Preparing an Incompressible-Flow Fluid Dynamics Code for Exascale-Class Wind Energy Simulations

November 16, 2021

Paul Mullowney (National Renewable Energy Lab, NREL), Ruipeng Li (Lawrence Livermore National Lab, LLNL), Stephen Thomas (NREL), Shreyas Ananthan (Siemans Gamesa, NREL), Ashesh Sharma (NREL), Jon S. Rood (NREL), Alan B. Williams (Sandia National Labs, SNL), Michael A. Sprague (NREL)

SC21
St. Louis, MO | science & beyond.

Abstract

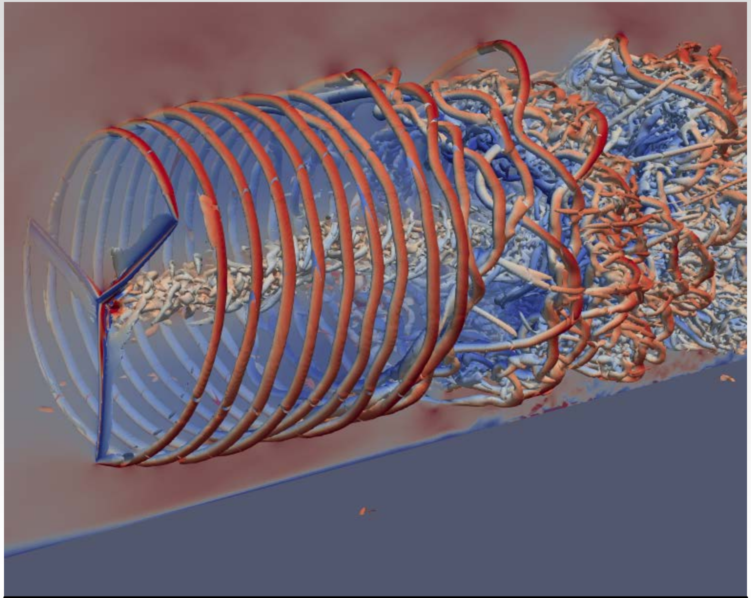
The U.S. Department of Energy has identified exascale-class wind farm simulation as critical to wind energy scientific discovery. A primary objective of the ExaWind project is to build high-performance, predictive computational fluid dynamics (CFD) tools that satisfy these modeling needs. GPU accelerators will serve as the computational thoroughbreds of next-generation, exascale-class supercomputers. Here, we report on our efforts in preparing the ExaWind unstructured mesh solver, Nalu-Wind, for exascale-class machines. For computing at this scale, a simple port of the incompressible-flow algorithms to GPUs is insufficient. To achieve high performance, one needs novel algorithms that are application aware, memory efficient, and optimized for the latest-generation GPU devices. The result of our efforts are unstructured-mesh simulations of wind turbines that can effectively leverage thousands of GPUs. In particular, we demonstrate a first-of-its-kind, incompressible-flow simulation using Algebraic Multigrid solvers that strong scales to more than 4000 GPUs on the Summit supercomputer.

Talk Outline

- Motivation
- Modeling strategy
- Linear solver innovations for GPUs
  - Assembly
  - AMG setup
  - Fast smoothers/preconditioners
- Computational results
  - Single-turbine, low-resolution performance
  - Single-turbine, high-resolution performance
  - Role of processor, compiler, and MPI-implementation on performance
- Emerging systems
- Looking forward

Motivation

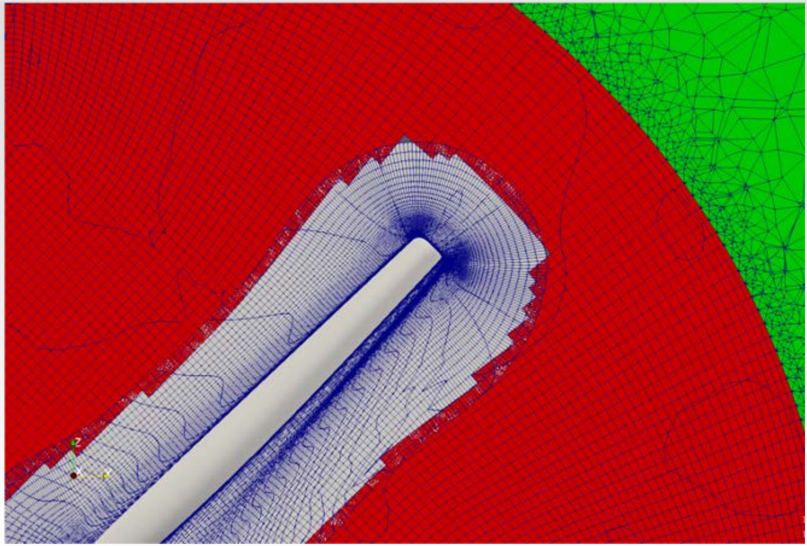
- ExaWind project goal is to build simulation software capability of modeling entire wind farms [1]
- Algorithms must be able to resolve
  - micron-scale boundary layers around turbine blades
  - kilometer-scale atmospheric boundary layers
- Software must handle blade-deformation and turbine motion in a complex environment including offshore
- Software **must** be high-performance in order to enable scientific exploration & engineering optimization
- ExaWind
  - Nalu-Wind : unstructured blade-resolved solver
  - AMR-Wind : structured background-solver
  - TIOGA : overset coupler between Nalu- and AMR-Wind



Flowfield (isosurfaces of Q-criterion colored by vorticity magnitude and a plane with vorticity-magnitude iso-contours) for the NREL 5-MW rotor with rigid blades operating in uniform inflow of 8 m/s.

Modeling Strategy

- Decoupled overset mesh methodology used to model moving turbine structures
- A Nalu-Wind mesh is a composition of multiple independent meshes that move with respect to one another
- Mesh motion (i.e. blade rotation around the rotor) requires continuous connectivity updates
- Benefit of decoupled overset approach:
  - Simple mesh creation process for wind farm simulations
  - Remove the need to reinitialize matrices at each time step
  - Enables a path to exascale through AMR-Wind/Nalu-Wind/TIOGA coupling for many turbines



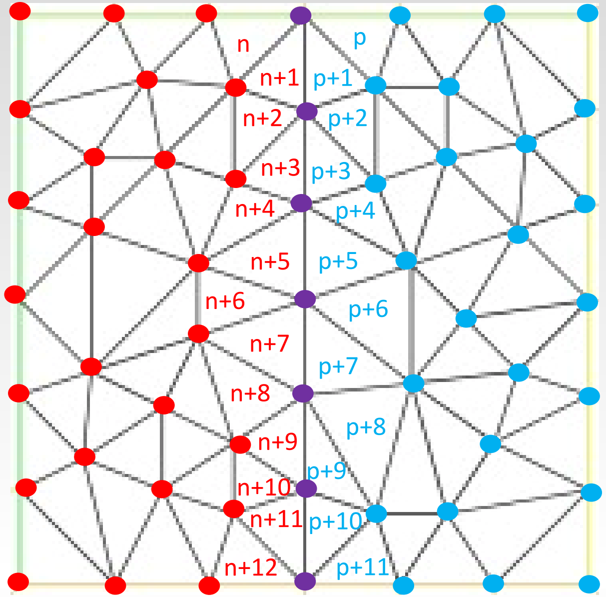
Example of the overset-meshing approach used in ExaWind. Shown are three overlapping Nalu-Wind meshes used to simulate the NREL Phase VI turbine

Nalu-Wind Software Stack

- STK (Sierra Toolkit) : handles the mesh data structures
- TIOGA : handles overset mesh capabilities
- Kokkos : Portable, parallel execution constructs
- Linear System Solvers :
  - Hypre : Boomer AMG, CUDA/HIP backends
  - Trilinos : Muelu, Tpetra, Kokkos
- Zoltan2 : Domain decomposition with ParMETIS, Scotch, RCB algorithms
- NetCDF/HDF5 : IO

Nalu-Wind Assembly

- Decoupled overset enables computation of the exact sparsity pattern for the global matrix for the entire simulation
- Each rank has an owned part, i.e. contributions to the matrix rows/rhs values on this rank
- Each rank might have a shared part, i.e. contributions to the matrix/rhs values on other ranks
- matrix/rhs contributions from mesh elements of same type (i.e. tetrahedron) are computed via atomics in a single Kokkos kernel



- Owned on rank 1
- Owned on rank 2
- Owned on rank 1, Shared on rank 2

Hypre Assembly

- Hypre Assembly API receives coordinate (coordinate) matrix with buffers of size

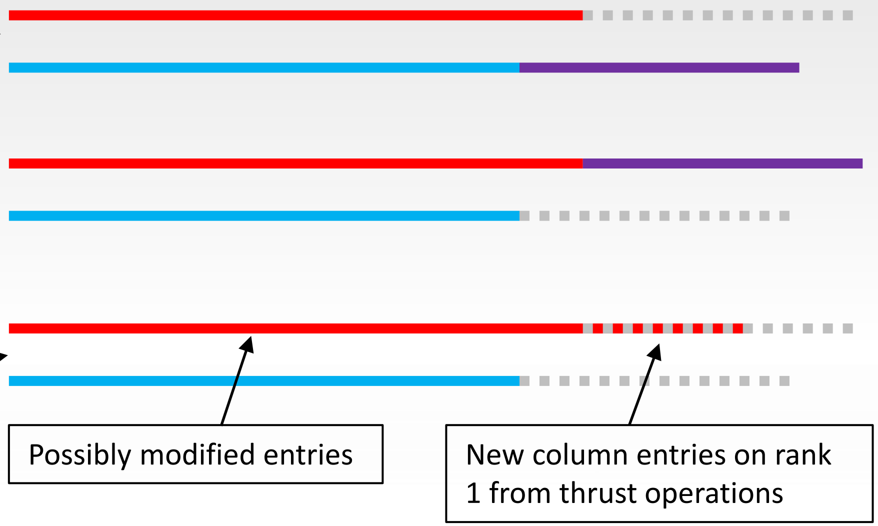
$$nnz_{local} = nnz_{owned} + \max(nnz_{shared}, nnz_{recv})$$

  Before assembly (MPI Messaging), data are stacked with owned part followed by shared

- After MPI Messaging, shared elements are overwritten by the values received from other ranks

- *thrust::stable_sort_by_key* and *thrust::reduce_by_key* are used to complete the global matrix assembly

Matrix values memory schematic. Similar data structures for row and column indices
- Top line: rank 1
- Bottom line: rank 2
- Dotted line: space is allocated but not used

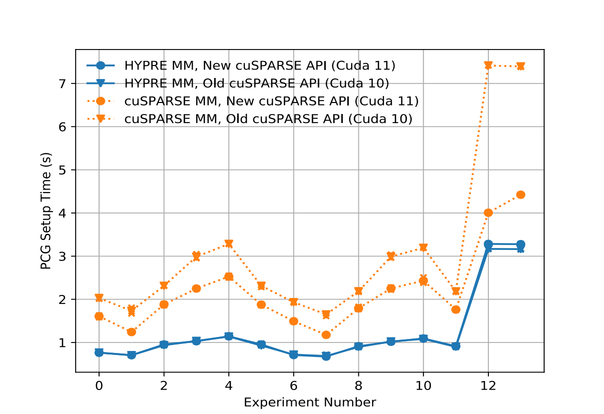


BoomerAMG setup on GPUs

- Coarsening algorithm: PMIS
  - massively parallel algorithm to find maximal independent set
  - uses cuRAND to generate random numbers on GPUs
- Aggressive coarsening to reduce the grid and operator complexity
  - corresponding two-stage interpolation
- Interpolation algorithms: direct interpolation and matrix-matrix based extended interp.
  - Bootstrap AMG (BAMG) direct interpolation by solving a local optimization problem
  - Distance-2 interp. in the form of mat-mat for better portability

$$-\left[\left(D_{FF}+D_{\gamma}\right)^{-1}\left(A_{FF}^{s}+D_{\beta}\right)\right]\left[D_{\beta}^{-1}A_{FC}^{s}\right]$$

  - More variants M-M ext+i/ ext+e
- Galerkin product RAP: use *hypre*'s SpGEMM kernel
  - Better performance than cuSPARSE



Boomer AMG Smoothers optimized for GPUs

- GMRES Krylov solver for momentum and pressure continuity
- Neumann Gauss-Seidel preconditioner and AMG smoother for pressure
- Based on the iteration for $Ax = b,\ \ A = D + L + U,\ r_k = b\ - Ax_k$

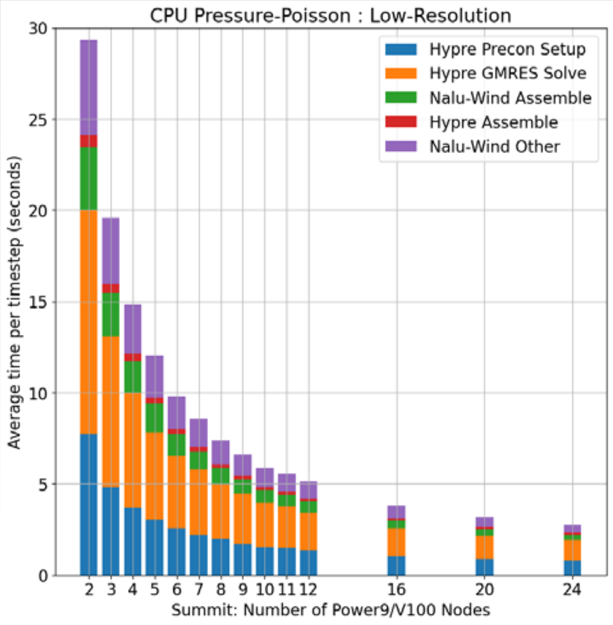
$$x_{k+1} = x_k + \sum_{j=0}^{n} (-DL)^j\ D^{-1}\, r_k$$

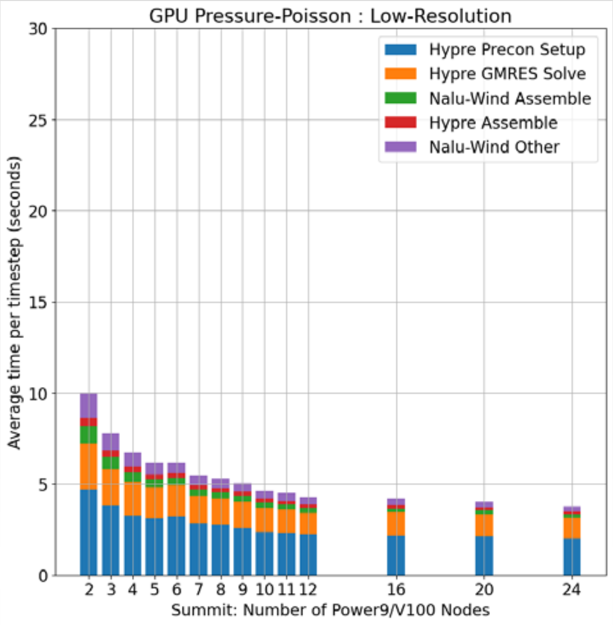
  - Exploits sparse matrix-vector products (SpMV)
  - SpMV are 25 to 50 times faster than direct triangular solver $Lx = b$ on GPU
  - Iterate for $k = 1, 2$
  - New smoother option in Hypre-BoomerAMG from LLNL

Computational Studies

- Results for 2 Turbines: unstructured, unbalanced
  - Low-Resolution, Single-Turbine: 23 million mesh nodes
  - High-Resolution, Single-Turbine: 635 million mesh nodes
- Simulation parameters
  - 50 times steps
  - 4 Picard iterations per time step. Each Picard iteration has
    - 1 pressure-Poisson solve, 3 momentum solves (decoupled), 2 scalar transport solves (TKE and SDR)
  - Solver residual tolerances set to 1.e-5
- All systems solved with Hypre
- Key Measurements
  - Per equation performance : average time spent solving each equation system
    - pressure-Poisson : Boomer AMG with 2 stage GS preconditioner
    - momentum and scalar transport : 2 stage GS preconditioner
  - Application-level performance : average time per time step

Pressure-Poisson/Hypre Boomer AMG



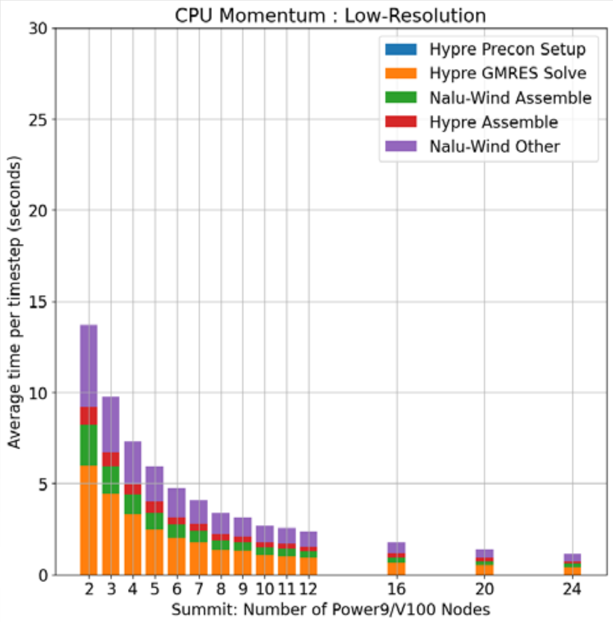


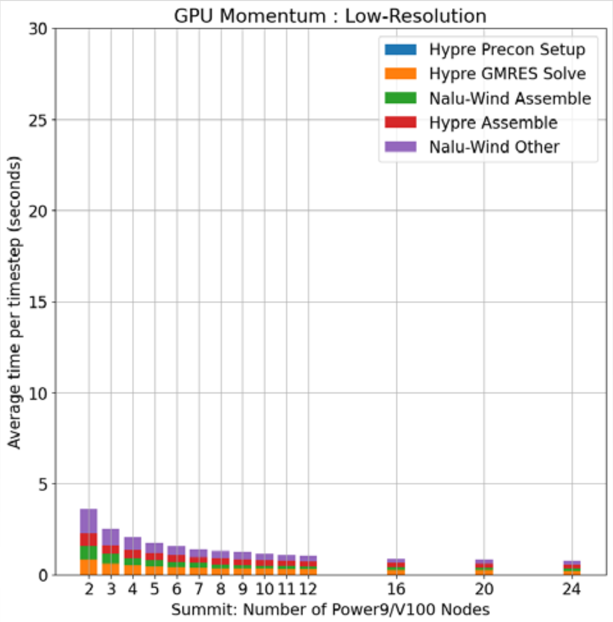
GPU implementation of Boomer AMG is substantially faster when there is *significant* work per device

EXASCALE COMPUTING PROJECT

NREL Sandia National Laboratories Lawrence Livermore National Laboratory

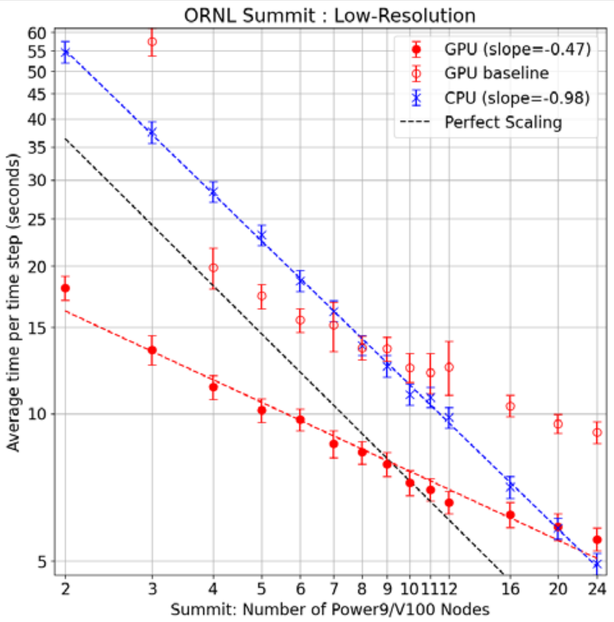
Momentum

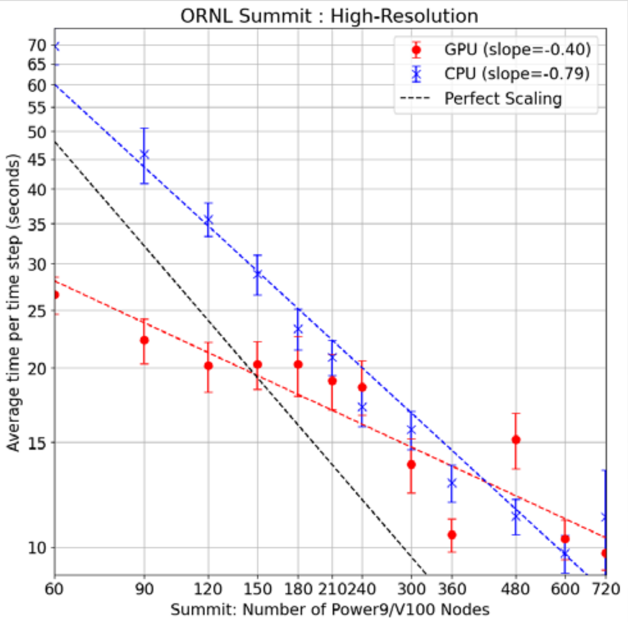




GPU-accelerated Krylov solves with simple, but effective preconditioners are competitive with CPU implementations down to O(10^5) unknowns per device

Low- vs High-Resolution Strong Scaling





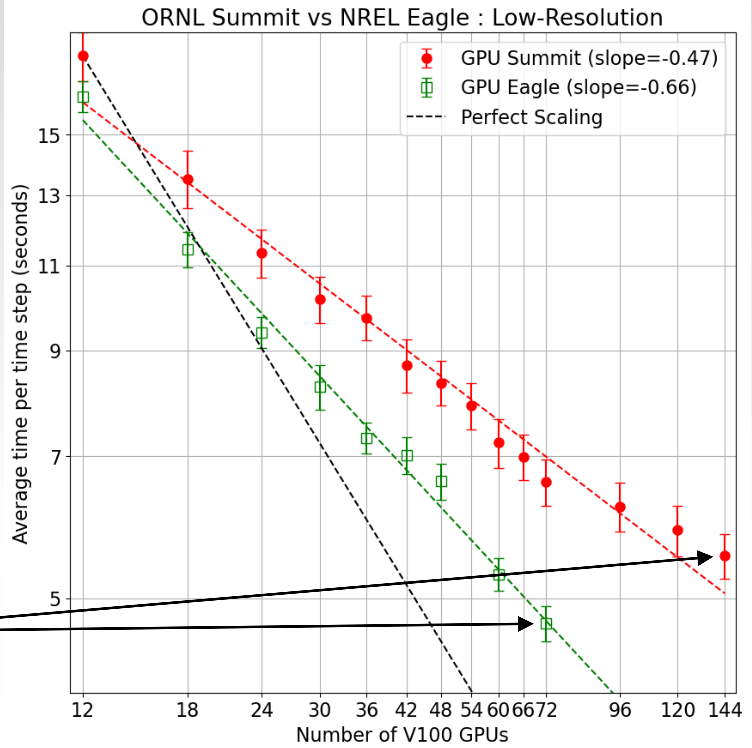
Application performance is good compared to CPUs when there is *significant* work per device though the scaling is degraded

EXASCALE COMPUTING PROJECT

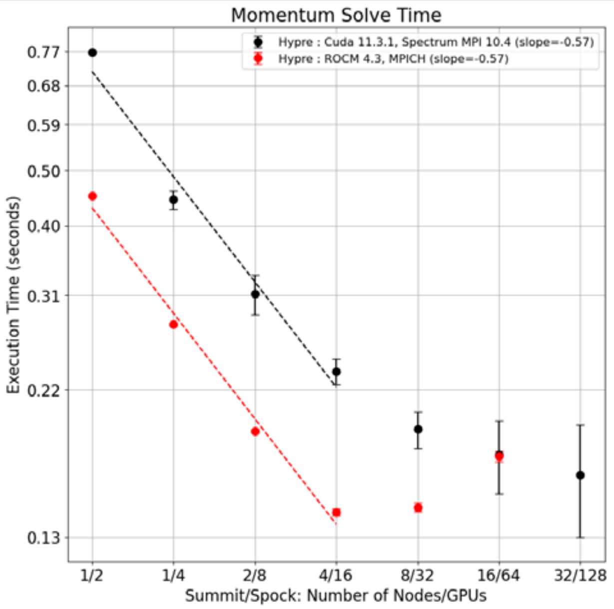
NREL | Sandia National Laboratories | Lawrence Livermore National Laboratory

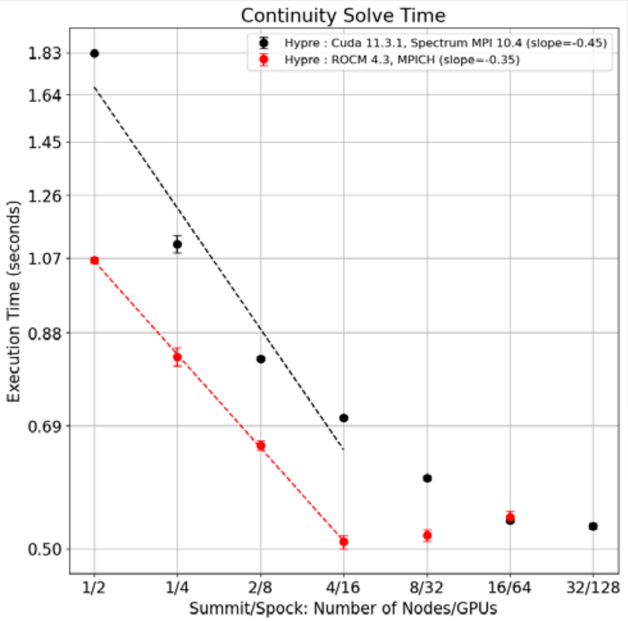
System Dependence

- NREL Eagle node configuration
  - 36 cores/Intel(R) Xeon(R) Gold 6150 CPU @ 2.70GHz
  - GCC 8.4.0
  - 2 NVIDIA V100 PCIE per node
  - MPI : HPE MPT
- ORNL Summit node configuration
  - 42 cores/POWER 9 @ 3.8 GHz
  - GCC 7.4.0
  - 6 NVIDIA V100 SXM2
  - MPI : Spectrum 10.3
- All GPU resources per node utilized
- Eagle is 15% faster with half the number of GPUs. Most of the gain is achieved in pressure-Poisson preconditioner setup and solve



Emerging Architectures, Krylov Solve performance, Summit vs Spock





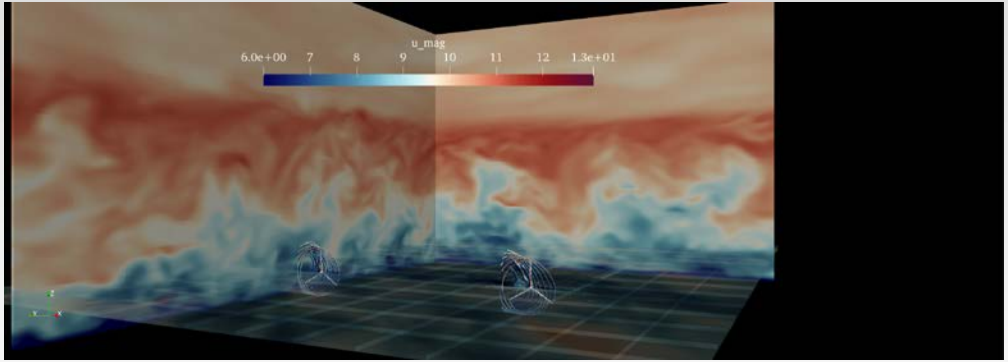
M100 GPUs show substantial gains in performance over V100s.
This bodes well for the ExaWind software stack on Frontier!

EXASCALE COMPUTING PROJECT

NREL Sandia National Laboratories Lawrence Livermore National Laboratory

Looking Forward

- In order to simulate entire Wind Farms, the ExaWind team is adopting a hybrid solver technique
  - Nalu-Wind around turbines
  - AMR-Wind everywhere else
  - TIOGA couples ALL the meshes
- Initial GPU version of the hybrid-solver nearly working
  - Loose coupling via TIOGA allows us to easily partition ALL compute resources (CPU/GPU)



Flowfield (isosurfaces and planes of Q-criterion colored by velocity magnitude) for two turbines operating in an ABL (atmospheric boundary layer). Turbines reside in the x-plane and are subject to inflow from the x and y directions.

ECP EXASCALE COMPUTING PROJECT

NREL Sandia National Laboratories Lawrence Livermore National Laboratory

Conclusions

- Substantial effort is required to enable Physics applications on unstructured meshes, such as CFD, to run well on Petascale-Level resources
- Weak scaling is hard to do well on unstructured models
  - 2x performance drop between low and high-resolution models (27.5x bigger)
- Performance can vary significantly between HPC systems, even when the underlying GPU accelerators are equivalent
- ExaWind software stack has a path to exascale that does not require global solves on unstructured meshes
  - ExaWind hybrid-solver will have global solves on AMR meshes
  - For comparable applications that don't have this path, the statements above are worth considering in the context of those applications

Acknowledgements

The National Renewable Energy Laboratory (NREL) is operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Lawrence Livermore National Laboratory operates under DOE Contract DE-AC52-07NA27344 (LLNL-PROC-821116). Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the DOE's National Nuclear Security Administration (NNSA) under contract DE-NA0003525. This report followed the Sandia National Laboratories formal review and approval process (SAND2021-12703 C). As such, the technical report is suitable for unlimited release. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the DOE Office of Science (SC) and the NNSA, and was performed using computational resources of the Oak Ridge Leadership Computing Facility, which is a DOE SC User Facility supported under Contract DE-AC05-00OR22725, and computational resources sponsored by the DOE's Office of Energy Efficiency and Renewable Energy and located at NREL. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.