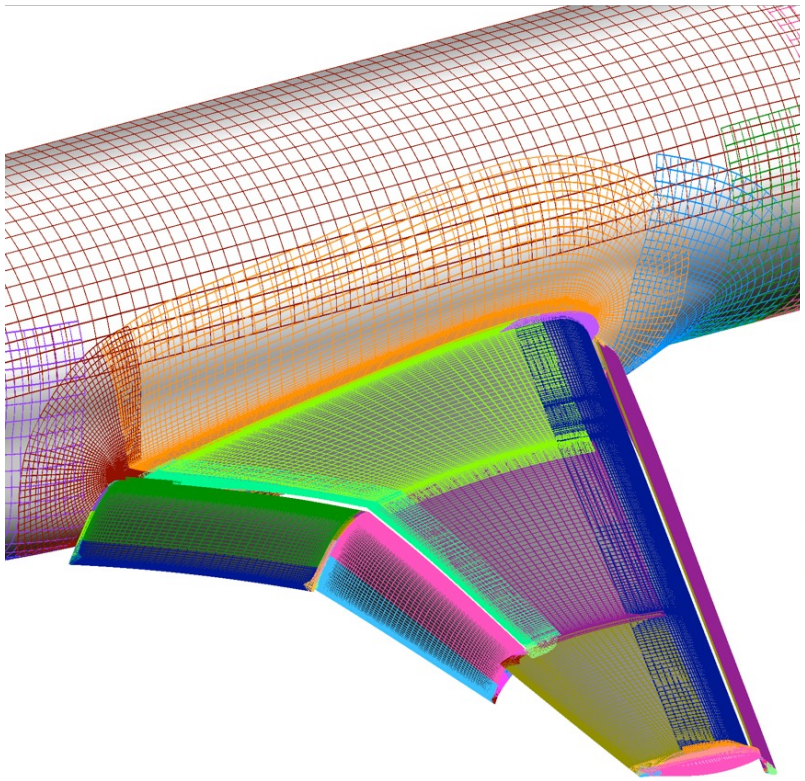
# GPU Implementation of the OVERFLOW CFD Code


| | |
|---|---|
| Chip Jackson | NASA Langley |
| David Appelhans | NVIDIA |
| Joe Derlaga | NASA Langley |
| Pieter Buning | NASA Langley |

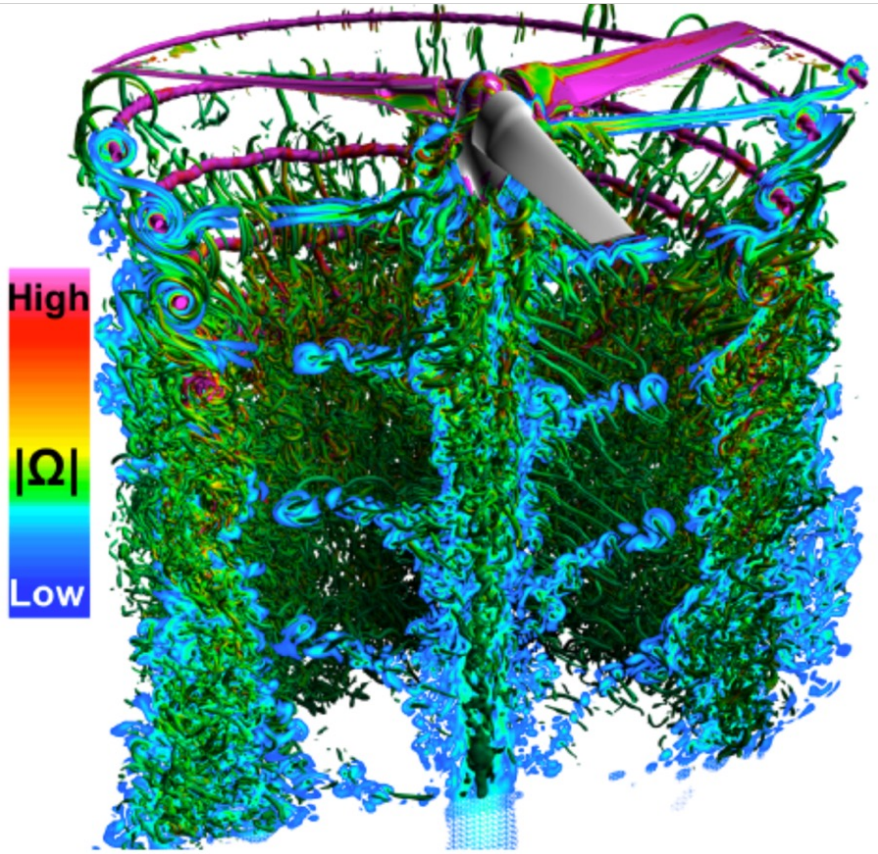

2024 AIAA SciTech

# OVERFLOW

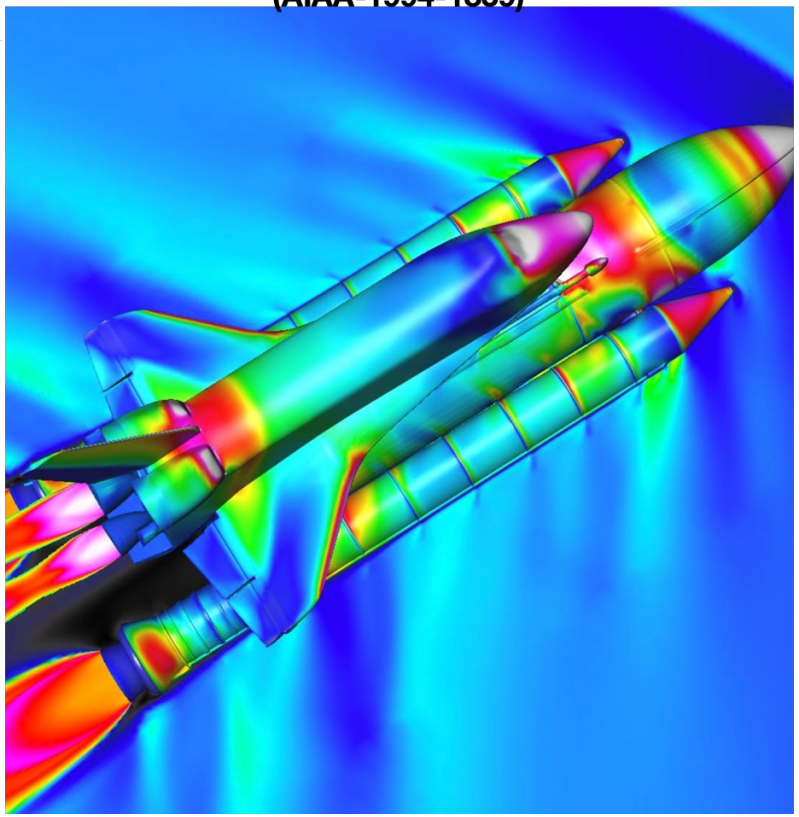
**High-Lift Common Research Model Grid System**
**(AIAA-2017-0362)**



**High Resolution Rotor Wake Simulation**
**(AHS 2011 Annual Forum)**



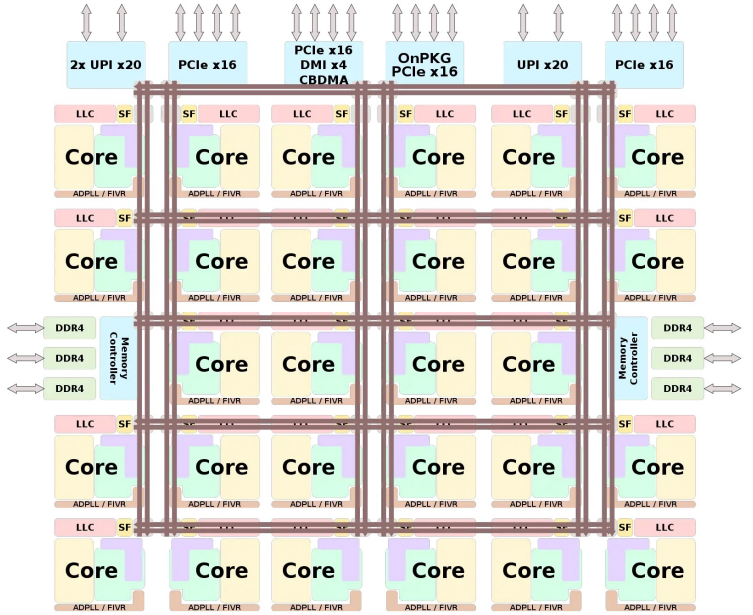

**Space Shuttle Launch Vehicle**
**Mach 1.25 Surface Pressure and Flow-Field Mach Number**
**(AIAA-1994-1859)**



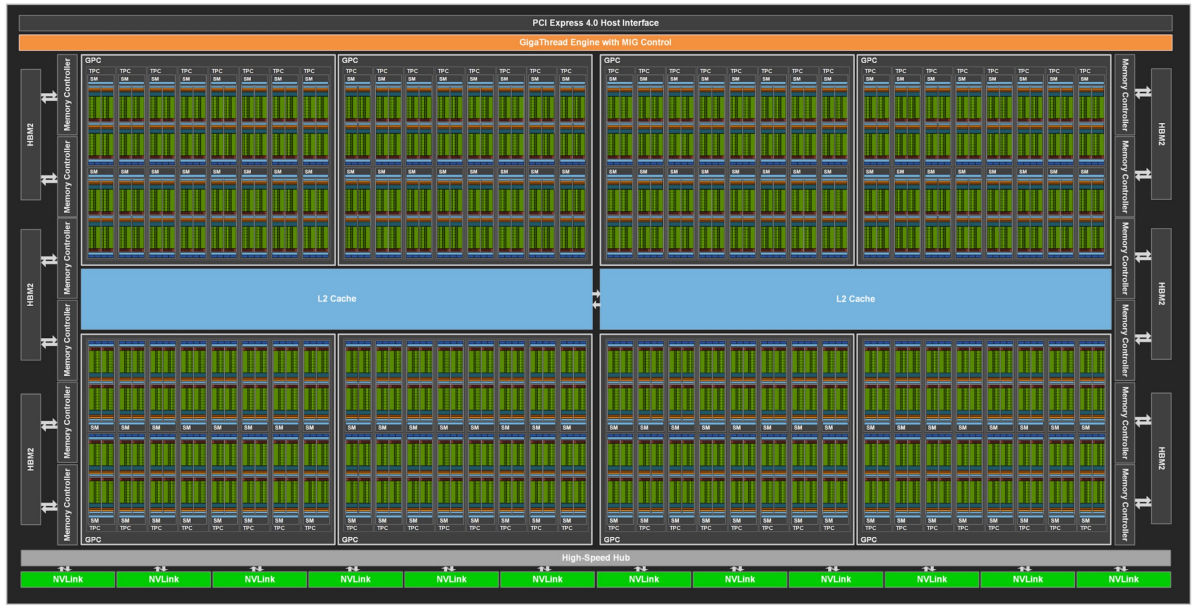
# Why GPUs?

- Higher Performance
  - Memory Bandwidth
  - Parallelism

- FLOPs/W
- FLOPs/$
- FLOPs/sqft




https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)




NVIDIA A100 Tensor Core GPU Architecture Whitepaper

# How do I run on GPUs?

- Many Options
- We selected OpenACC, CUDA Fortran, and CUDA C++

# Code Modifications

- Data Movement
- Asynchronous Kernel Launching
- Loop Grouping and Grid Batching
- Data Ordering
- Merging Kernels
- Shared Memory
- Multiple GPUs

# Code Modifications

- Data Movement
- Asynchronous Kernel Launching
- Loop Grouping and Grid Batching
- Data Ordering
- Merging Kernels
- Shared Memory
- Multiple GPUs

Expose Parallelism

# Code Modifications

- Data Movement
- Asynchronous Kernel Launching
- Loop Grouping and Grid Batching
- Data Ordering
- Merging Kernels
- Shared Memory
- Multiple GPUs

Reduce Memory Traffic

# Code Modifications

- Data Movement
- Asynchronous Kernel Launching
- Loop Grouping and Grid Batching
- Data Ordering
- Merging Kernels
- Shared Memory
- Multiple GPUs

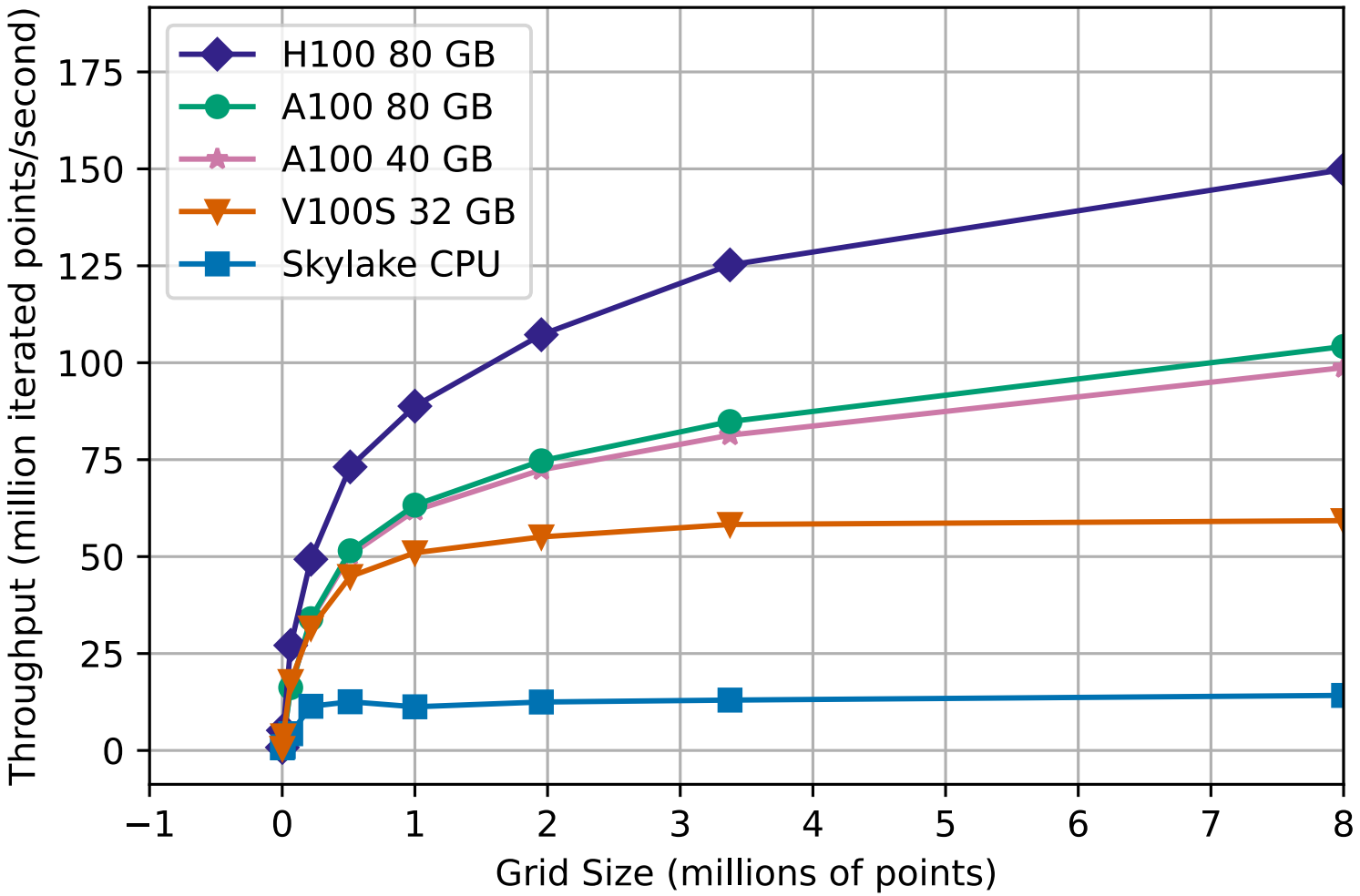
Improve Data Access

# Current Status

- Significant changes resulted in two paths through OVERFLOW
  - Same startup, shutdown routines
  - Some shared routines, like boundary conditions and I/O routines
- Implemented the central difference scheme, with scalar pentadiagonal solver, SA turbulence model
- Using GPU-aware MPI to communicate between multiple GPUs

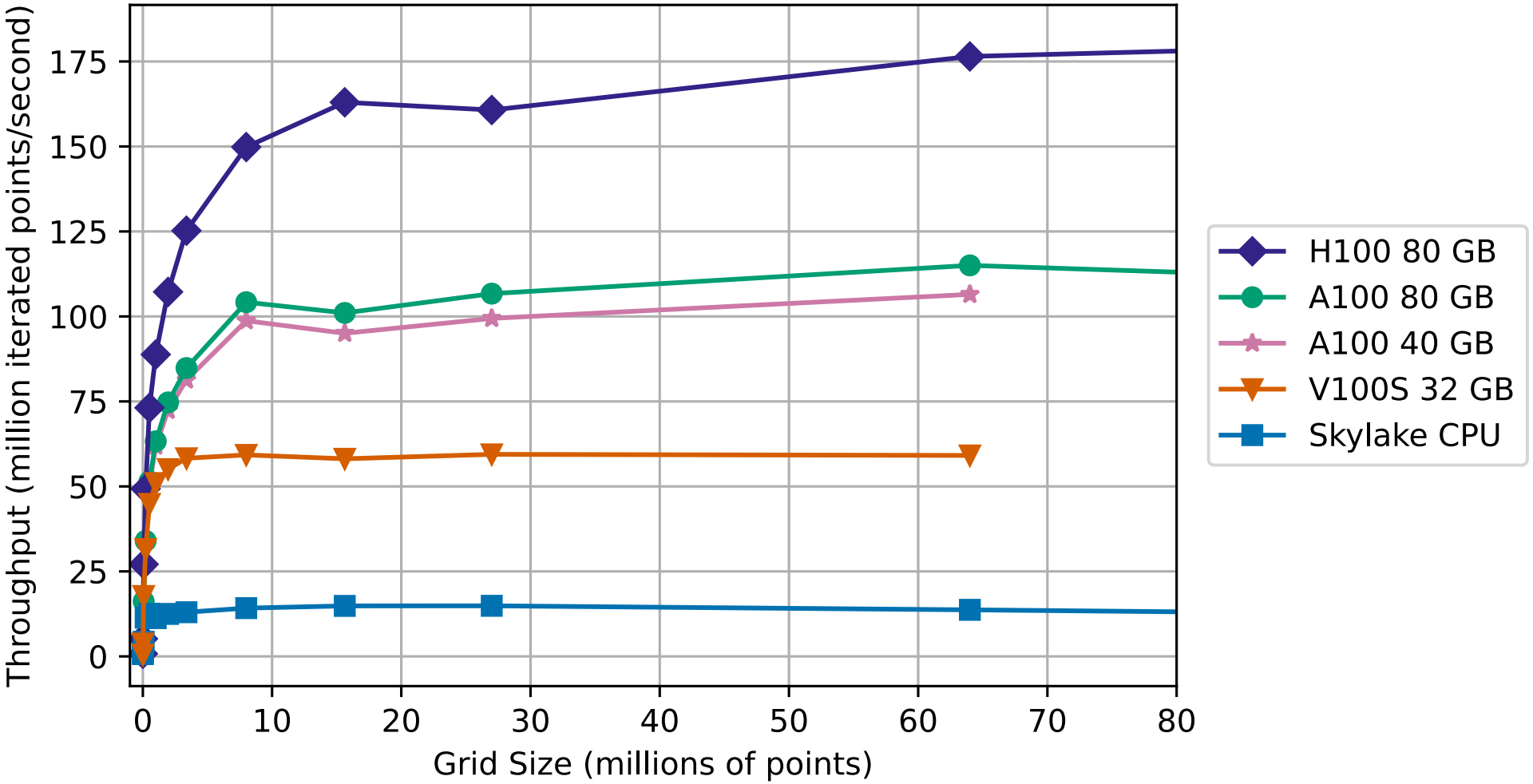
# How big of a grid do I run?

# How big of a grid do I run?



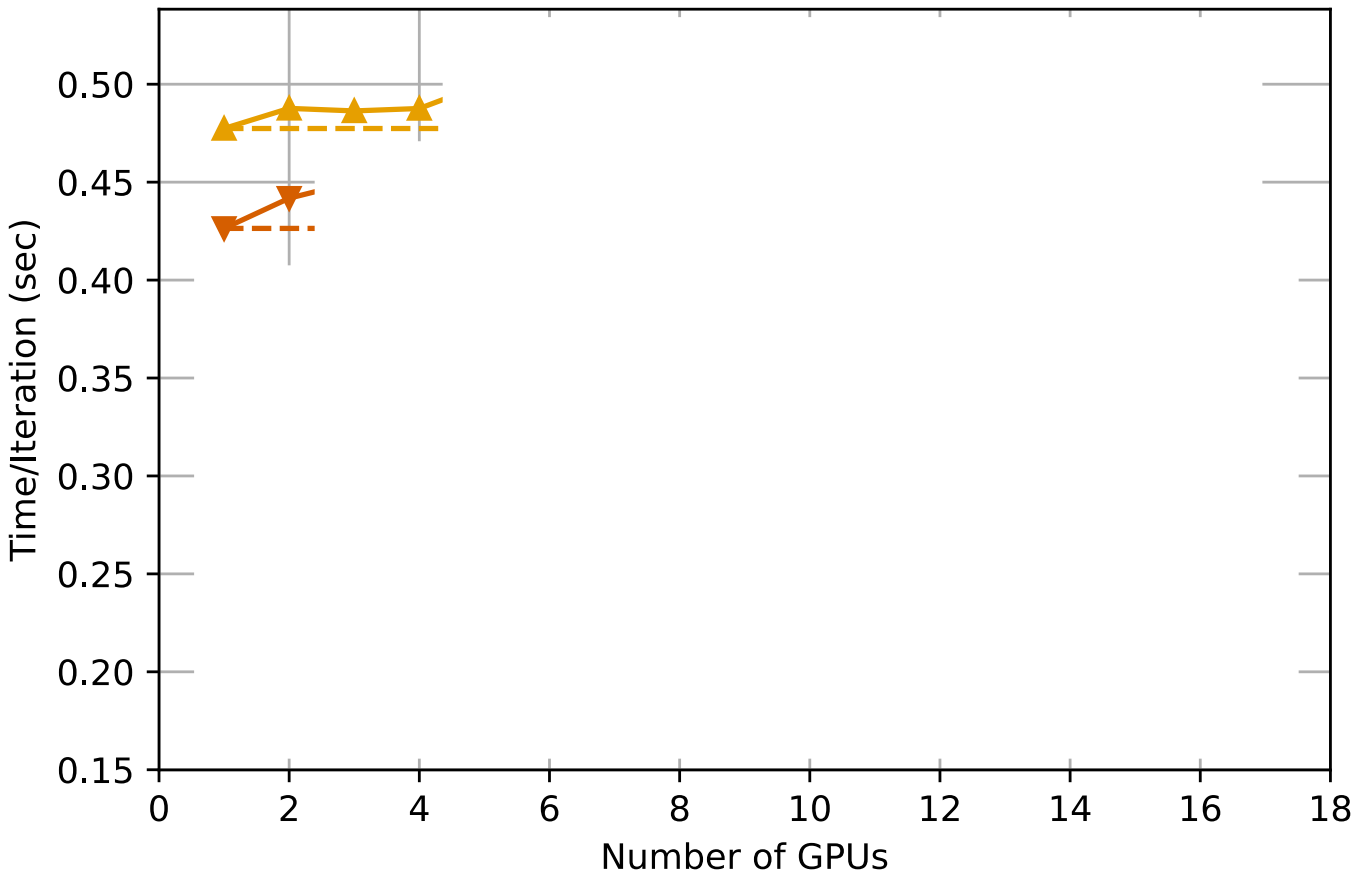

Skylake CPU is a dual socket, 20-core Xeon Gold 6148 node running at 2.4 GHz

# How big of a grid do I run?



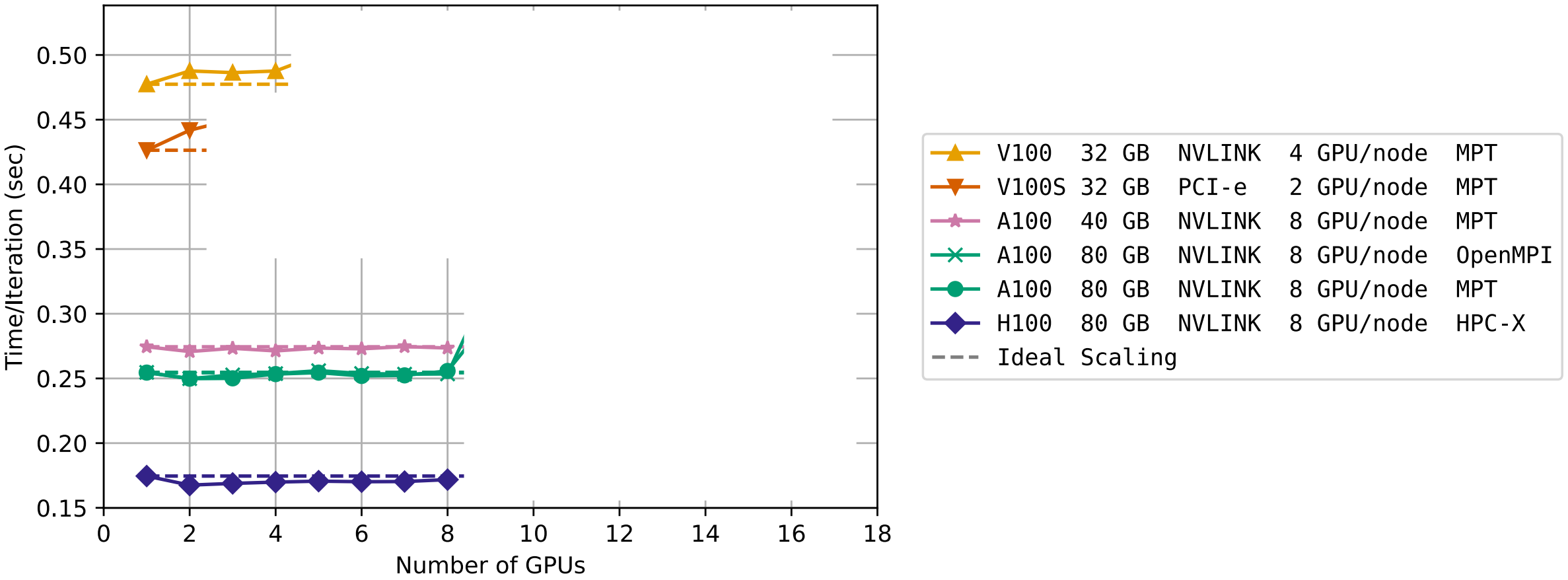
# How well does it run on multiple GPUs?

# How well does it run on multiple GPUs?

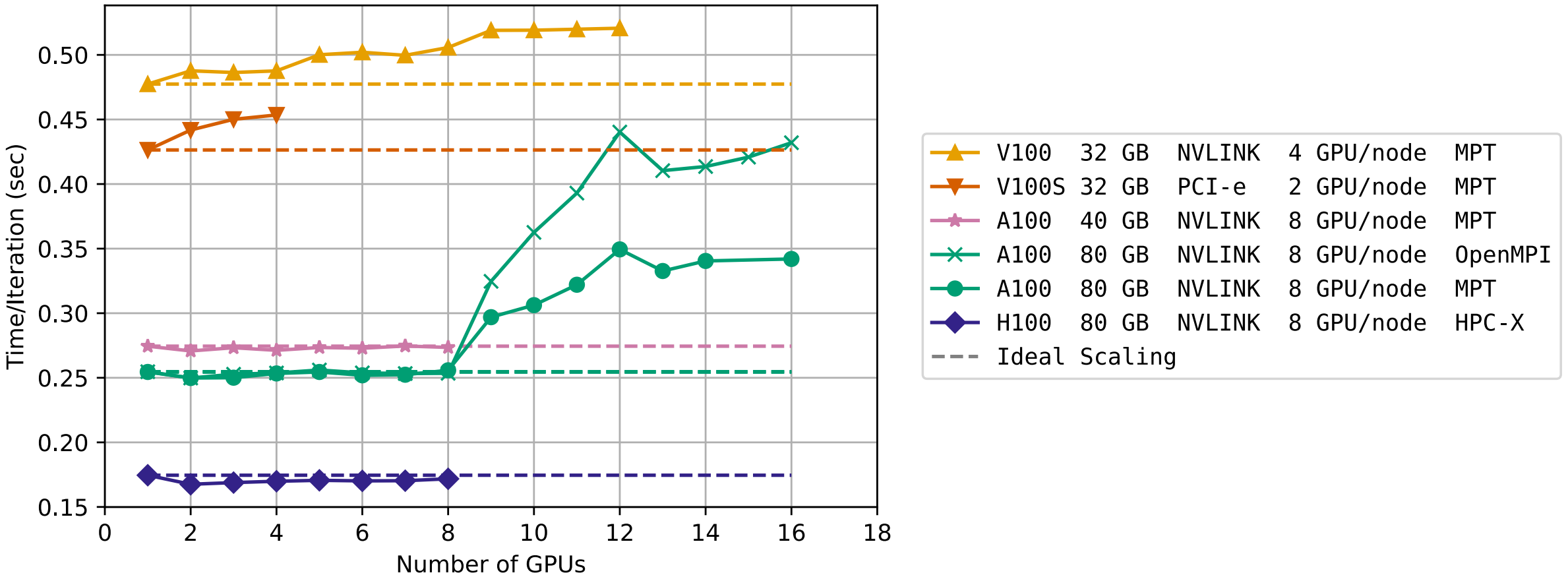



| | | | | |
|---|---|---|---|---|
| V100 | 32 GB | NVLINK | 4 GPU/node | MPT |
| V100S | 32 GB | PCI-e | 2 GPU/node | MPT |
| A100 | 40 GB | NVLINK | 8 GPU/node | MPT |
| A100 | 80 GB | NVLINK | 8 GPU/node | OpenMPI |
| A100 | 80 GB | NVLINK | 8 GPU/node | MPT |
| H100 | 80 GB | NVLINK | 8 GPU/node | HPC-X |
| Ideal Scaling | | | | |

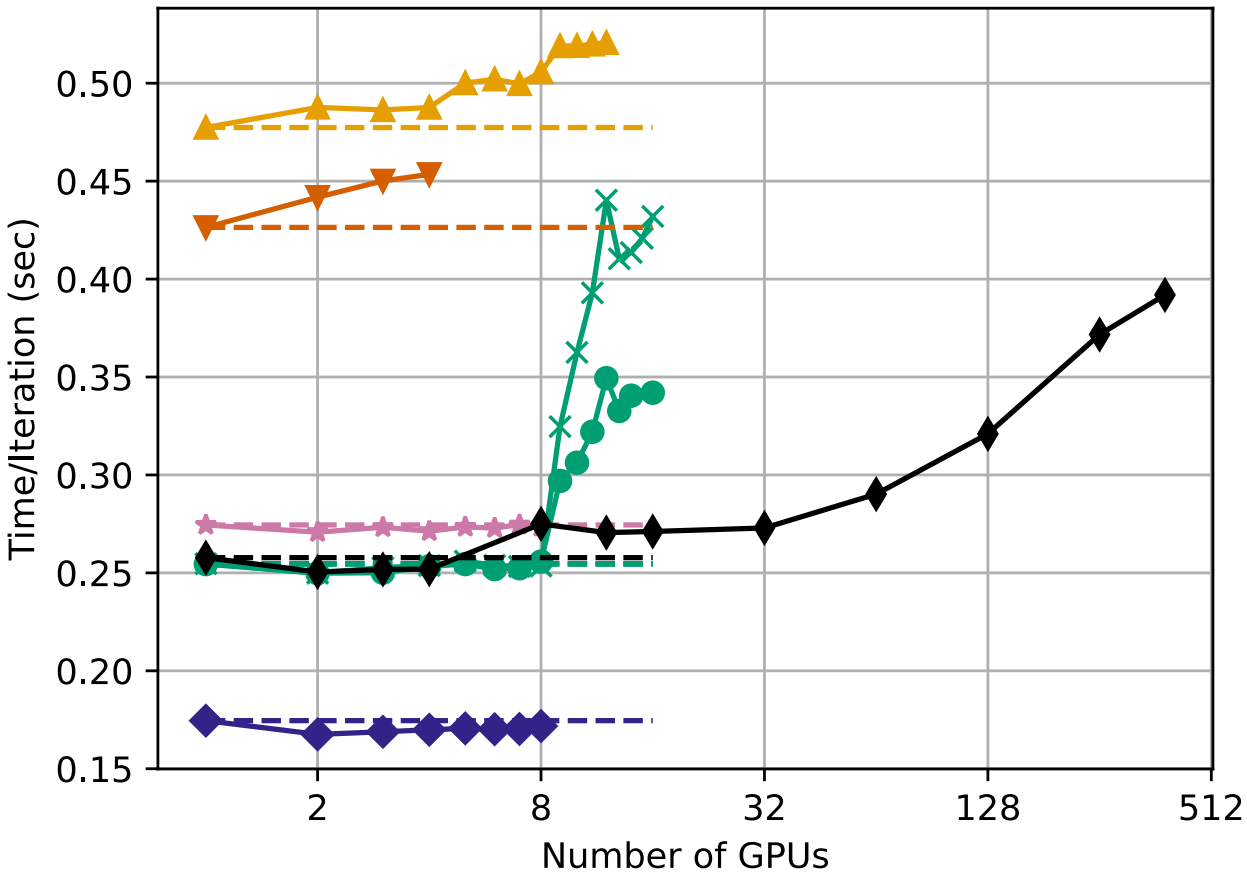
# How well does it run on multiple GPUs?



# How well does it run on multiple GPUs?



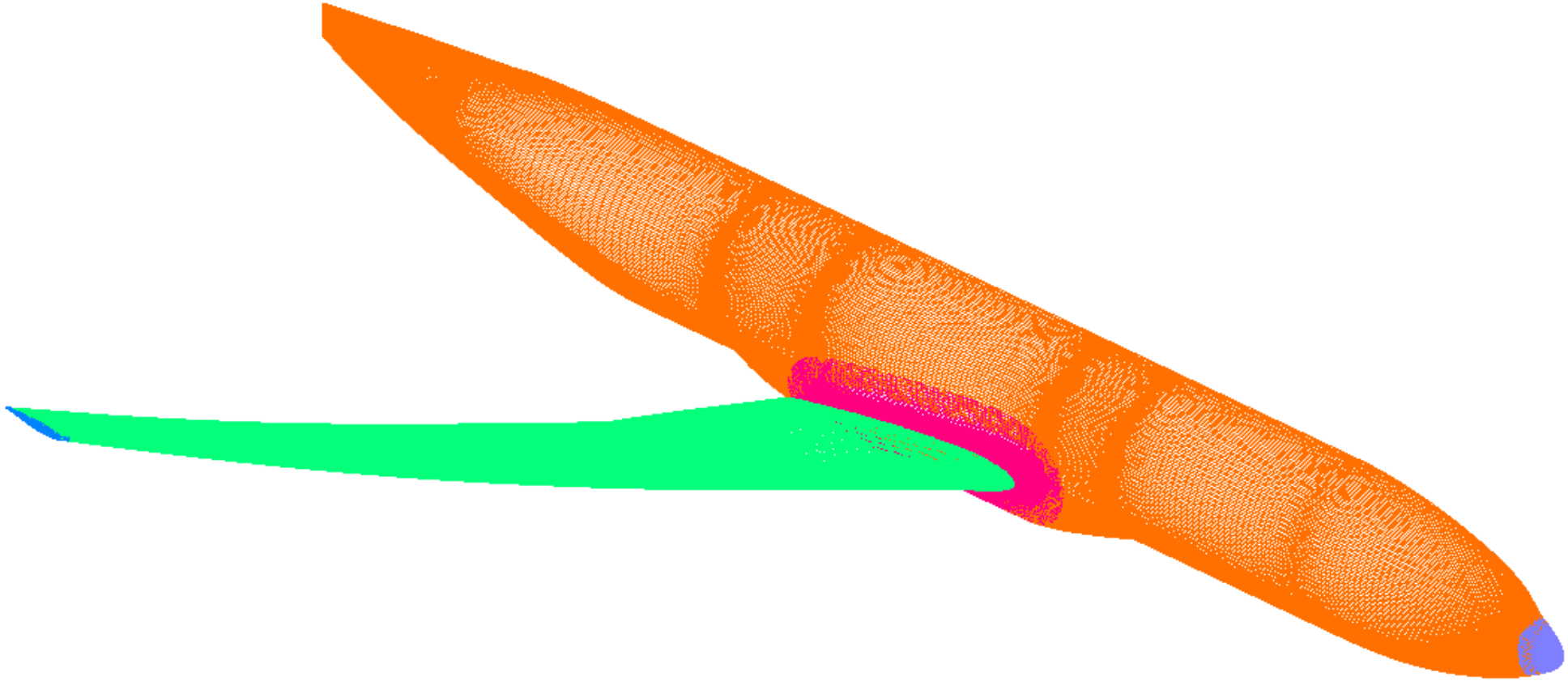
# How well does it run on multiple GPUs?



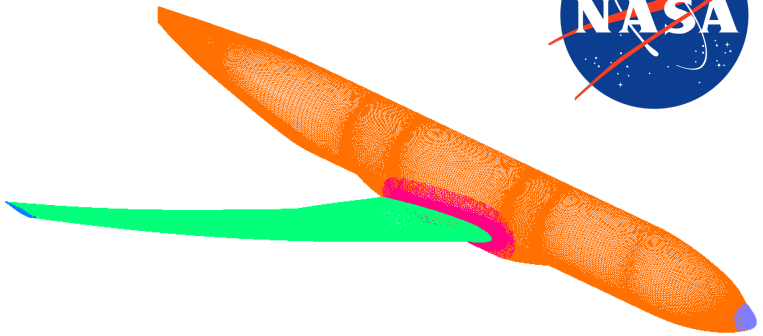

| | | | | | |
|---|---|---|---|---|---|
| A100 | 80 GB | NVLINK | 4 GPU/node | MPT | NAS |
| V100 | 32 GB | NVLINK | 4 GPU/node | MPT | |
| V100S | 32 GB | PCI-e | 2 GPU/node | MPT | |
| A100 | 40 GB | NVLINK | 8 GPU/node | MPT | |
| A100 | 80 GB | NVLINK | 8 GPU/node | OpenMPI | |
| A100 | 80 GB | NVLINK | 8 GPU/node | MPT | |
| H100 | 80 GB | NVLINK | 8 GPU/node | HPC-X | |
| Ideal Scaling | | | | | |

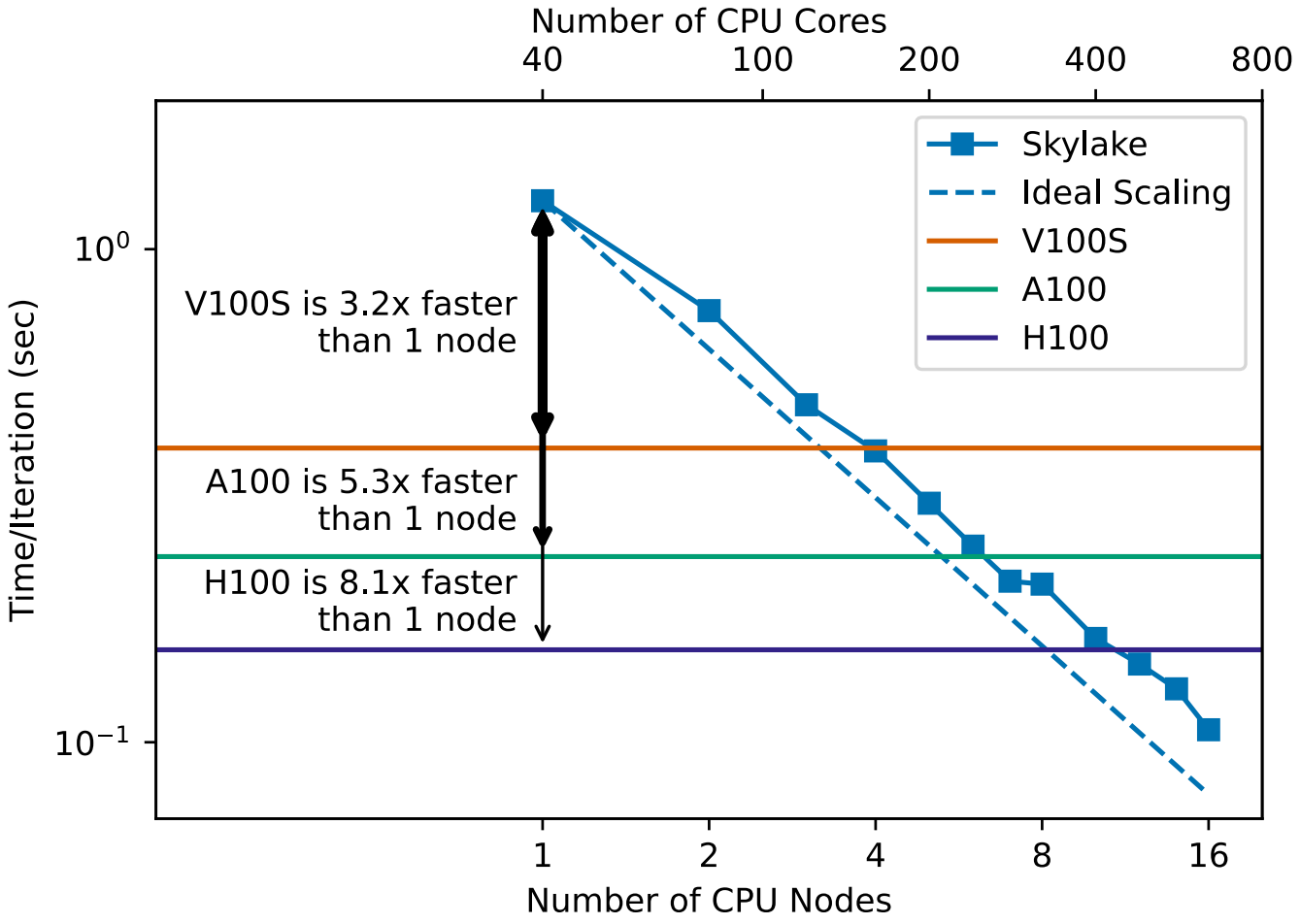
# What about a real problem?

# Drag Prediction Workshop 6 Case



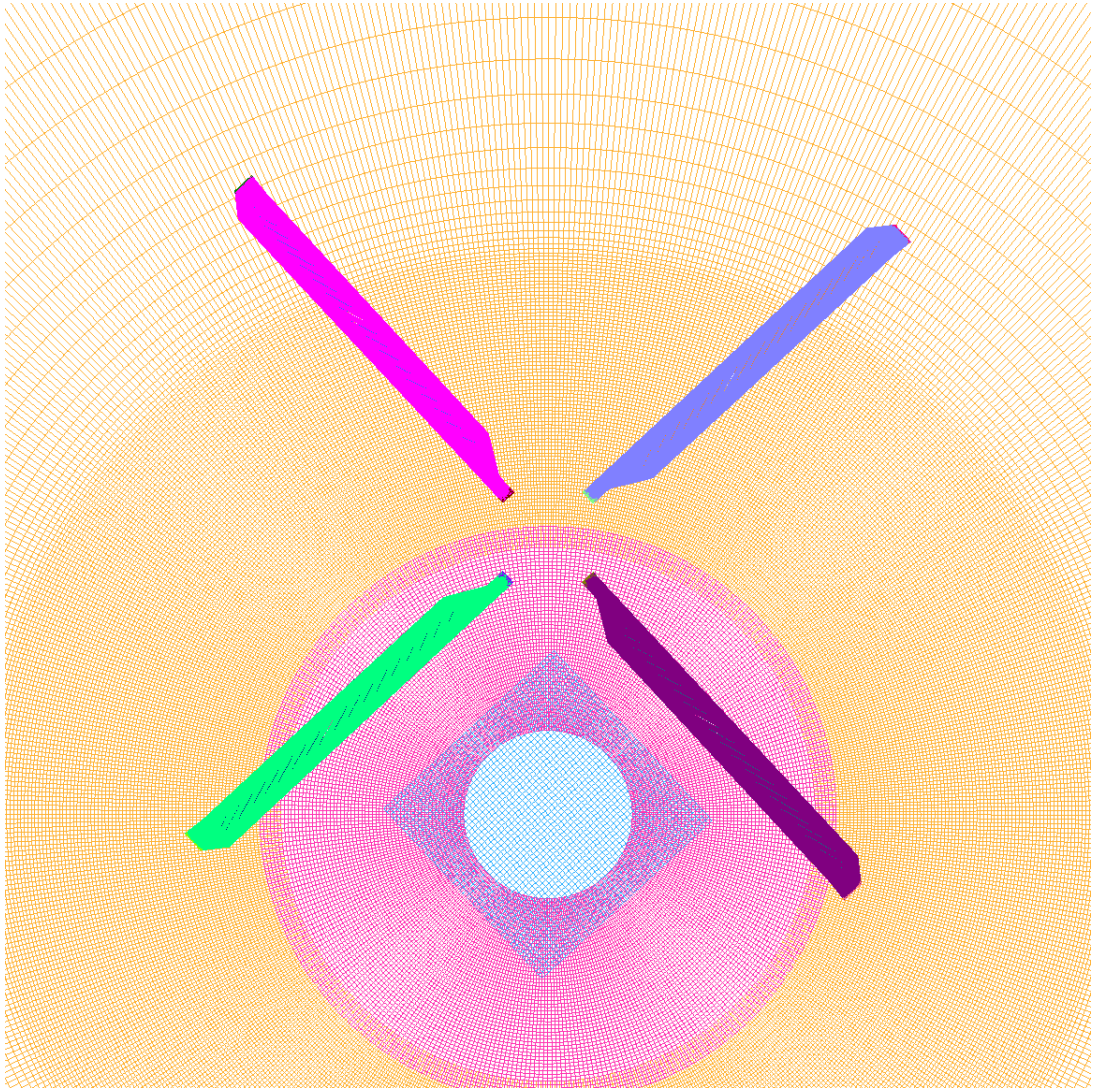
# DPW6 Case



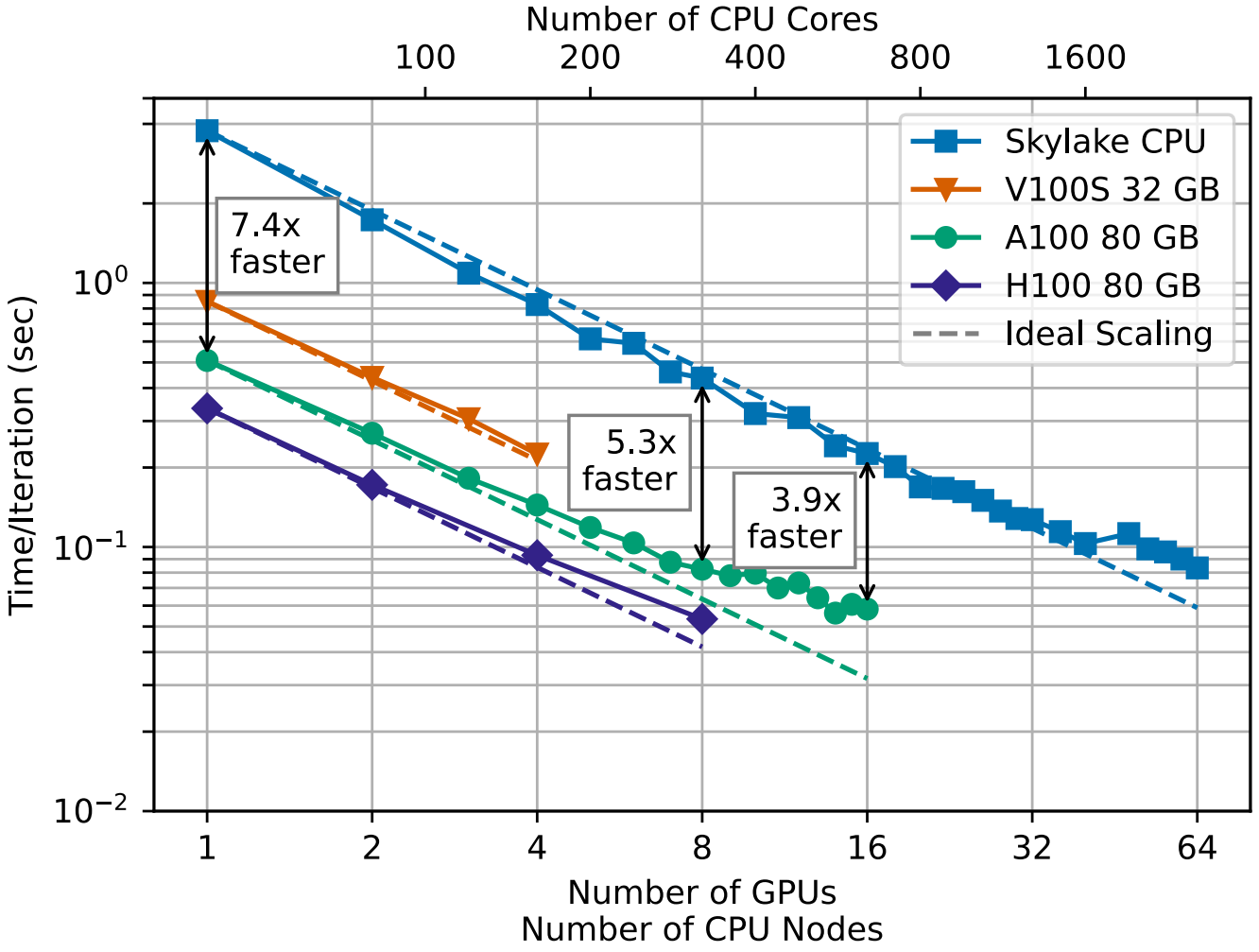


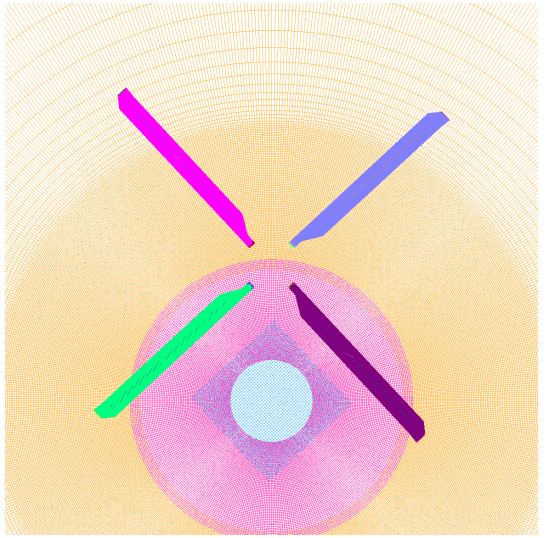

| GPU | Speedup Relative to Skylake Node | Equivalent Number of Skylake Cores |
|---|---|---|
| V100S | 3.2 | 158 |
| A100 | 5.3 | 253 |
| H100 | 8.1 | 437 |

# Hover Validation Acoustic Baseline Case



# HVAB Case



| GPU | Speedup Relative to Skylake Node | Equivalent Number of Skylake Cores |
|---|---|---|
| V100S | 4.4 | 156 |
| A100 | 7.4 | 265 |
| H100 | 11.3 | 390 |

# Concluding Remarks

- OVERFLOW is now able to run on GPUs
  - Central Difference Scheme, Scalar-Pentadiagonal LHS, Fixed Grids
  - Work is continuing on other options (moving grids, upwinding schemes, SSOR)
- Many lessons-learned and code modifications from this effort
- Running on GPUs is significantly faster than CPUs
  - 1 A100 GPU is 5 – 8 times faster than a dual socket Skylake node
  - 1 A100 GPU is equivalent to approximately 260 Skylake cores (7 nodes)
- Currently in testing mode and will be in the next major release of OVERFLOW

# Acknowledgements


- Would like to thank NASA's Revolutionary Vertical Lift Technology Project for funding this work.
- Also, thanks to NVIDIA and their technical staff for their support and assistance during this work.

# Questions?

# How do I run on GPUs?

| GPU Vendor | CUDA | | HIP | | SYCL | | Standard | | OpenMP | | OpenACC | | Kokkos | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | F | C | F | C | F | C | F | C | F | C | F | C | F |
| **NVIDIA** | ✓ | ✓ | ▼ | ⊘ | ▲ | ⊘ | ✓ | ✓ | ◆ | ◆ | ✓ | ✓ | ▲ | ⊘ |
| **AMD** | ▼ | ⊘ | ✓ | ⊘ | ▼ | ⊘ | ▲ | ⊘ | ✓ | ✓ | ▲ | ▲ | ▲ | ⊘ |
| **Intel** | ▼ | ⊘ | ⊘ | ⊘ | ✓ | ⊘ | ⊘ | ◆ | ✓ | ✓ | ⊘ | ⊘ | ▲ | ⊘ |

✓ Full Vendor Support

◆ Partial Vendor Support

▼ Indirect Vendor Support

▲ Comprehensive support, not from vendor

⊘ Very limited or no support

C = C++ (sometimes C), F = Fortran

* As of November 2022 (adapted from Herten 2022)

# Loop Grouping

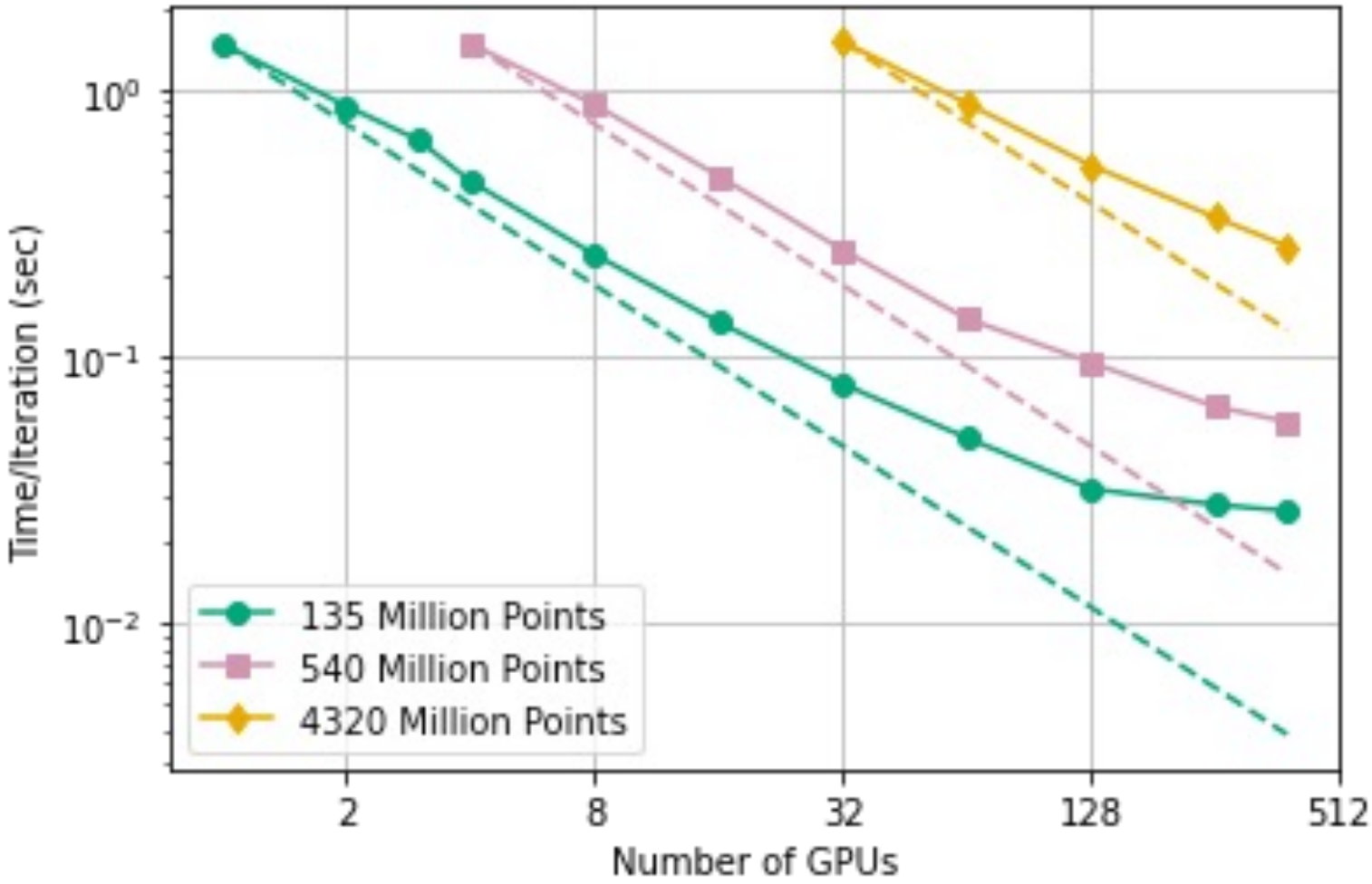
```
subroutine time_step()
    do ig = 1, ngrids
        call do_everything_on_a_grid()
    end do
end subroutine time_step

subroutine do_everything_on_a_grid()
!$omp loop
    do l = 1, ld
        ! Work on an l-plane
        call sub1(l,...)
        call sub2(l,...)
        call sub3(l,...)
        ...
    end do
!$omp loop
    do k = 1, kd
        ! Work on a k-plane
        call sub1_k(k,...)
        ...
    end do
end subroutine do_everything_on_a_grid

subroutine sub1(l,...)
    do k = 1, kd
    do j = 1, jd ! Vectorized loop
        ! Do only one thing
    end do
    end do
end subroutine sub1
```

```
subroutine time_step()
    do ig = 1, ngrids
        call do_everything_on_a_grid()
    end do
end subroutine time_step

subroutine do_everything_on_a_grid()
    ! Work on a grid
    call sub1()
    call sub2()
    call sub3()
    ...
end subroutine do_everything_on_a_grid

subroutine sub1()
!$acc parallel loop collapse(3) async
    do l = 1, ld
    do k = 1, kd
    do j = 1, jd
        ! Do only one thing
    end do
    end do
    end do
end subroutine sub1
```

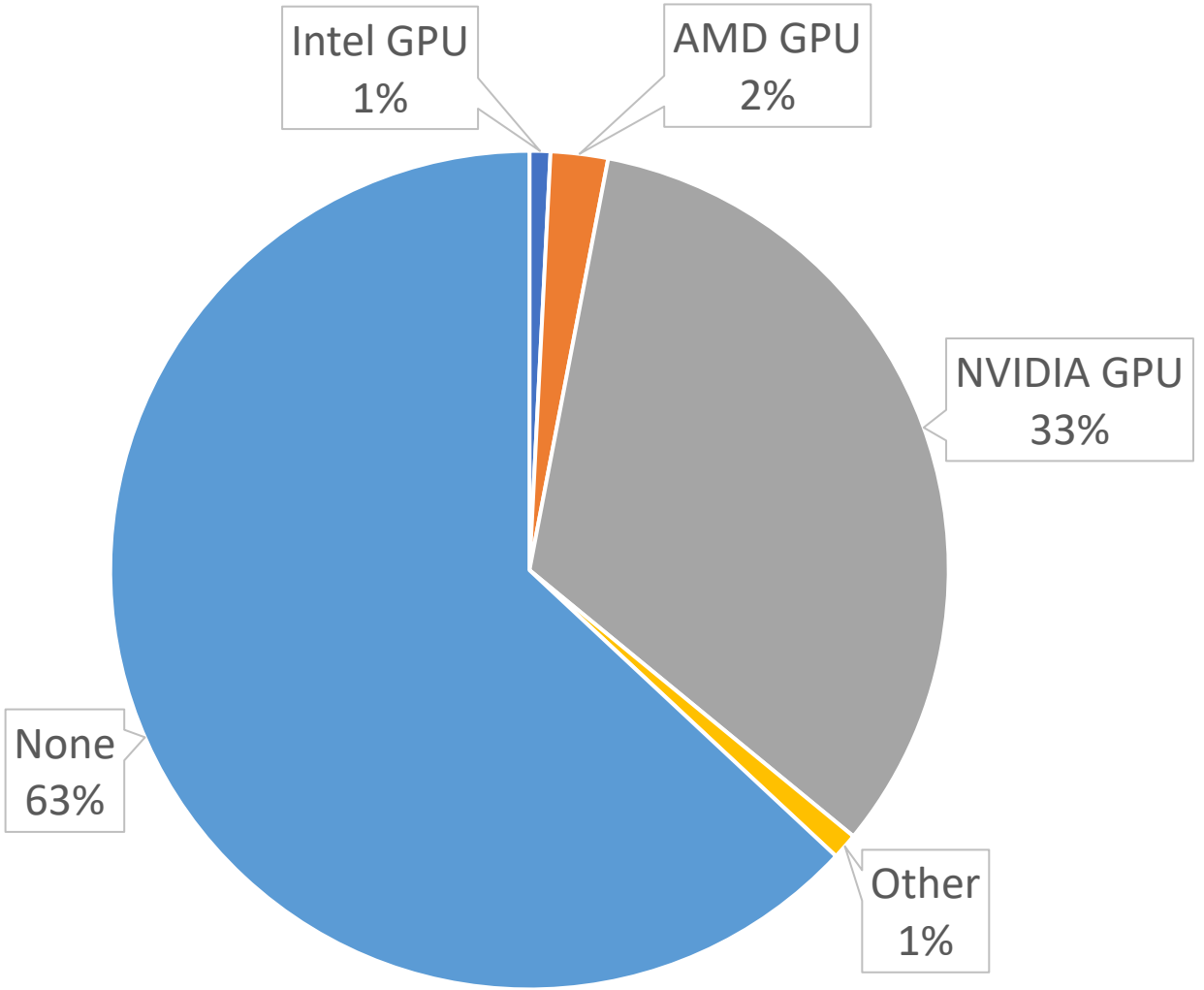
# Grid Batching

```
subroutine time_step()
    do ib = 1, nbatch
        call do_everything_on_a_batch(batch(ib),...)
    end do
end subroutine time_step

subroutine do_everything_on_a_batch(batch,...)
    ! Work on a batch of grids
    call sub1(batch,...)
    call sub2(batch,...)
    call sub3(batch,...)
    ...
end subroutine do_everything_on_a_batch

subroutine sub1(batch,...)
!$acc parallel loop gang collapse(2) async    &
!$acc    present(batch,grid)                  &
!$acc    private(ig)
    do iig = 1, batch%ngrids
    do l   = 1, batch%lmax
        ! Gang parallelism over grids in a batch and l-planes
        ig = batch%ig_map(iig)
        if( l <= grid(ig)%ld ) then

!$acc loop vector collapse(2)
            do k = 1, grid(ig)%kd
            do j = 1, grid(ig)%jd
                ! Vector parallelism within an l-plane
                ! Do only one thing
            end do
            end do

        end if
    end do
    end do
end subroutine sub1
```

# Strong Scaling



# Top 500
---
# Accelerator's Usage



# Top 500
---
# Accelerator's Performance