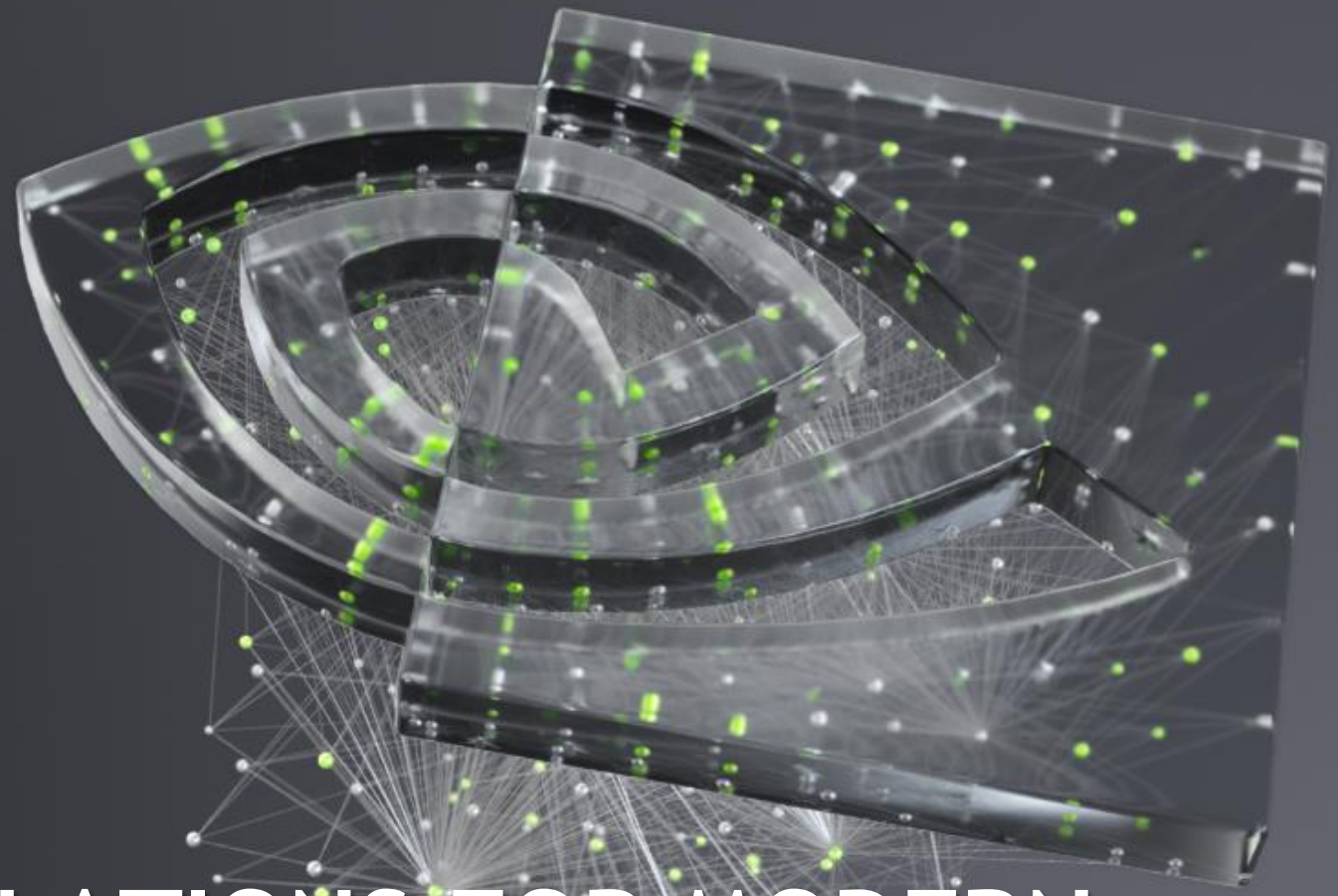


# DESIGNING MD SIMULATIONS FOR MODERN HETEROGENEOUS PARALLEL ARCHITECTURES

Alan Gray (NVIDIA) and Szilárd Páll (KTH)

PASC, 29th June 2022

# INTRODUCTION

Modern servers are **heterogeneous** with components including CPUs, GPUs and interconnects which should be used together as **asynchronously** as possible.

The **critical path** of the application should be dominated by compute kernels running efficiently on the fast GPUs (and spend minimal time waiting for tasks scheduled to the other components).

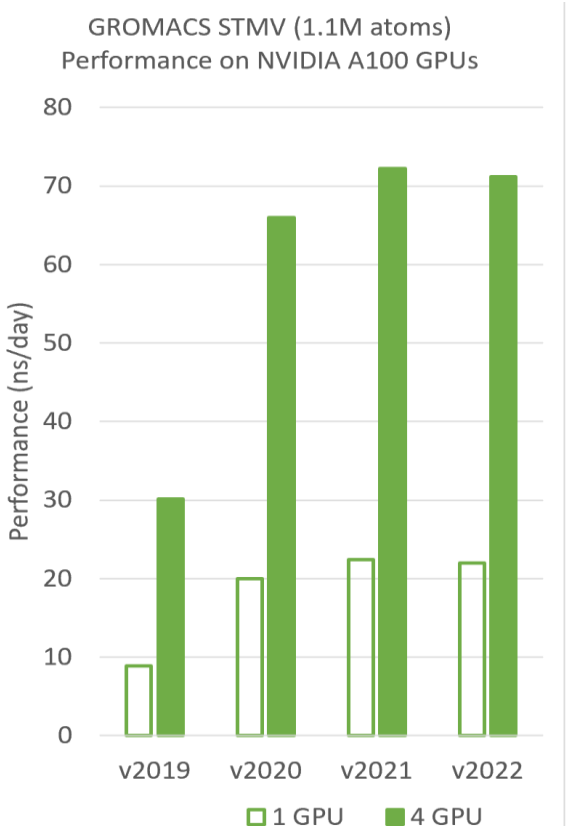
As GPUs continue to become faster, asynchronous design becomes increasingly vital.

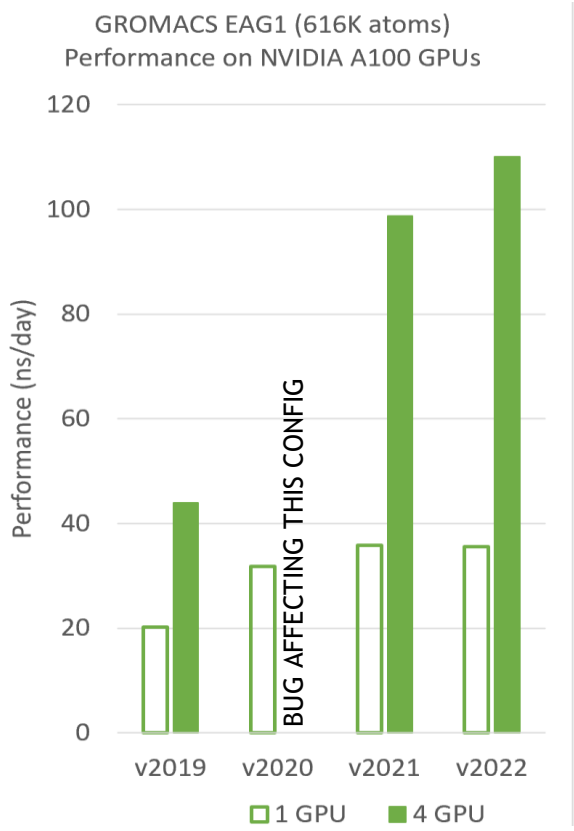
We will show how these basic principles have led to dramatic performance improvements in the GROMACS MD application (with concepts transferrable to other apps).

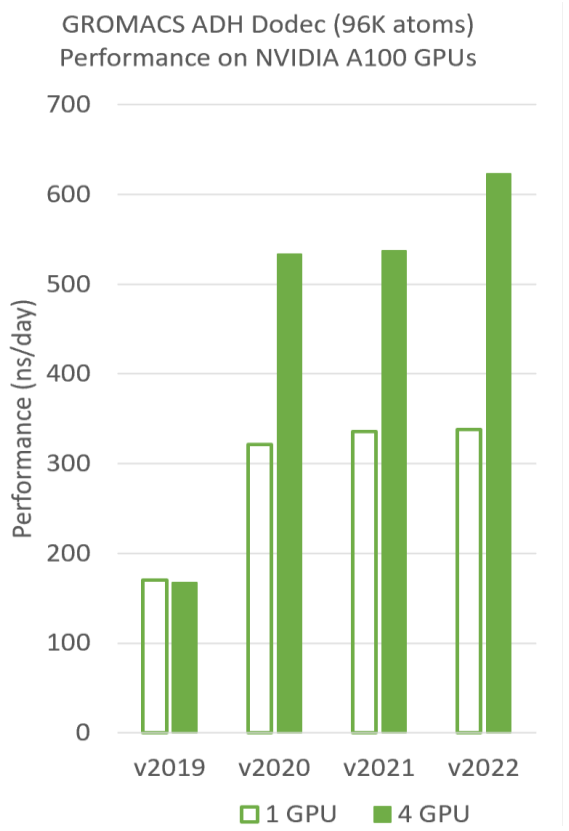
- Simulation package for biomolecular systems - one of the most highly used scientific software applications worldwide, and a key tool in understanding important biological processes.
- Evolves systems of particles using the Newtonian equations of motion through repeated updates. Forces between particles dictate their movement. Parallel implementation results highly complex schedule of (often microsecond-scale) tasks.
- Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS (https://aip.scitation.org/doi/full/10.1063/5.0018516)

Work done in collaboration between NVIDIA and core GROMACS developers.

# GROMACS IMPROVEMENTS OVER TIME



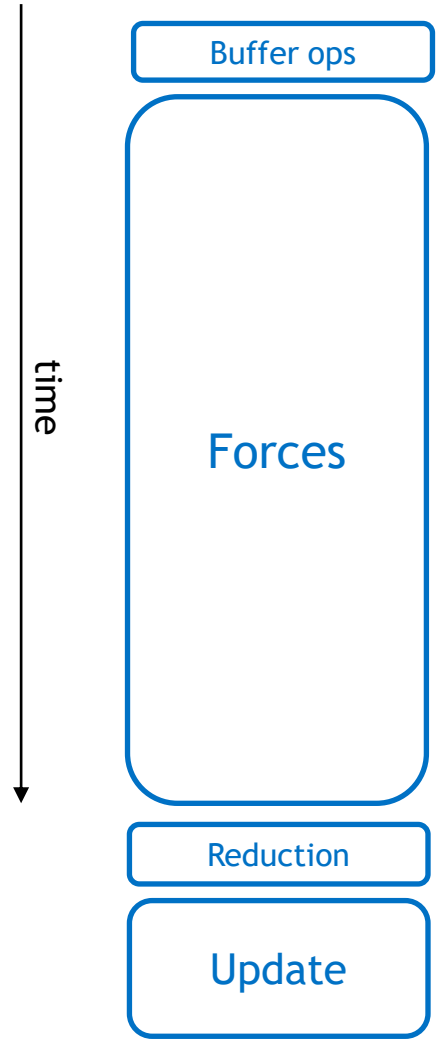

v2020: large improvements with introduction of GPU-resident step

v2021: main focus on code quality improvement, but some improvements with kernel tuning

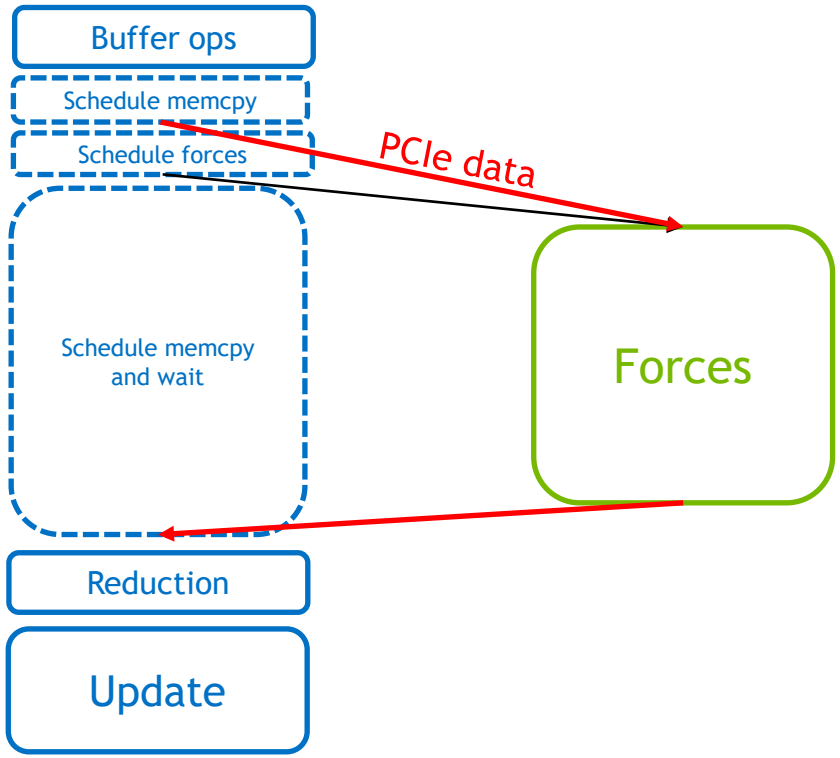
v2022: A number of improvements affecting different cases in different ways, esp. aimed at improved overlapping with heterogeneous force calcs (EAG1) and reduced sensitivity to API overheads (ADH Dodec).

Also, large multi-node improvements in v2022 through CUDA-aware MPI support and in upcoming v2023 through PME GPU decomposition (not covered here).

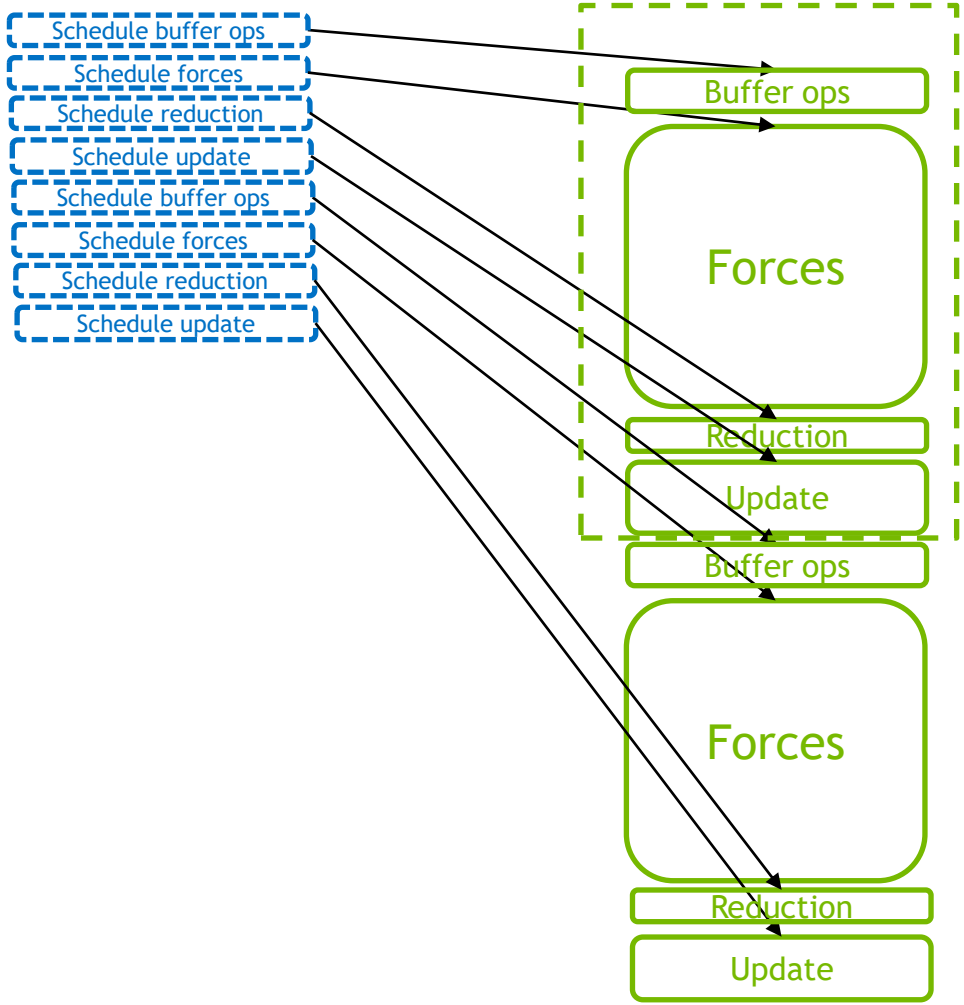
# GROMACS REGULAR TIMESTEP (SINGLE-GPU)




CPU-only (with multi-core threading)



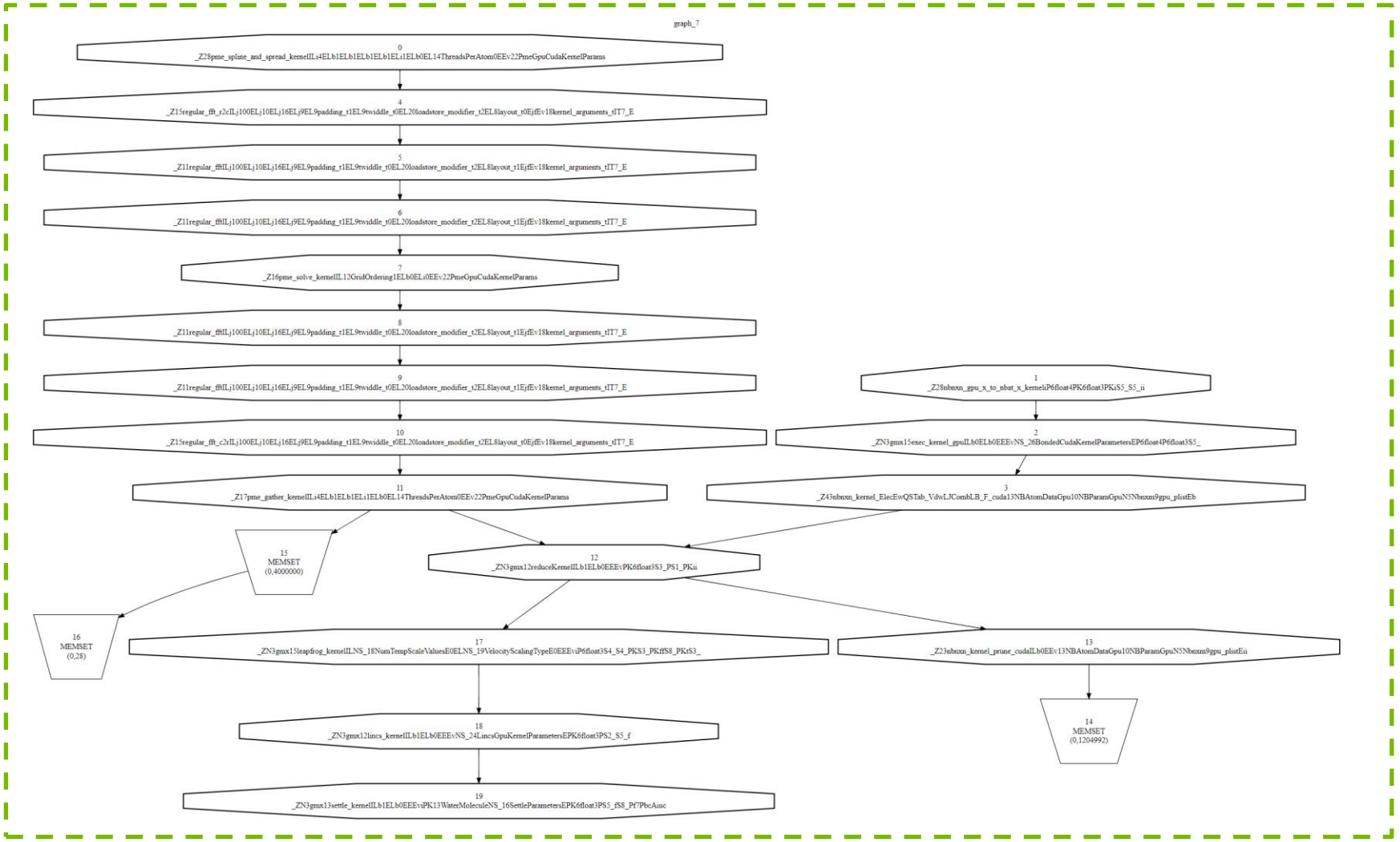

Pre-v2020: GPU acceleration of forces only.



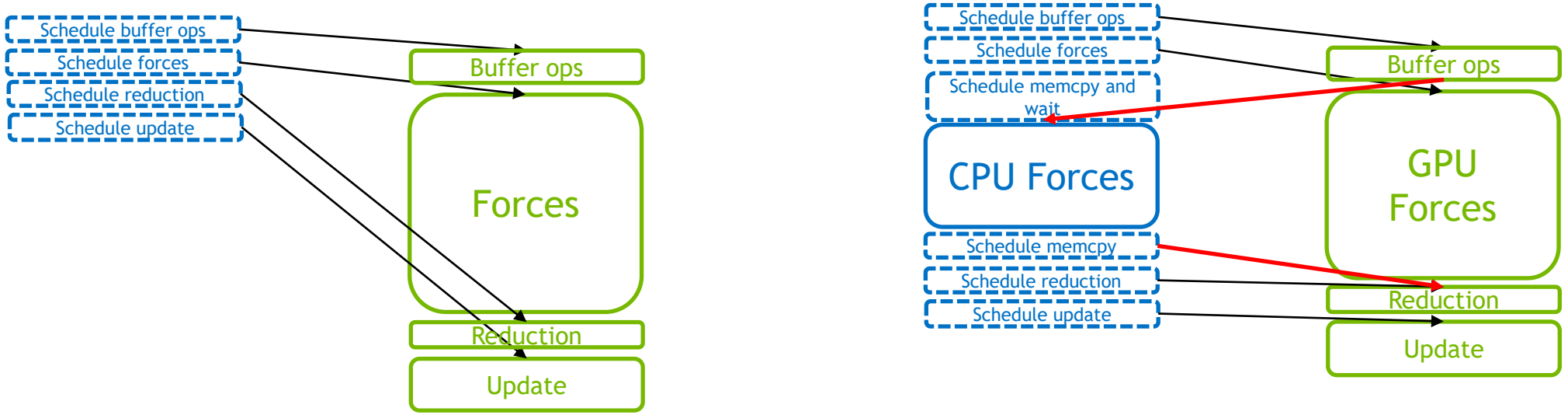

GPU-resident step with asynchronous task scheduling (from v2020)

https://developer.nvidia.com/blog/creating-faster-molecular-dynamics-simulations-with-gromacs-2020/

# ACTUAL GPU-SIDE ASYNC ACTIVITIES ON GPU-RESIDENT STEP



# HETEROGENEOUS FORCES




GROMACS supports heterogeneous force calculations.

- Some cases include force components which don't support GPUs (including EAG1).
- Bonded forces can optionally be assigned to either GPU or CPU `(-bonded {cpu,gpu})`

These are typically relatively cheap, and can be "back-offloaded" to CPU in shadow of main GPU kernel without disrupting critical path.

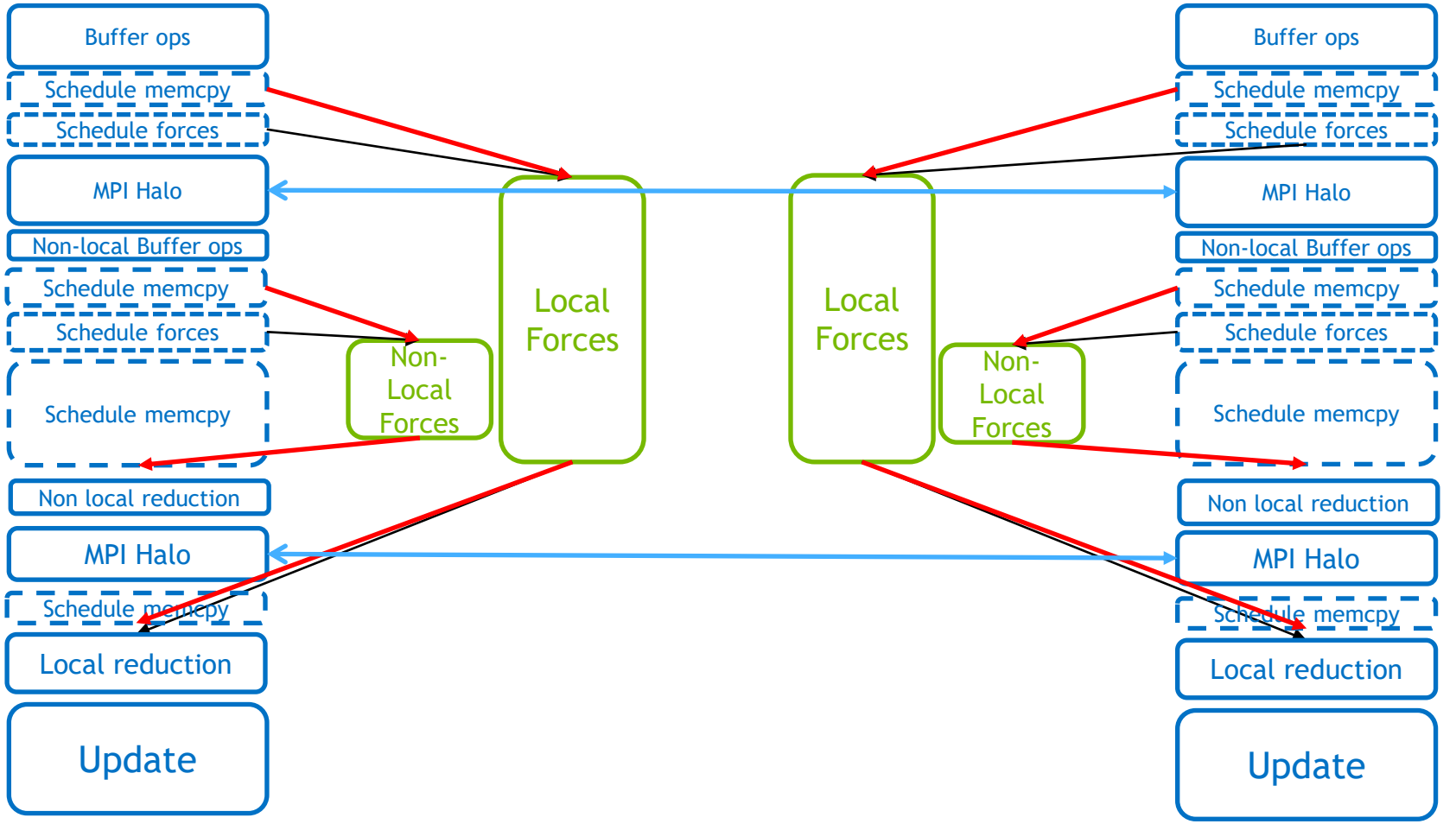
- Often, all data transfer and CPU compute can remain completely overlapped

Can offer performance advantages by reducing load on GPU. Depends on hardware (users should experiment with bonded force assignment).

Upcoming NVIDIA Grace+Hopper architecture especially suited.

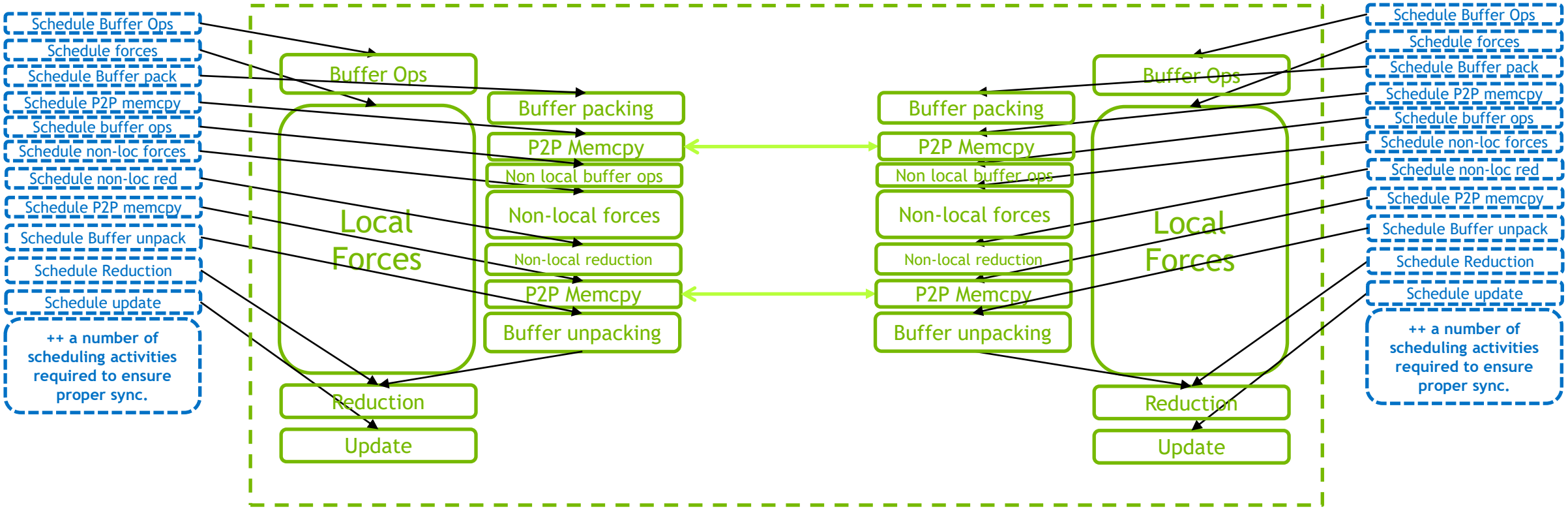
New developments in v2022 improve asynchrony of such heterogeneous cases on multi-GPU.

# GROMACS ON MULTI GPU




Pre GROMACS v2020. Shows PP decomposed across 2 GPUs (without PME). Only forces are GPU-accelerated.

# ASYNCRONOUS GPU-RESIDENT MULTI-GPU STEPS

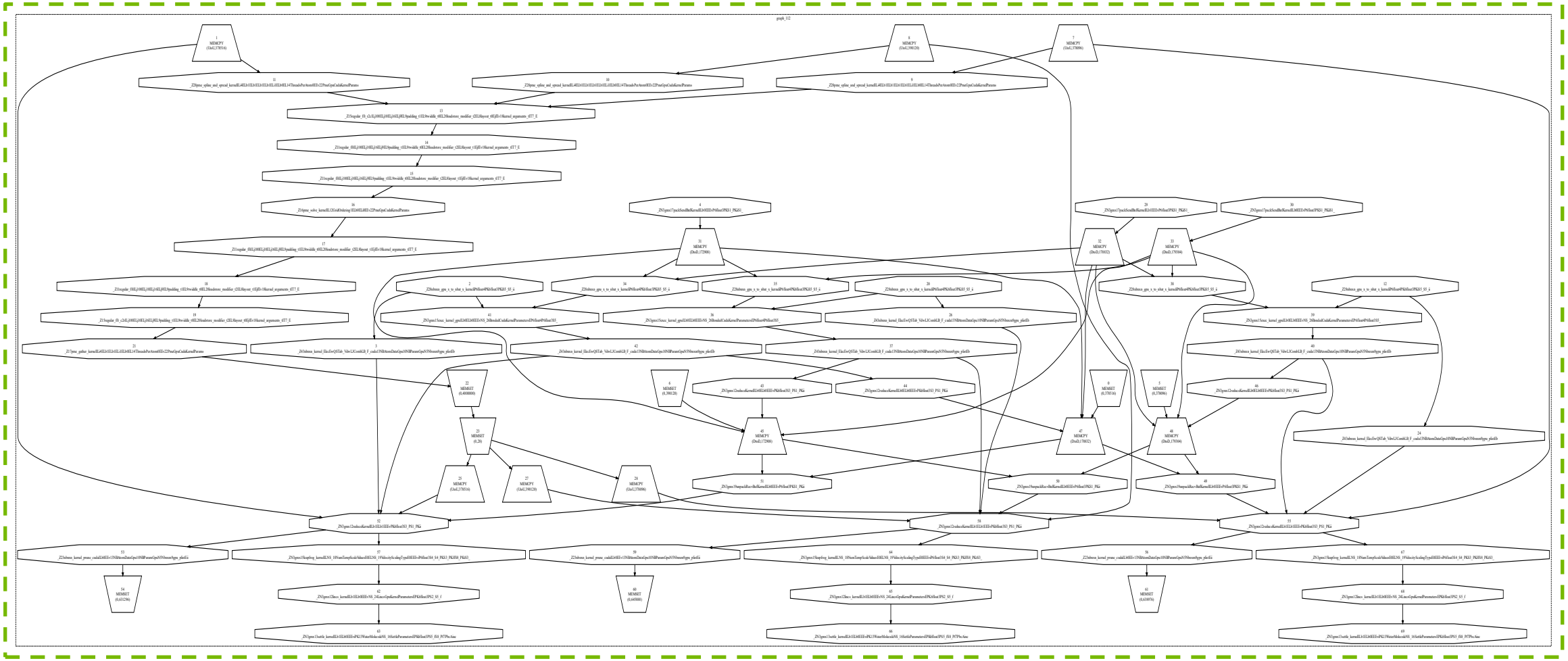



Latest GROMACS. Shows PP decomposed across 2 GPUs (without PME).

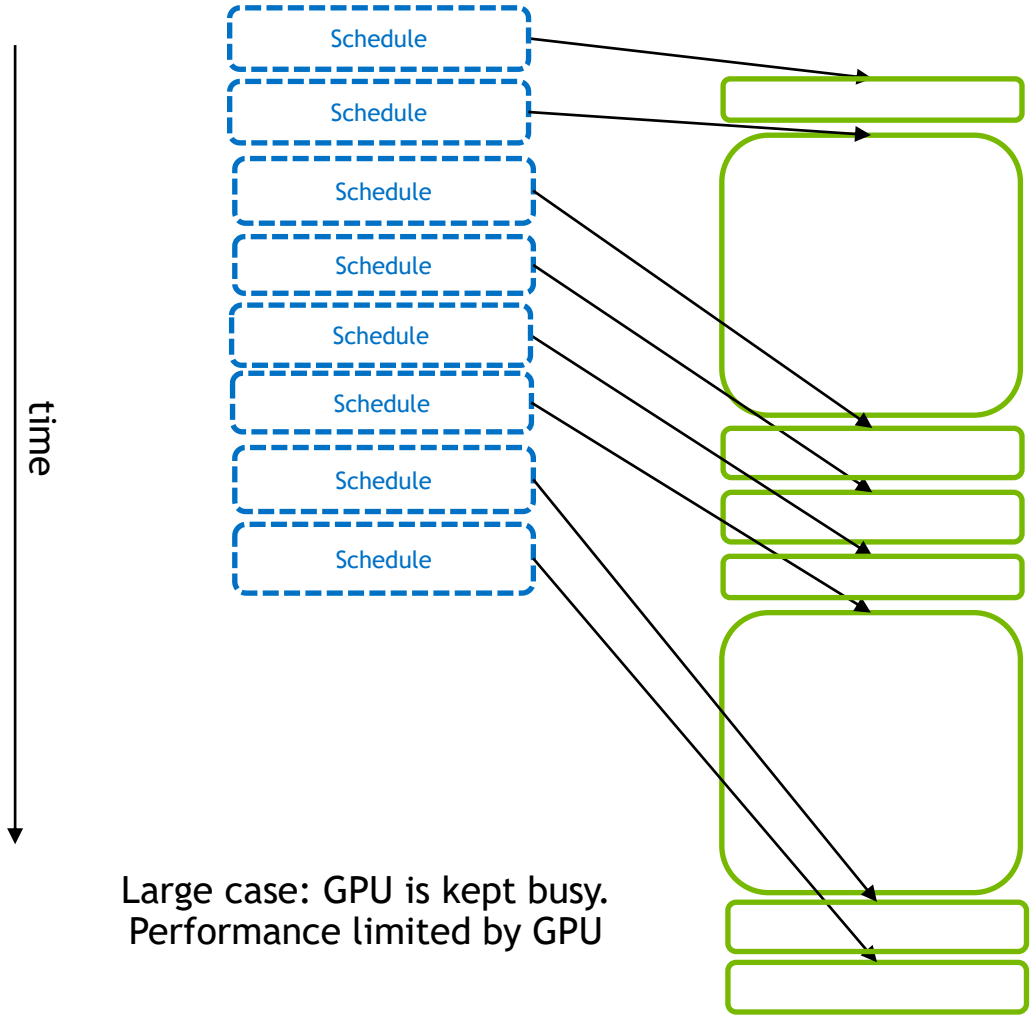
In practice, have more PP GPUs and also have PME assigned to a separate GPU (PME GPU decomposition currently in development). Many more GPU and scheduling activities.

Dashed green box shows asynchronous GPU component. Following slide shows *actual* equivalent for 3xPP + 1xPME GPUs

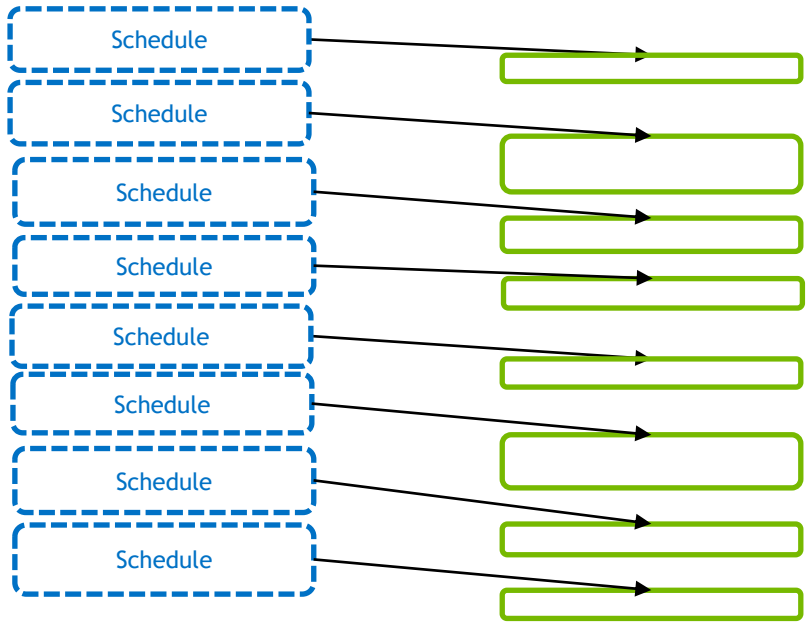
# ACTUAL GPU-SIDE ASYNC ACTIVITIES ON 4XGPU RESIDENT STEP



# SCHEDULING OVERHEADS




Large case: GPU is kept busy.
Performance limited by GPU




Small case: GPU is starved.
Performance is limited by
CPU scheduling overheads

Biomolecular systems have a fixed number of atoms, which poses challenges in the continued drive to expose more and more parallelism.

For many cases, we have reached the point where each GPU activity is now so fast that CPU scheduling becomes the limiting factor. Especially on multi-GPU, where there are a lot of sync-related scheduling activities.

# CO-DESIGN TO IMPROVE SCHEDULING OVERHEADS

Several recent developments have strived to reduce CPU scheduling overheads

Both at the application level (CPU threading of scheduling calls associated with PME-PP data transfer; several improvements to reduce CPU-side synchronization in scheduling code) and at the lower CUDA level (improvements in CUDA to reduce serialization overhead when multiple CPU threads drive multiple GPUs).

As GPUs continue to get faster, this issue will become more serious in the future.

NVIDIA has developed the CUDA Graphs feature to address such problems,

- We are working to integrate in GROMACS
- A patch with initial support for cases that support GPU-resident steps is under review
- Offers significant performance benefits
- https://gitlab.com/gromacs/gromacs/-/issues/4277

Continued 2-way improvements in co-design with CUDA team.

# CUDA GRAPHS

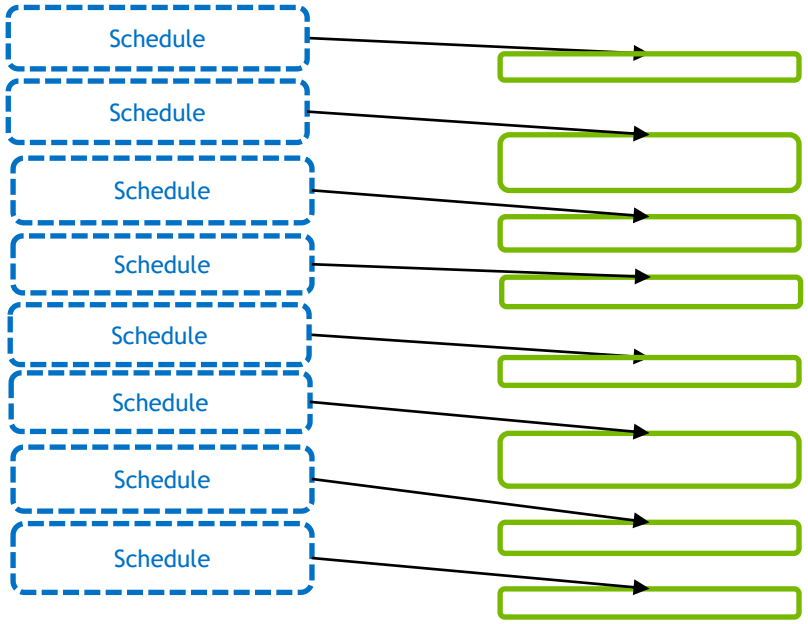
As it stands, GROMACS launches each CUDA activity independently. For small cases, the CPU overheads associated with launches are on the critical path, such that the GPU is "starved" of work.

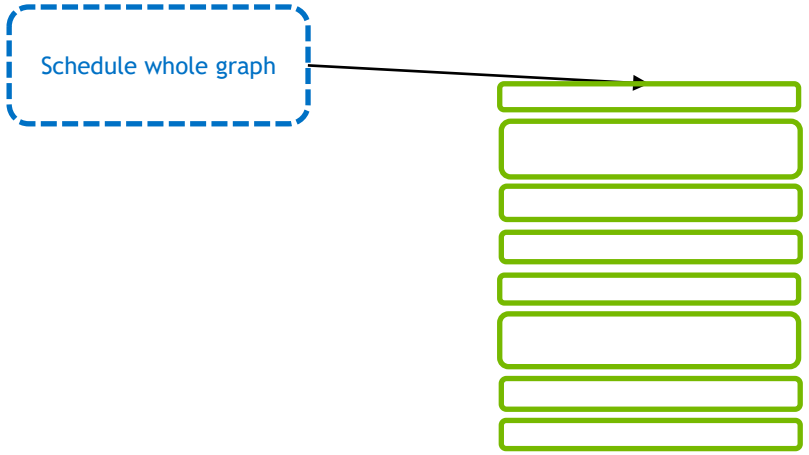
CUDA Graphs aims to address this problem by allowing multiple activities to be launched as a single "graph", such that a single CPU API call can launch multiple GPU activities.

Graph can be defined through “capturing” of existing stream-based CUDA code.

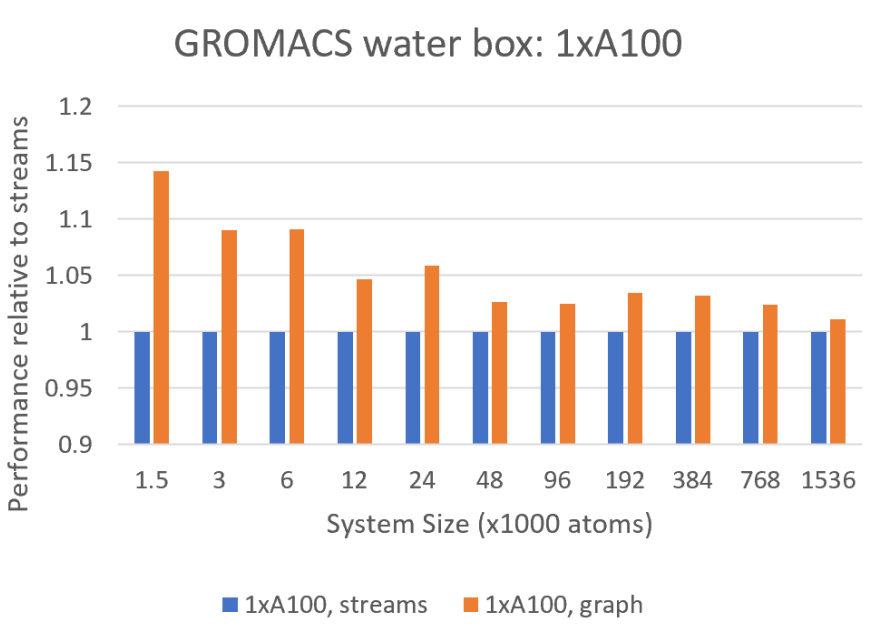
There are also benefits on the GPU side: since CUDA has more awareness about the workflow it can optimize execution and reduce GPU-side launch latencies.

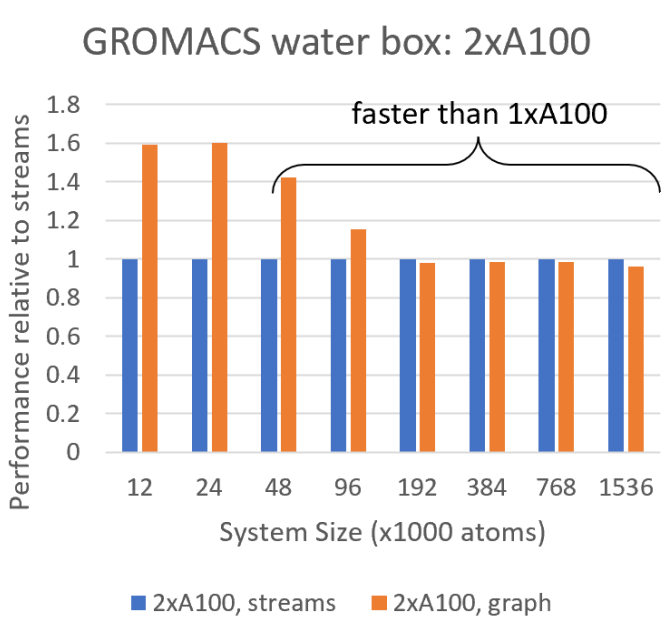
Getting Started with CUDA Graphs: https://developer.nvidia.com/blog/cuda-graphs/

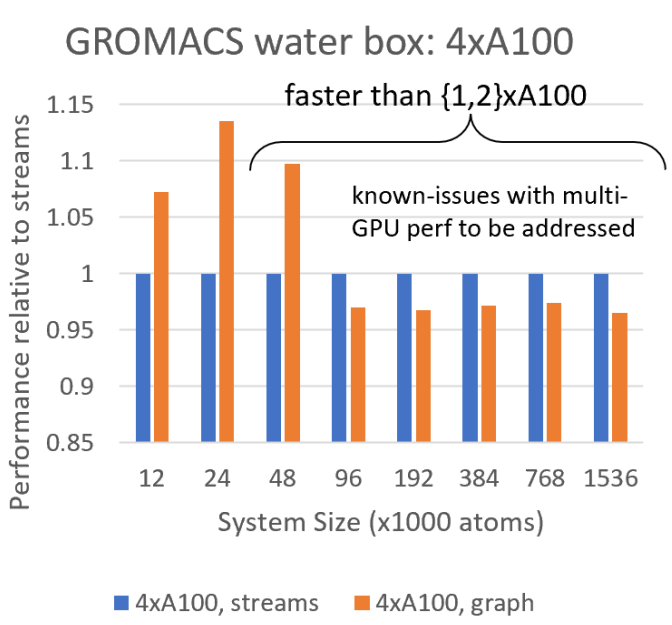


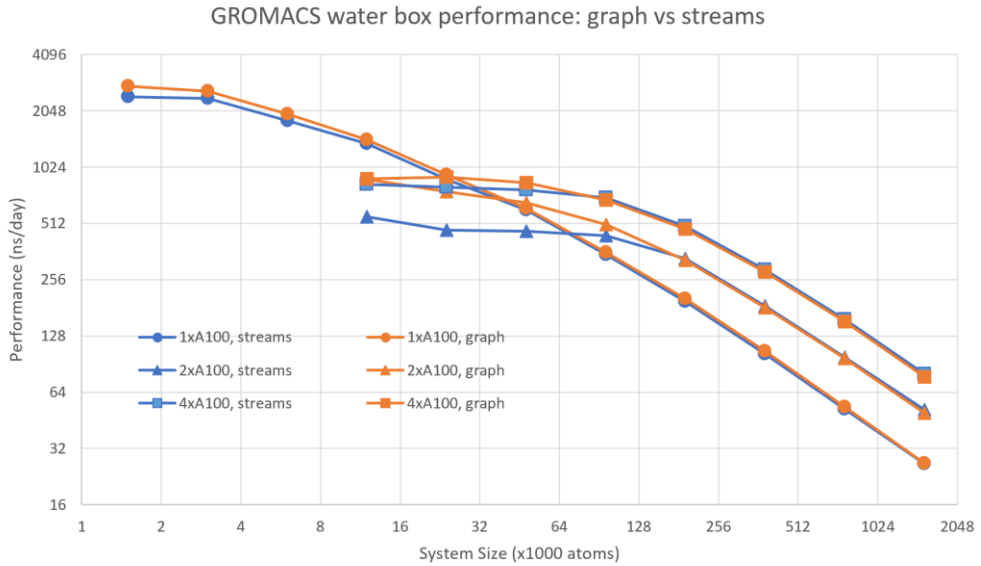



# GROMACS PERFORMANCE WITH CUDA GRAPHS



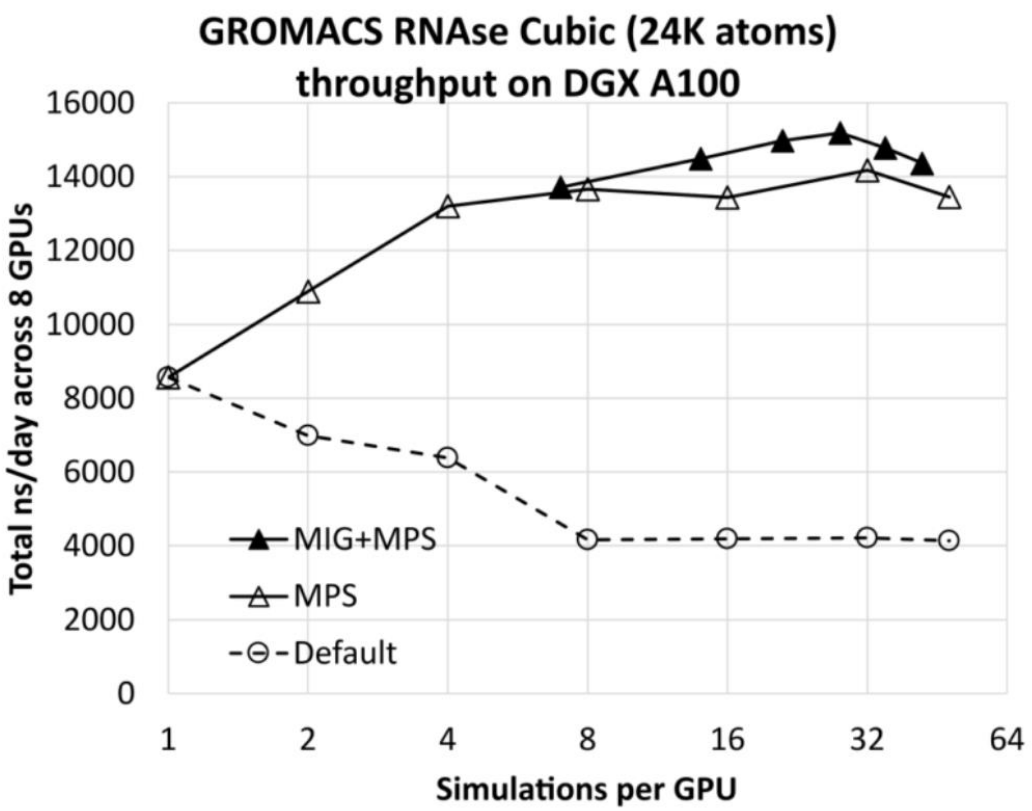



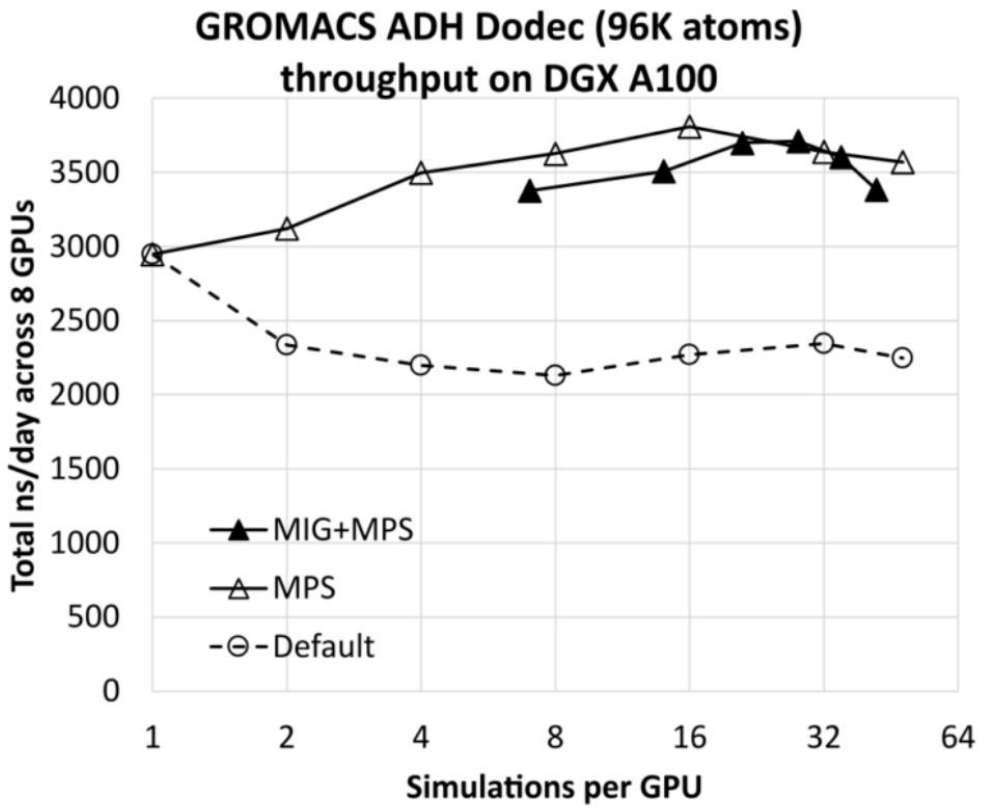




The smaller the system, the higher the benefit from use of graphs.

Further work to push region of benefit further out to larger systems, particularly for multi-GPU.

# ULTIMATE ASYNCHRONY: MULTIPLE SIMULATIONS PER GPU




Maximizing GROMACS Throughput with Multiple Simulations per GPU Using MPS and MIG | NVIDIA Technical Blog

https://developer.nvidia.com/blog/maximizing-gromacs-throughput-with-multiple-simulations-per-gpu-using-mps-and-mig/

# SUMMARY

GROMACS continues to evolve to operate as efficiently as possible on modern heterogeneous servers

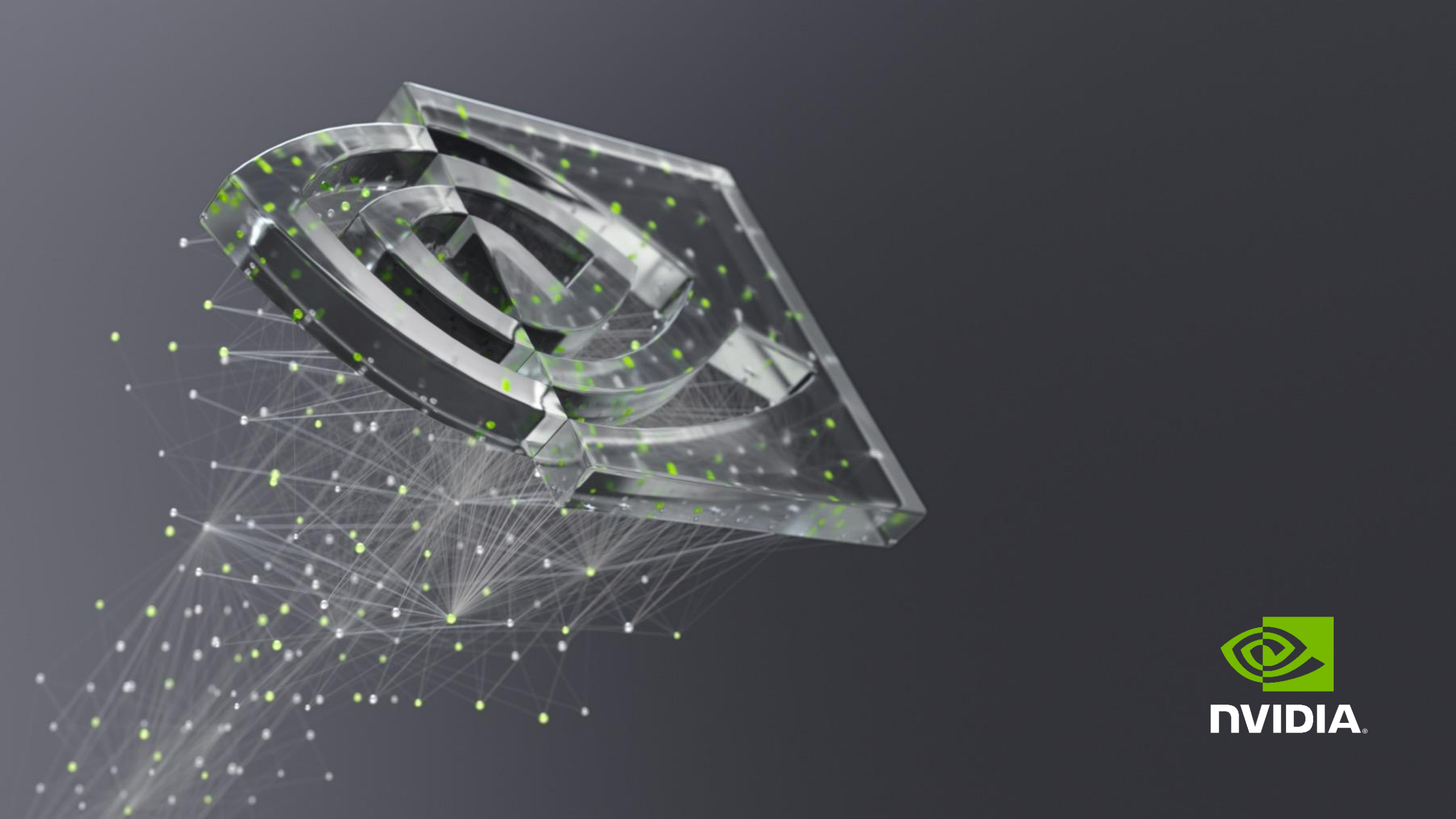
Dramatic performance improvements have been realized in the last few years through code adaptations towards efficient asynchronous use of the different hardware components.

As GPUs continue to become faster, the challenge becomes harder (especially for multi-GPU).

In particular, for many cases the time taken by the CPU to schedule tasks on the GPU is critical.

Much recent and planned work is aimed at minimizing scheduling overhead, in co-design with the NVIDIA CUDA platform.

For use-cases involving an ensemble of many simulations, there are large advantages to running multiple simulations per GPU in parallel.


NVIDIA