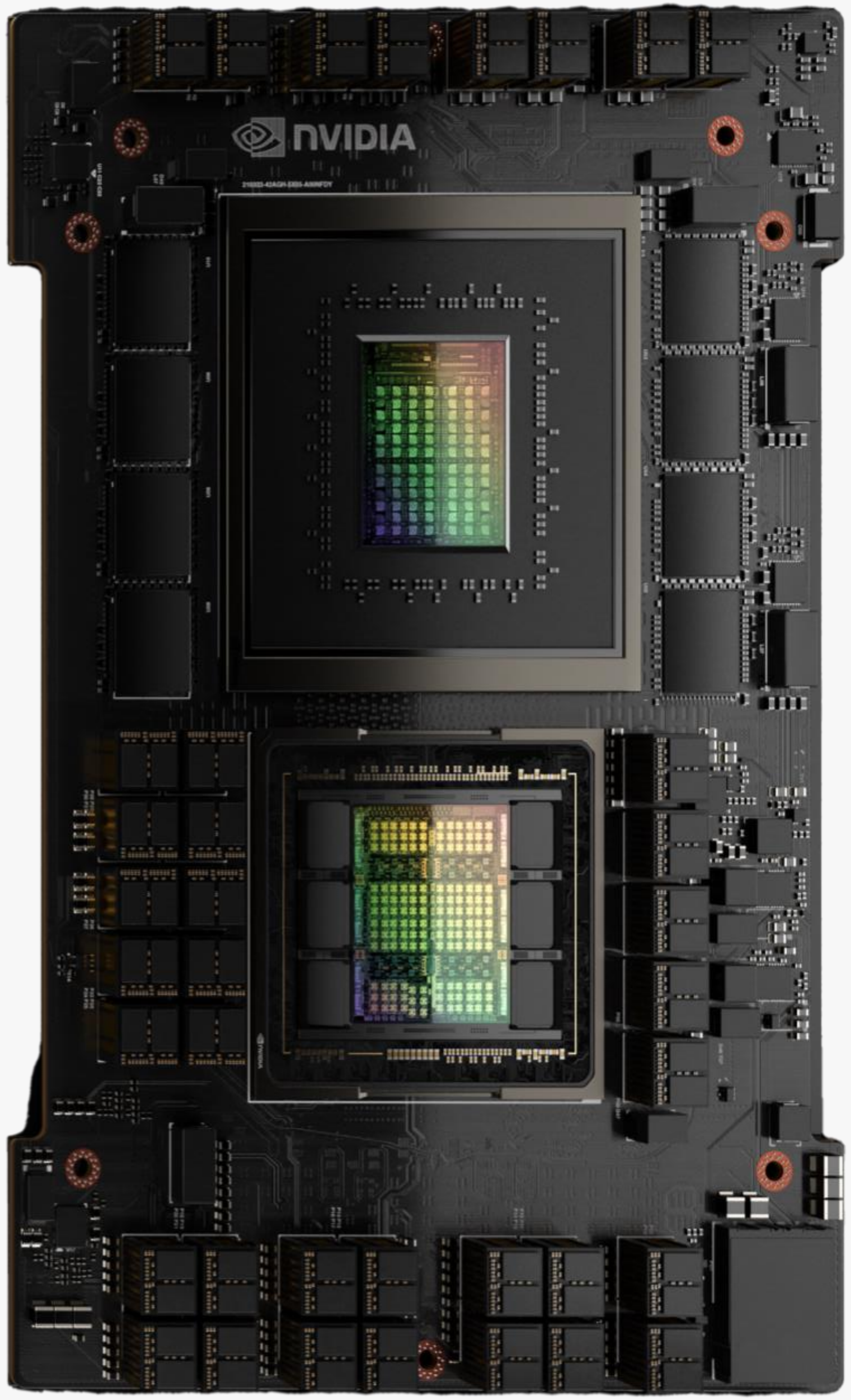
0.591 ms

0.468 ms



CPU parts **sustain** performance even when certain pages have migrated to GPU memory, as long as we use enough processes to saturate C2C BW

GPU kernels become faster as more and more pages migrate to GPU



Science with Grace Hopper

Better than ever

- Balanced platform with
  - Hopper GPU
  - Grace CPU
  - C2C Interconnect (High Bandwidth)
  - Unified Memory Space
- Easier to accelerate
  - Widening bottlenecks unleashes performance
  - Easier to program
- Multiple Installations coming online

NVIDIA®