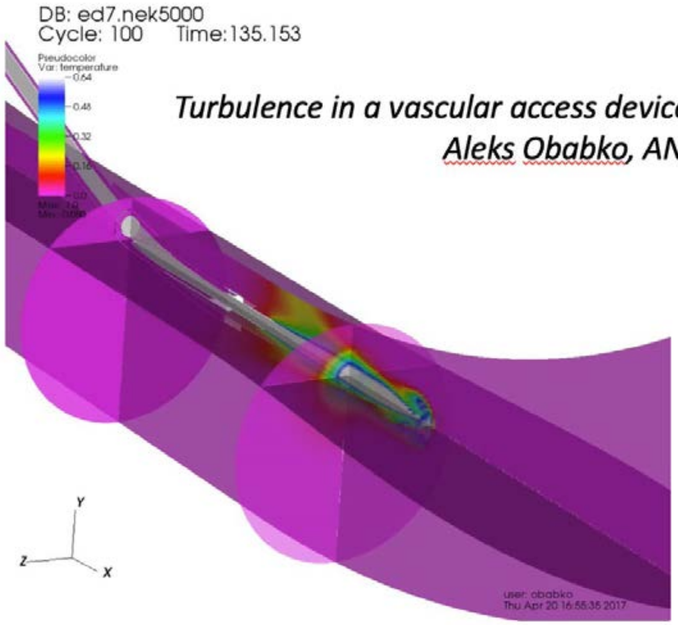
Exascale: A User’s Perspective

Paul Fischer **University of Illinois Urbana-Champaign** *Computer Science; Mechanical Science & Engineering*
Argonne National Laboratory *Mathematics and Computer Science*

Stefan Kerkemeier – K2 / ANL
Yu-Hsiang Lan – ANL
James Lottes– ANL (Google)
Elia Merzari – Penn State/ANL
Misun Min – ANL
Aleks Obabko – ANL
Malachi Phillips – UIUC
Thilina Rathnayake – UIUC
Ananias Tomboulides – ATU/ANL
Tim Warburton – Virginia Tech



Turbulence in a vascular access device.
Aleks Obabko, ANL

Supported by DOE Applied Math Research and, formerly, by the Center for Efficient Exascale Discretizations (CEED) under the DOE Exascale Computing Project.

Example: Compressed Turbulence *(Nek5000-CPU)*

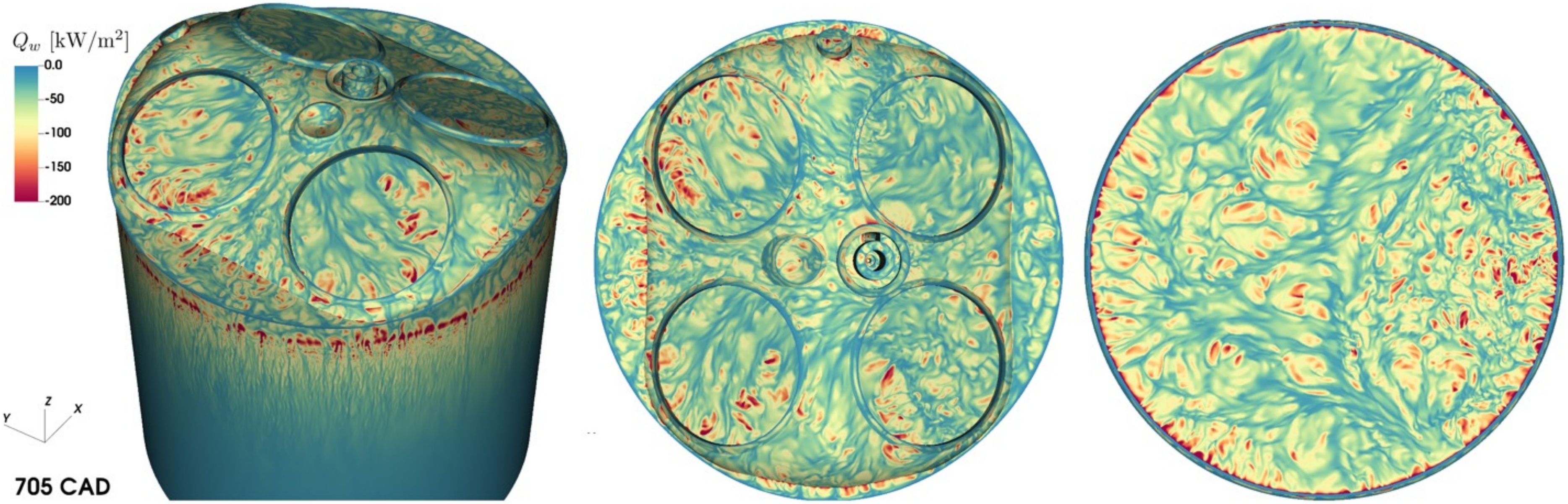
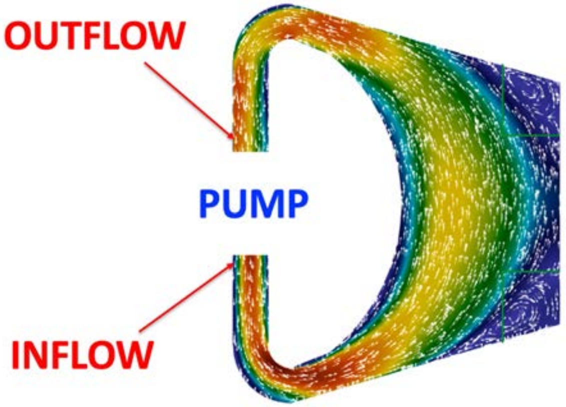
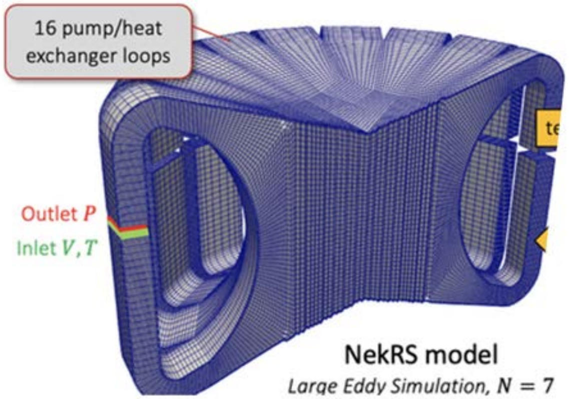


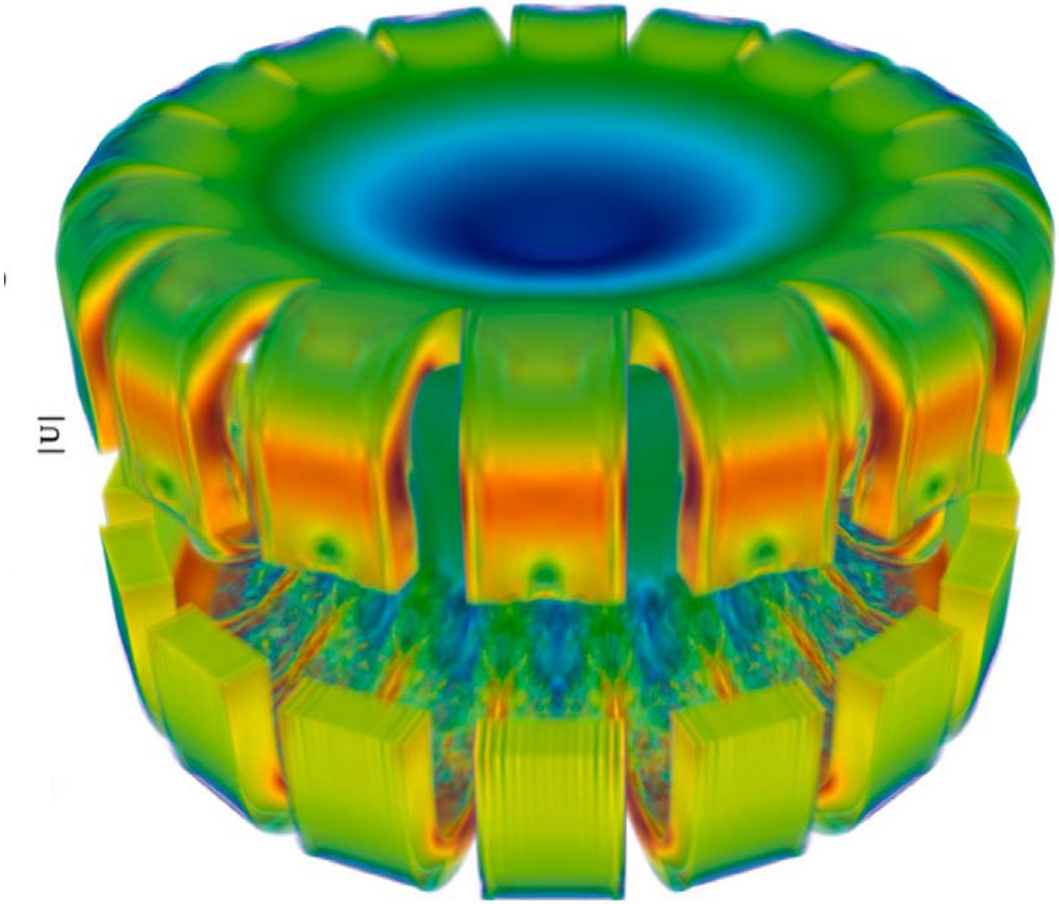
FIGURE 2.9 DNS of compression in an optical engine. Iso-contours of heat flux along the cylinder walls at 15^o bTDC, left-to-right: bird's eye view, cylinder head, piston.

G. Giannakopoulos, K.Keskinen, J.Kochand, M.Bolla, C.E.Frouzakis, Y.M. Wright, K. Boulouchos, M. Schmidt, B. Böhm and A. Dreizler, Characterizing the evolution of boundary layers in IC engines by combined laser-optical diagnostics, direct numerical and large-eddy simulations, *Flow, Turbulence and Combustion.*

NekRS Example: Molten Salt Reactor

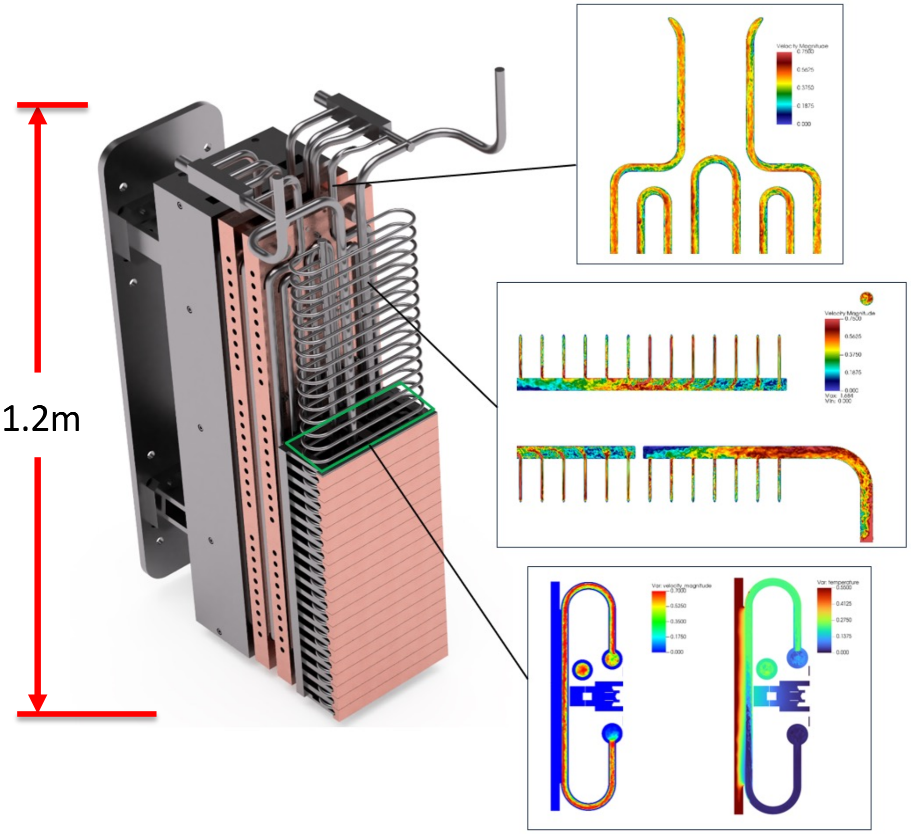


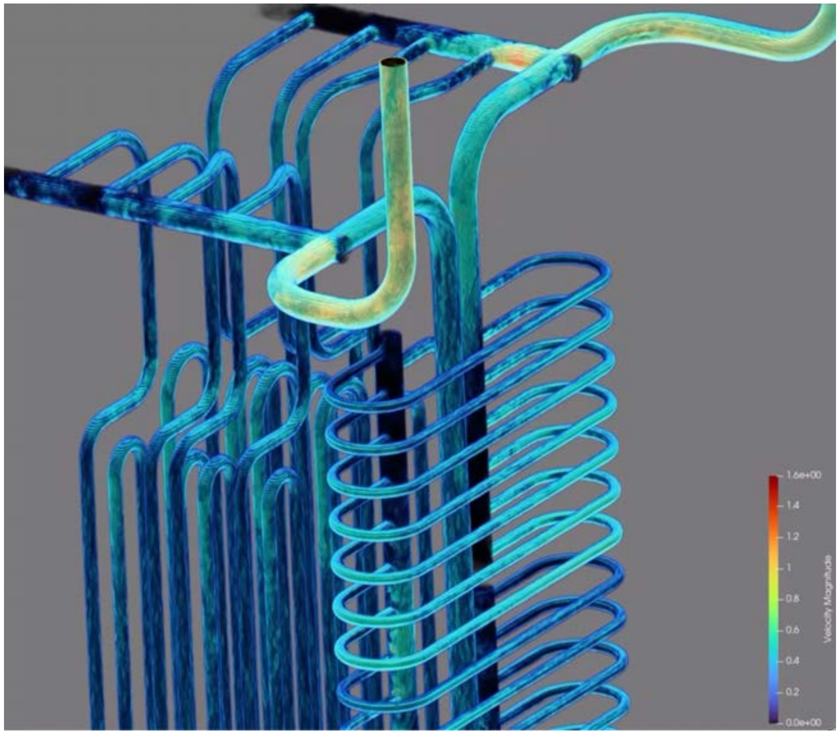




J. Fang, M. Tano, N. Saini, A. Tomboulides, V. Coppo-Leite, E. Merzari, B. Feng, D. Shaver, *Nucl. Eng. & Des*., 424, CFD simulations of Molten Salt Fast Reactor core cavity flows, 2024

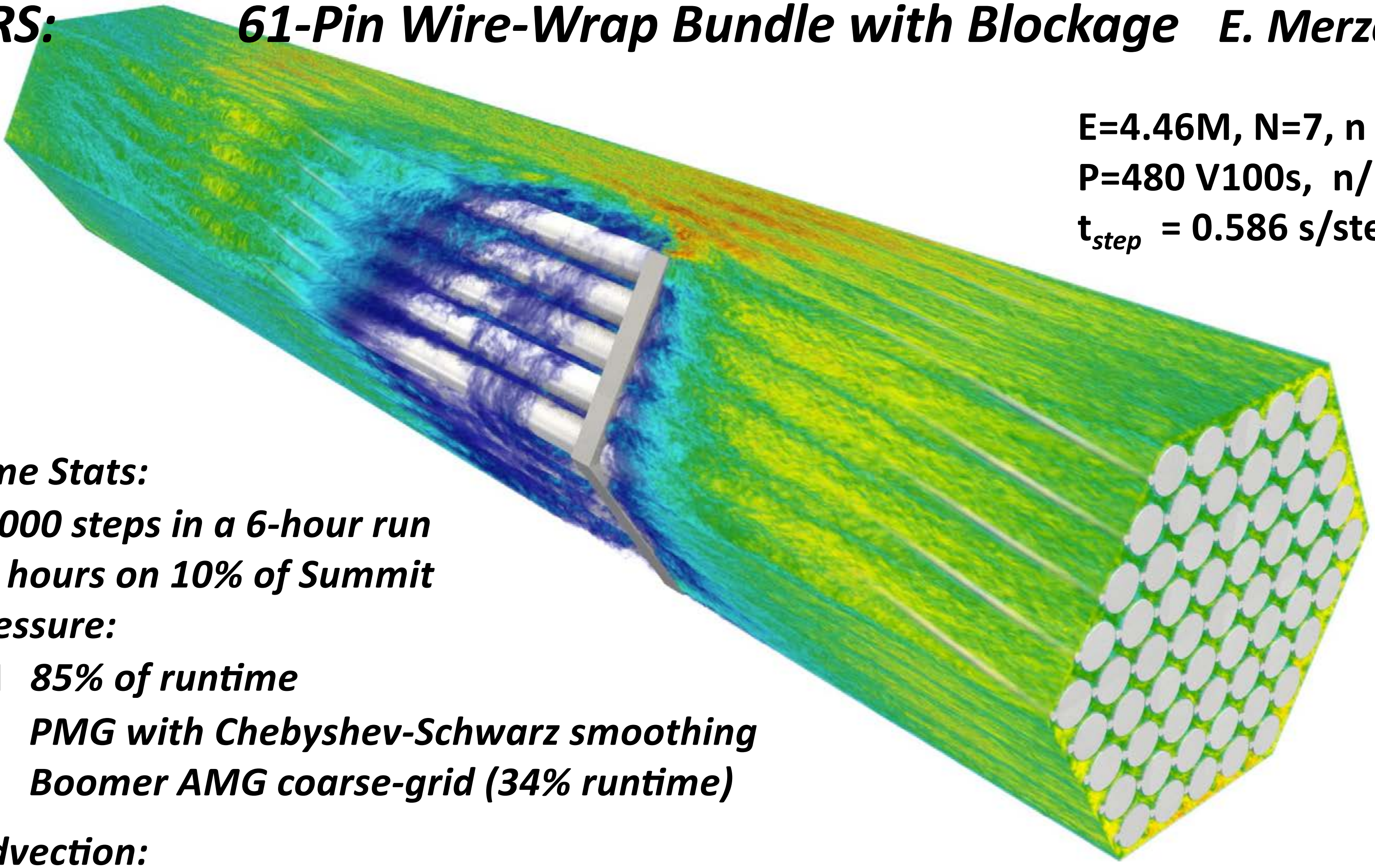
Example: UKAEA *Chimera* Experimental Fusion Blanket/Diverter Model





- ❑ Full conjugate heat transfer - up to 1 trillion points on Frontier, 1.6 B spectral elements.
- ❑ MHD is planned for future simulations / experiments.

NekRS: 61-Pin Wire-Wrap Bundle with Blockage *E. Merzari, PSU*



E=4.46M, N=7, n = 1.55B

P=480 V100s, n/P = 3.24M

t_{step} = 0.586 s/step

Runtime Stats:

- ❑ ***36000 steps in a 6-hour run***
- ❑ ***60 hours on 10% of Summit***
- ❑ ***Pressure:***
  - ❑ ***85% of runtime***
  - ❑ ***PMG with Chebyshev-Schwarz smoothing***
  - ❑ ***Boomer AMG coarse-grid (34% runtime)***
- ❑ ***Advection:***
  - ❑ ***2nd-order characteristics: CFL=1.5 (10% runtime)***

ILLINOIS University of Illinois at Urbana-Champaign · CEED Exascale Discretizations · Argonne National Laboratory

Exascale Challenges - Scalability

Q: Why P=480 V100s ?

Why not 4000?

Why not 27648?

A: You no longer get [significant] *"speed-up" beyond P=480*

Q: Why not?

Did we do something wrong?

How would you know?

The essence of this talk…

High-Performance Computing and Computational Fluid Dynamics

❑ How do we make CFD fast for Users?

❑ Why do we / they care?

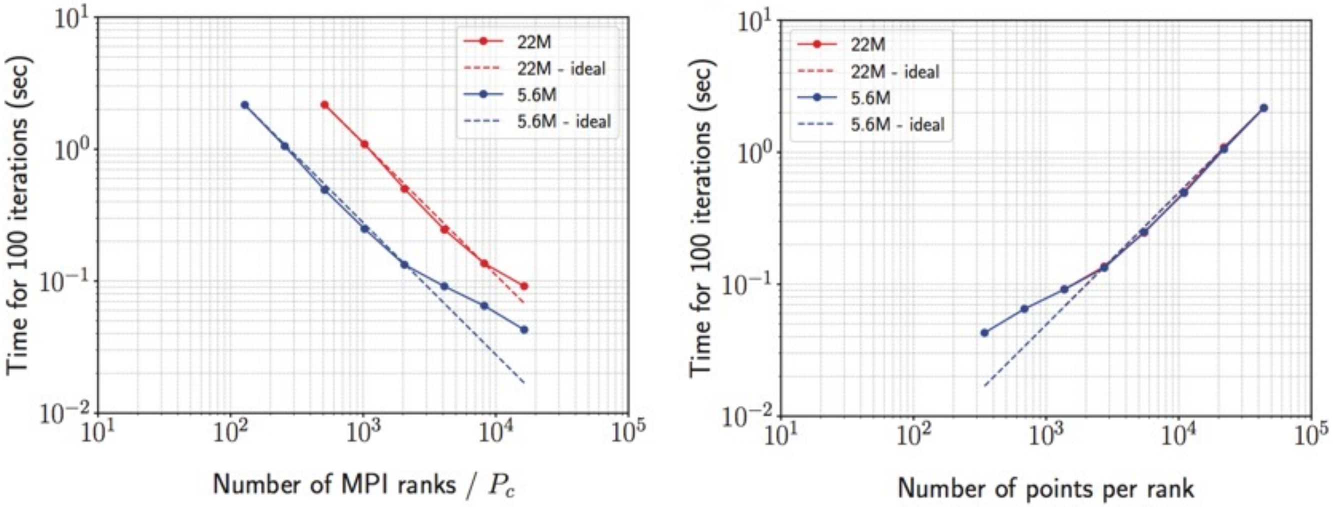
ECP-NASA Meeting: NASA needs

- ***1000s of CPUs***
- ***Weeks → Months of runtime***
- ***Need larger P (or GPUs?)***

❑ What is the role of HPC in this context?

Parallelism: Stong-Scaling, Time to Solution, and Energy Consumption

Fixed Problem Size (n) with Increasing Number of Processors (or Processes) - Mira



Observations:

1. Time-to-solution goes down with increasing P, particularly for η = 1.
2. For η = 1, energy consumption ~ P x t_{sol} = *constant* — no penalty for increased P.
3. The red curve can use more processors than the blue. ***WHY?***
4. Why (for a problem of any size), do we find η < 1?
   - What is the root cause of the fall-off, ***and can we do something about it??***

How Do We Make CFD Fast for Users?

❑ Improved numerical discretizations:

- For a given accuracy, reduce the problem size

  n = number of grid points

  without compromising performance,

  time-to-solution per grid point

❑ High "on-core" or "on-GPU" performance

- make the code as fast as possible on each GPU

❑ Enable the use of "as many GPUs as possible"

- How many is that? And why?

OUTLINE

- ❑ ***Improved Discretizations***
- ❑ ***High-performance on each GPU***
- ❑ ***As many GPUs as possible***

Incompressible Navier-Stokes Equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \;=\; -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u}$$

$$\nabla \cdot \mathbf{u} \;=\; 0$$

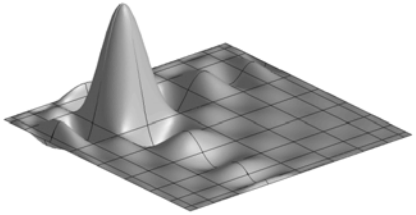
- Key algorithmic / architectural issues:
  - Unsteady evolution implies many timesteps, significant reuse of preconditioners, data partitioning, etc.
  - div **u** = 0 implies ***long-range global coupling at each timestep***
    → communication intensive iterative solvers
  - Small dissipation → large number of scales, large number of timesteps
    → large number of grid points for high Reynolds number, Re

Expensive per step, because of tight coupling.

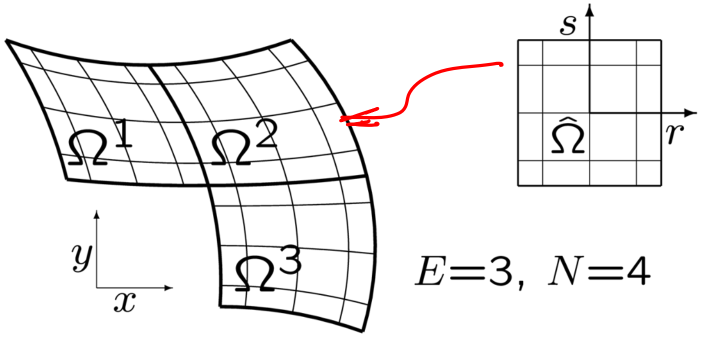
Expensive, because huge range of scales.

Spectral Element Method *A. Patera ‘84*

- Variational method, similar to FEM, using *GL* quadrature.
- Domain partitioned into $\boldsymbol{E}$ high-order hexahedral elements
- Trial and test functions represented as $\boldsymbol{N}$th-order tensor-product polynomials within each element. ($\boldsymbol{N}$ ~ 4 - 15, typ.)
- Qualitatively different from p-type FEM
  - $\boldsymbol{n \sim EN^3}$ grid points in 3D
  - Fast operator evaluation: ***O(n)*** storage, ***O(nN)*** work
- Converges *exponentially fast* with $\boldsymbol{N}$ for smooth solutions.



2D basis function, N=10



$E=3,\ N=4$

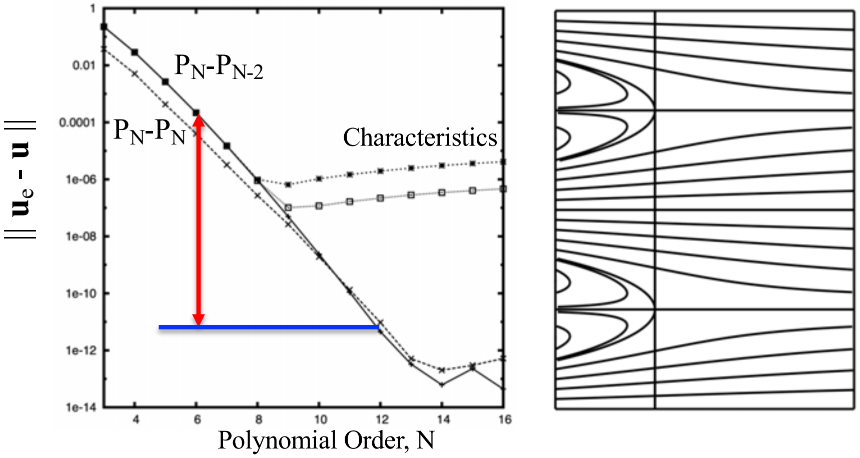
Spectral Element Convergence: Exponential with N

- ❑ ***8 orders-of-magnitude*** *error reduction when doubling the resolution in each direction*

  (smooth solutions…)

- ❑ *For a given error,*
  - ❑ *Reduced number of gridpoints*
  - ❑ *Reduced memory footprint.*
  - ❑ *Reduced data movement.*

Exact Navier-Stokes Solution (Kovazsnay '48)



$$
\begin{aligned}
v_x &= 1 - e^{\lambda x}\cos 2\pi y \\
v_y &= \frac{\lambda}{2\pi} e^{\lambda x}\sin 2\pi y \\
\lambda &:= \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}
\end{aligned}
$$

Influence of Scaling on Discretization Kreiss & Oliger, '72

Large problem sizes enabled by peta- and exascale computers allow propagation of small features (size λ) over distances $L \gg 1$.

If speed ≈ 1, then $\tau_{\text{final}} \approx L/\lambda$, τ=time to move τ wavelengths.

- Dispersion errors accumulate linearly with time:

$$\begin{aligned}\epsilon = \textbf{error} &\sim |\textbf{correct speed - numerical speed}| \times \tau \\ &= |\textbf{numerical dispersion error}| \times \tau,\end{aligned}$$

  for each wavenumber, k.

- Final error:

$$\epsilon_f = \textbf{error}(\tau_{\text{final}}) : \ \epsilon_f \sim (L/\lambda) * |\textbf{numerical dispersion error}|$$

- For fixed final error, ϵ_f, require:

$$\textbf{numerical dispersion error} \sim (\lambda/L)\, \epsilon_f \ll 1.$$

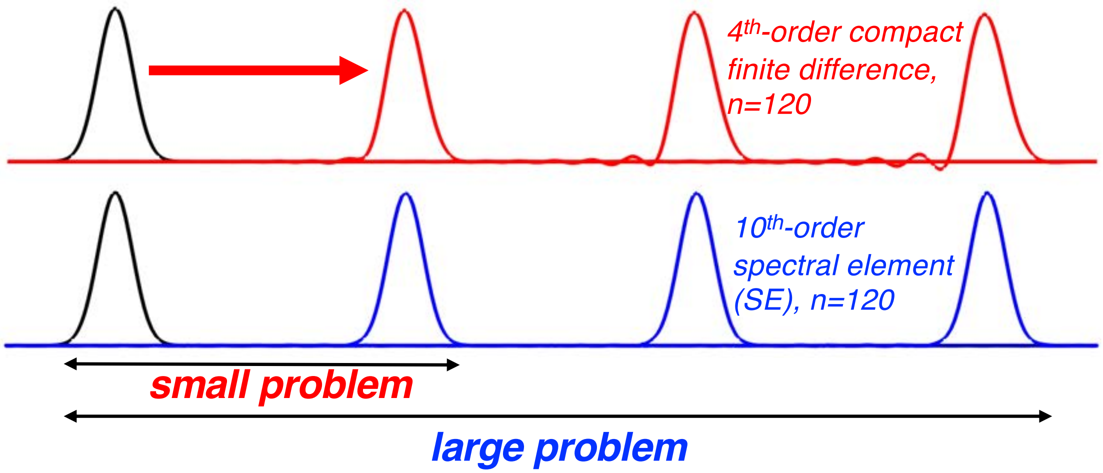
← *Requires accurate spatial discretization.*

High-Order efficiently delivers accuracy.

H .O. Kreiss, J. Oliger, Comparison of accurate methods for the integration of hyperbolic problems, *Tellus* **24**, 199–215, 1972.

Mathematical Foundation: High-Order Approximations

- ❑ For fixed cost (number of grid points, ***n)***, high-order methods ***minimize dispersion errors*** in advection-dominated transport (e.g., turbulent flow, heat transfer, electromagnetics)



- ❑ For large (e.g., exascale) problems, dispersion error accumulates

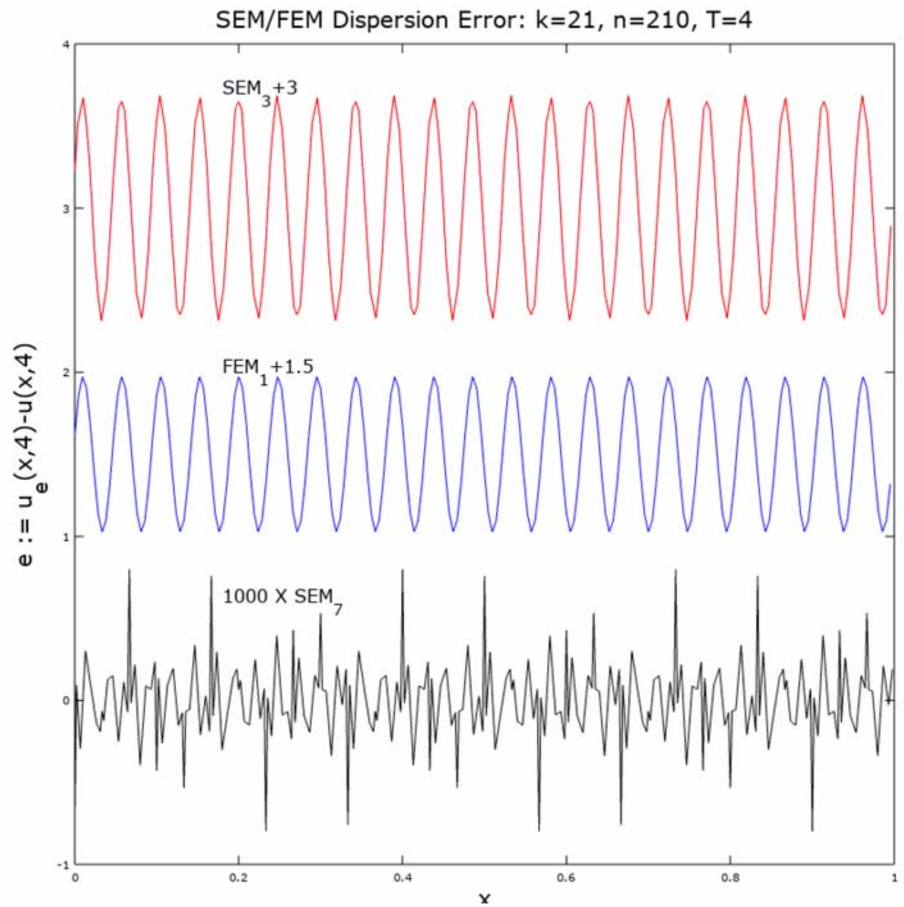
  → ***high spatial accuracy becomes important!***

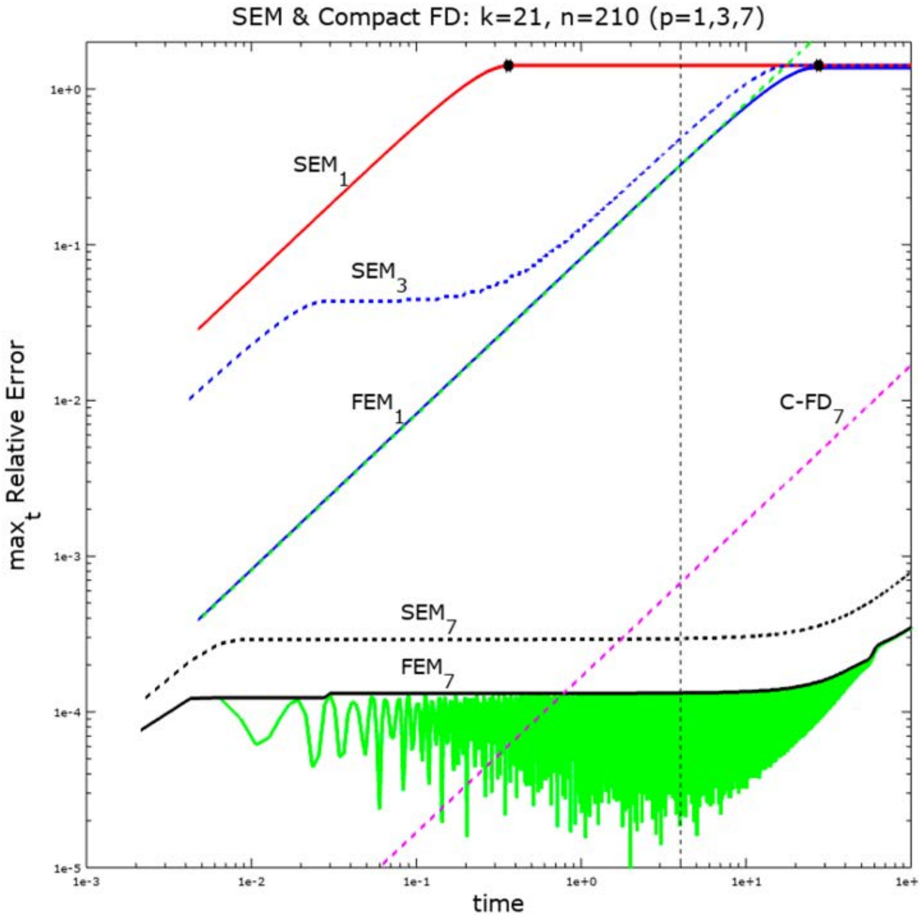
- ❑ Our goal is to make high-order methods *fast, stable, and robust*

- ❑ We are the *leaders in the development of high-order meths* for science/engineering applications

Mathematical Foundation: High-Order Approximations

- ❑ Error behavior for 1D model problem



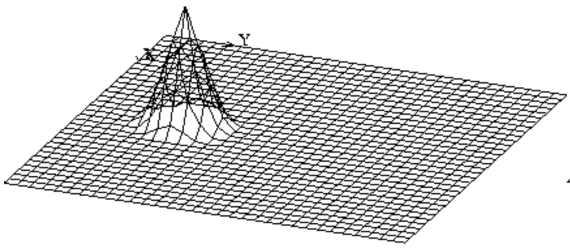


Excellent Transport Properties, even for Nonsmooth Solutions

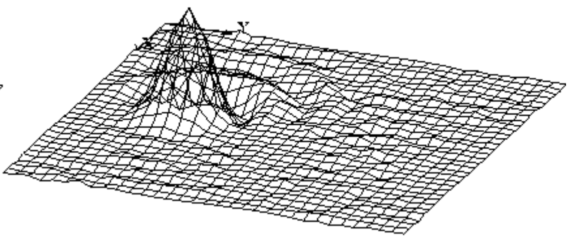
- ❑ *Convection of nonsmooth data on a 32x32 grid*
  - ❑ *E_1 x E_1 spectral elements of order N*

- ❑ *As noted in Gottlieb & Orszag ('77), spectral methods work remarkably well, even for nonsmooth solutions.*

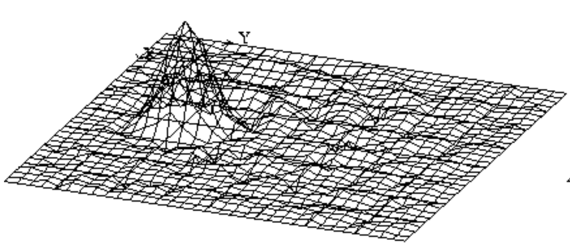
- ❑ *One way to think about this is that the **resolved modes** are propagated very accurately.*



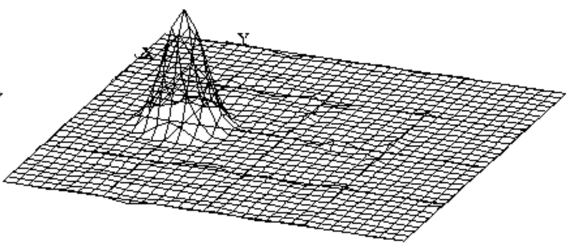
Initial Condition



E_1 = 16, N=2



E_1 = 8, N=4



E_1 = 4, N=8

(cf. Gottlieb & Orszag, Spectral Methods, 1977)

What About Turbulence?

- So far, we've demonstrated a high-order advantage only for *linear problems*.
  - ***What about highly nonlinear problems, e.g., turbulence?***
  - ***Is control of numerical dispersion important in the presence of significant physical dispersion?***

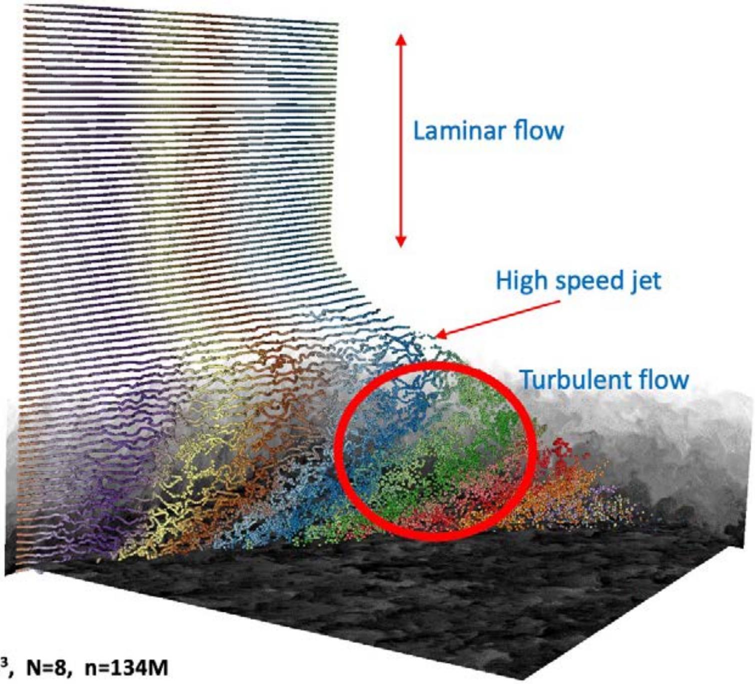
- To address this question, we look at a highly turbulent case that is significantly under-resolved - with the lack of resolution address through sophisticated turbulence models.
  - We compare with a contemporary 2nd-order finite volume code

Under-Resolved Example: GABLS Atmospheric Boundary Layer Benchmark

M. Min, A. Tomboudlies, P. Fischer (ANL), M. Brazell, M. Churchfield, M. Sprague (NREL)

Re = 50 M, *WMLES*

400m x 400m x 400m, Background flow 8 m/s

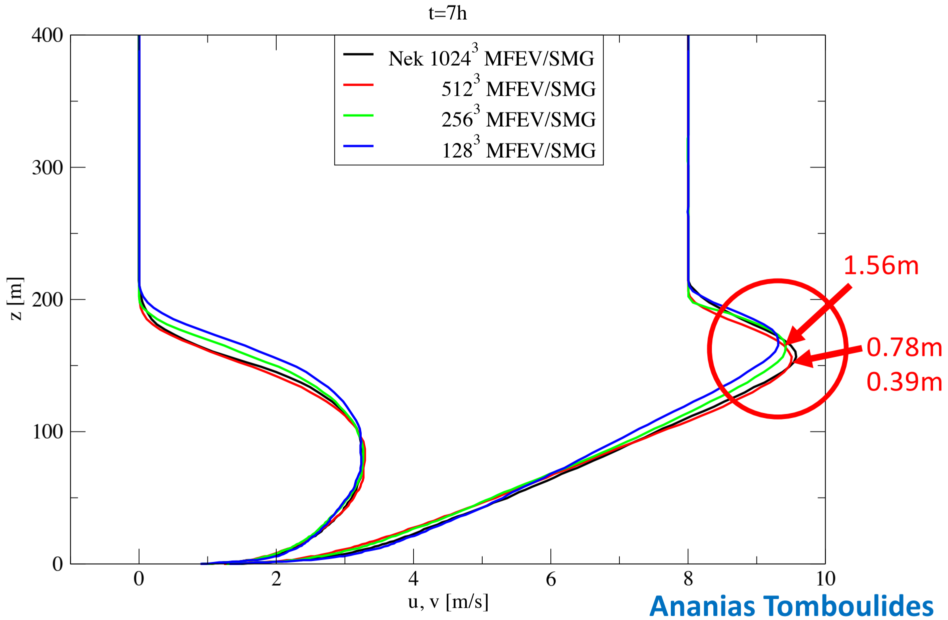


E=64^3, N=8, n=134M

NekRS simulation using 60 GPUs/V100s on Summit

Tracer particles: N. Lindquist (UTK), M. Min (ANL)

- ❑ Flow analysis with spatial and time-averaged velocity
- ❑ 0.78m resolution already converges to 0.39m case
- ❑ 1.56m resolution is quite close to 0.78m case



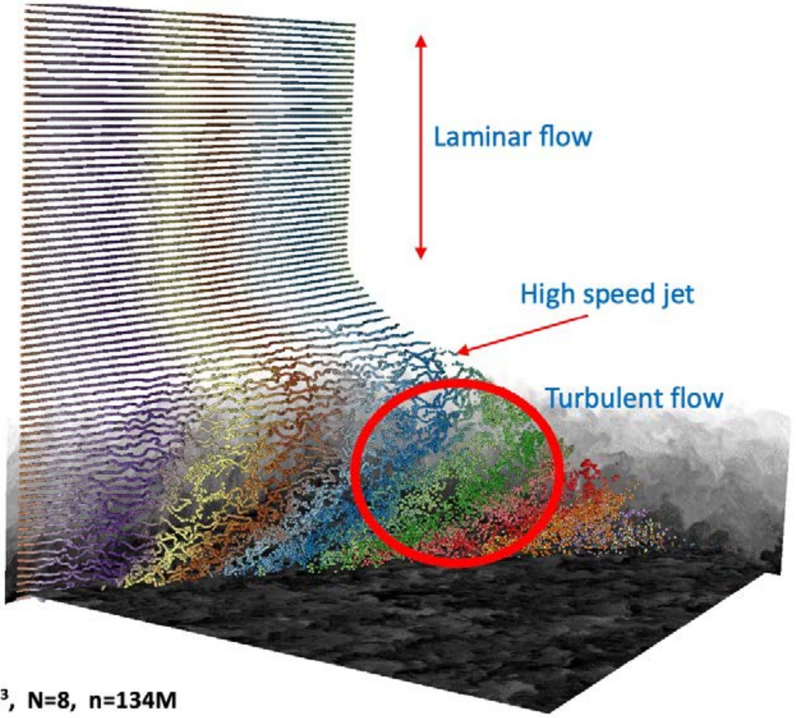
Ananias Tomboulides (AUTH)

Under-Resolved Example: GABLS Atmospheric Boundary Layer Benchmark

M. Min, A. Tomboudlies, P. Fischer (ANL), M. Brazell, M. Churchfield, M. Sprague (NREL)

Re = 50 M, WMLES

400m x 400m x 400m, Background flow 8 m/s



E=64^3, N=8, n=134M

NekRS simulation using 60 GPUs/V100s on Summit

Tracer particles: N. Lindquist (UTK), M. Min (ANL)

Accuracy:

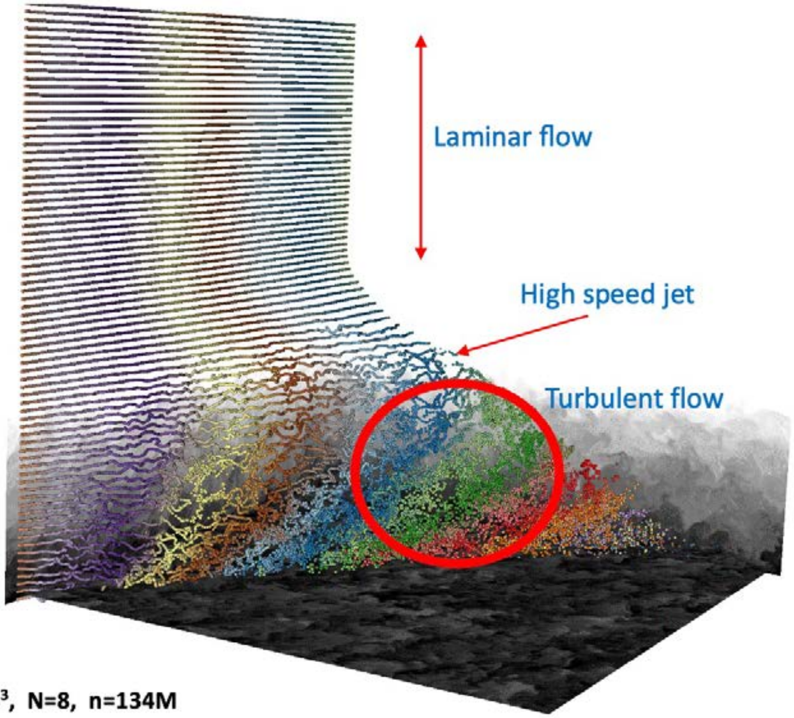
We compare the spectra to ***Kolmogorov '41*** theory, which predicts a $k^{-5/3}$ behavior in the inertial subrange (i.e., before viscous dissipation sets in).

Under-Resolved Example: GABLS Atmospheric Boundary Layer Benchmark

M. Min, A. Tomboudlies, P. Fischer (ANL), M. Brazell, M. Churchfield, M. Sprague (NREL)

Re = 50 M, WMLES

400m x 400m x 400m, Background flow 8 m/s

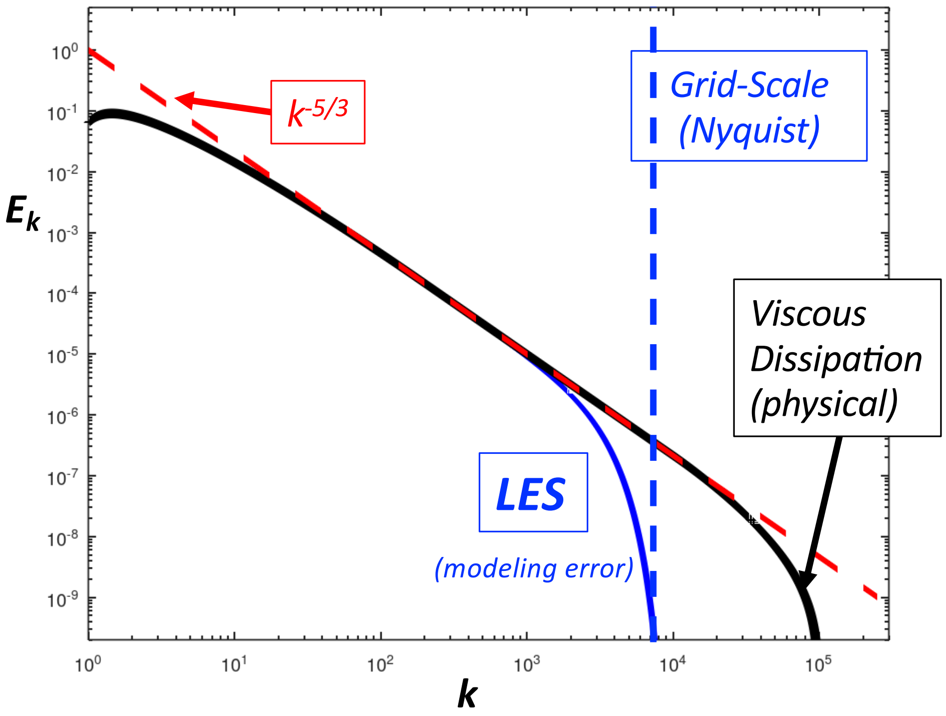


$E=64^3$, $N=8$, $n=134M$

NekRS simulation using 60 GPUs/V100s on Summit

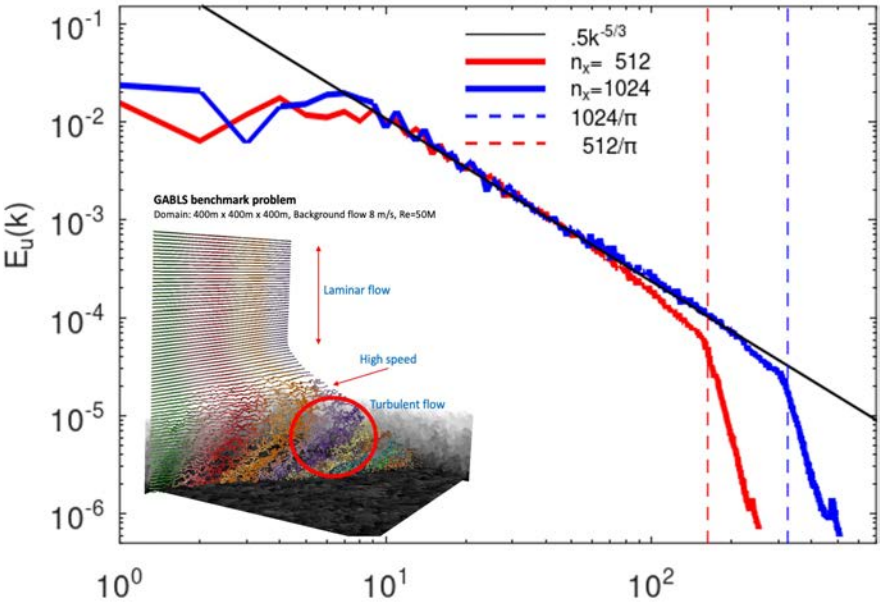
Tracer particles: N. Lindquist (UTK), M. Min (ANL)

LES Turbulence Spectra Illustration



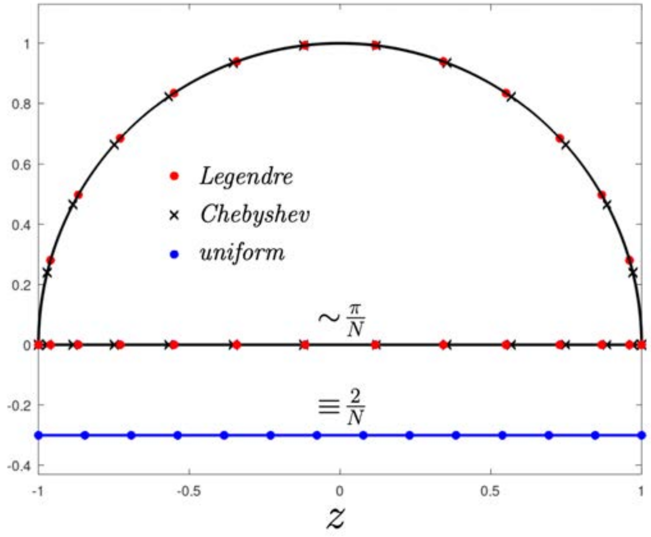
Nek5000/RS Accuracy: GABLS Turbulence Spectra @ z=100m, t=7h

Ananias Tomboulides (ANL/AUTh), Matt Churchfield (NREL),
Misun Min (ANL), PF, Mike Sprague (NREL)



512^3

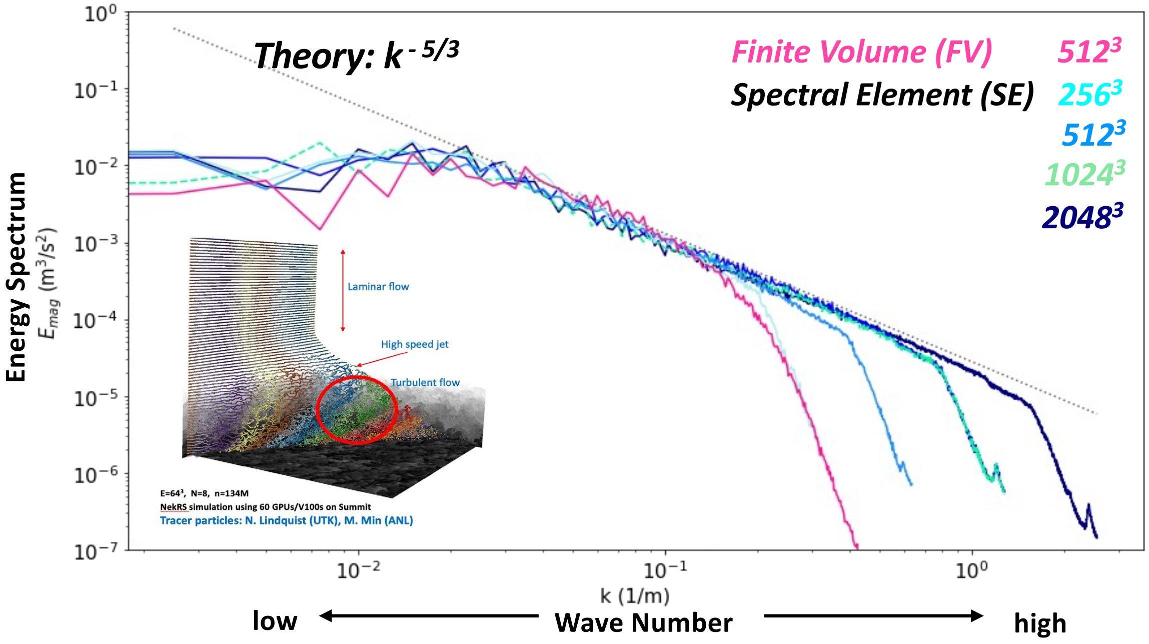
1024^3

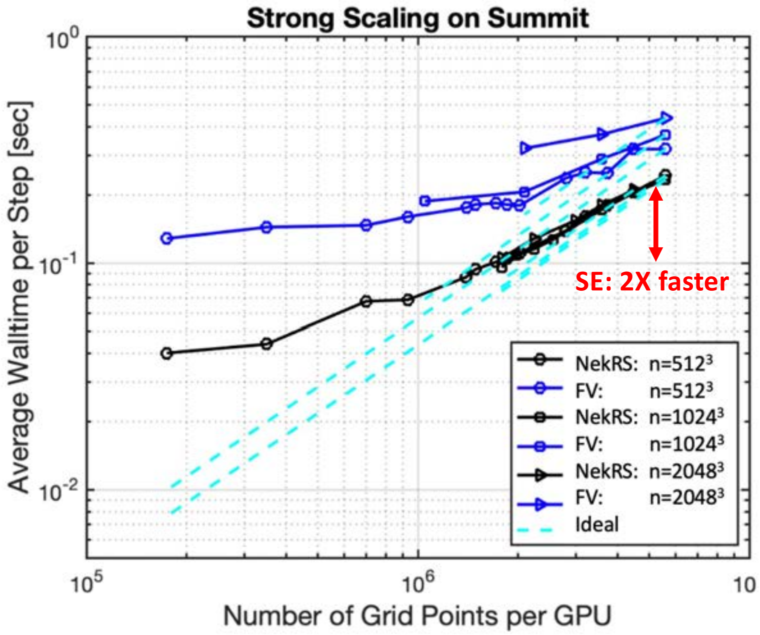


- ❑ The Nek5000/RS WMLES spectra demonstrate that the 8th-order SEM results are close to the maximum realizable.
- ❑ ***All scales, out to the Nyquist sampling frequency of n/π, are faithfully resolved.***

High-Order is Good for Accuracy and Performance

- Energy spectrum through extensive simulations for atmospheric boundary layers [1,2]
- Performance comparisons: ***256^3*** *order-8 SE* outperforms **512^3** *order-2 FV* simulations
- 1/8 the number of gridpoints, 1/2 the number of timesteps, 2X performance per gridpoint → ***32X advantage***





[1] M. Min, M. Brazell, A. Tomboulides, M. Churchfield, P. Fischer, and M. Sprague. Towards exascale for wind energy simulations. *Int. J. High Perf. Comp. Appl.,* 2024

[2] A. Tomboulides, M. Churchfield, P. Fischer, M. Sprague, and M. Min. Simulating Atmospheric Boundary Layer Turbulence, *J. Atm. Science,* in preparation.

OUTLINE

- ❑ *Improved Discretizations*
- ❑ ***High-performance on each GPU***
- ❑ *As many GPUs as possible*

SEM Operator Evaluation

- An important observation is that, for 3D problems, the fastest solvers are iterative methods, which only require operator evaluation - not operator formation

- Thus, along with timestepping, the only thing required for 3D flow simulations is ***operator evaluation*** (and advanced multilevel preconditioners for the pressure Poisson problem).

- For PDEs, this entails evaluating derivatives and integrals

- The high-order ***GLL-based polynomials*** of the SEM provide ***accuracy and stability*** needed for this purpose.

- The ***tensor-product forms provide efficiency.***

Fast Operator Evaluation: Matrix-Matrix Products / Tensor Contractions

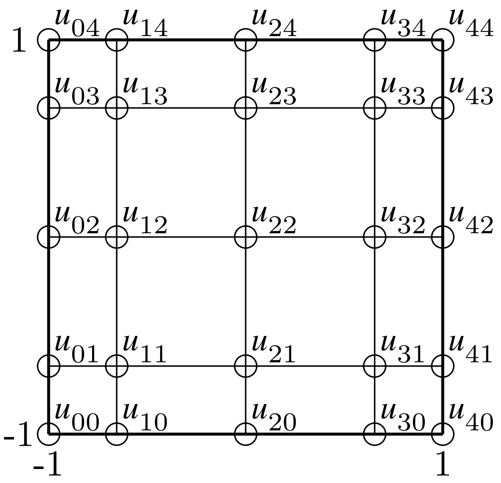
Consider a single element, $(r,s) \in \hat{\Omega} := [-1,1]^2$:

- $$u(r,s) = \sum_{j=0}^{N} \sum_{i=0}^{N} u_{ij}\, l_i(r) l_j(s), \qquad l_i(r_j) = \delta_{ij}$$

- $$\left.\frac{\partial u}{\partial r}\right|_{r_i,s_j} = \sum_q \sum_p u_{pq} \left.\frac{dl_p}{dr}\right|_{r_i} l_q(s_j)$$

$$= \sum_p \boxed{\hat{D}_{ip} u_{pj}} = (I \otimes \hat{D})\underline{u} \qquad \left\{ \begin{array}{l} \text{matrix-matrix product} \\ \text{tensor contraction} \end{array} \right.$$

BLAS3



u_{04} u_{14} u_{24} u_{34} u_{44}
u_{03} u_{13} u_{23} u_{33} u_{43}
u_{02} u_{12} u_{22} u_{32} u_{42}
u_{01} u_{11} u_{21} u_{31} u_{41}
u_{00} u_{10} u_{20} u_{30} u_{40}
1
-1
-1
1

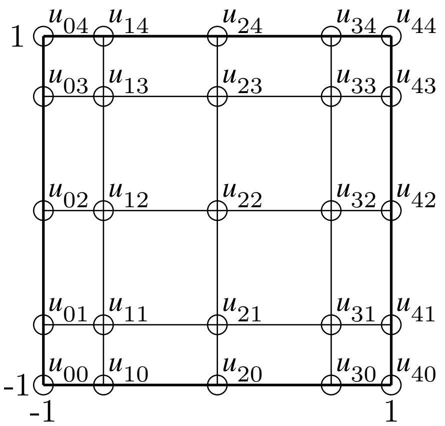
- SEM performance design objective:

  All evaluations with $O(n) = O(EN^3)$ *reads and* $\leq O(EN^4)$ *ops.*

Fast Operator Evaluation: Matrix-Matrix Products / Tensor Contractions

Complexity improves with increasing space dimension, $d = 1,\ 2,\ 3$:

- 1D : $\underline{u}_r \; = \sum_p \hat{D}_{ip} u_p \qquad N^2$ reads, $\quad 2N^2$ ops.
- 2D : $\underline{u}_r \; = \sum_p \hat{D}_{ip} u_{pj} \qquad 2N^2$ reads, $\quad 2N^3$ ops.
- 3D : $\underline{u}_r \; = \sum_p \hat{D}_{ip} u_{pjk} \qquad N^3$ reads, $\quad 2N^4$ ops.


u_{04} u_{14} u_{24} u_{34} u_{44}
u_{03} u_{13} u_{23} u_{33} u_{43}
u_{02} u_{12} u_{22} u_{32} u_{42}
u_{01} u_{11} u_{21} u_{31} u_{41}
u_{00} u_{10} u_{20} u_{30} u_{40}
1
-1
-1
1

Note, nodes on $\hat{\Omega} = [-1, 1]^d$ are taken to be tensor products of the Gauss-Lobatto-Legendre points, ξ_j, the roots of $(1 - \xi^2)P'_N(\xi)$, where P_N is the Legendre polynomial of degree N.

- Stability: Condition number of $\hat{A} = O(N^3)$, $\hat{a}_{ij} := \int_{\hat{\Omega}} \nabla\phi_i \cdot \nabla\phi_j \, dV$.
- Accurate quadrature: *diagonal mass matrix*

Spectral Element Matrix-Free Operator Evaluation *Orszag 80*

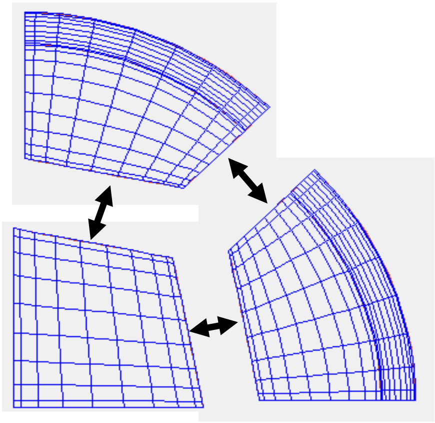
- ❑ Spectral element coefficients stored in local ($\underline{u}_L$) form, not global ($\underline{u}$)
- ❑ Example: *Application of SEM Laplacian*

$$\underline{w} = A\underline{u} = Q^T A_L Q\underline{u}, \qquad \underline{w}_L := Q\underline{w}, \qquad \underline{u}_L := Q\underline{u}$$

$$\underline{w}_L = \boxed{QQ^T} A_L \underline{u}_L$$

local work (matrix-matrix products)

nearest-neighbor (gather-scatter) exchange



QQ^T**: Gather-Scatter**

→ *Communication*

$$A_L \; := \begin{bmatrix} A^1 & & & \\ & A^2 & & \\ & & \ddots & \\ & & & A^E \end{bmatrix}$$

$$A^e \underline{u}^e = \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix}^T \begin{pmatrix} G_{rr} & G_{rs} & G_{rt} \\ G_{rs} & G_{ss} & G_{st} \\ G_{rt} & G_{st} & G_{tt} \end{pmatrix} \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix} \underline{u}^e$$

Matrix free form :

$\cdot\; 7N^3$ memory ref.

$\cdot\; 12N^4 + 15N^3$ ops.

$$D_r = (I \otimes I \otimes \hat{D}) \qquad\qquad G_{rs} = J \circ B \circ \left(\frac{\partial r}{\partial x}\frac{\partial s}{\partial x} + \frac{\partial r}{\partial y}\frac{\partial s}{\partial y} + \frac{\partial r}{\partial z}\frac{\partial s}{\partial z} \right)$$

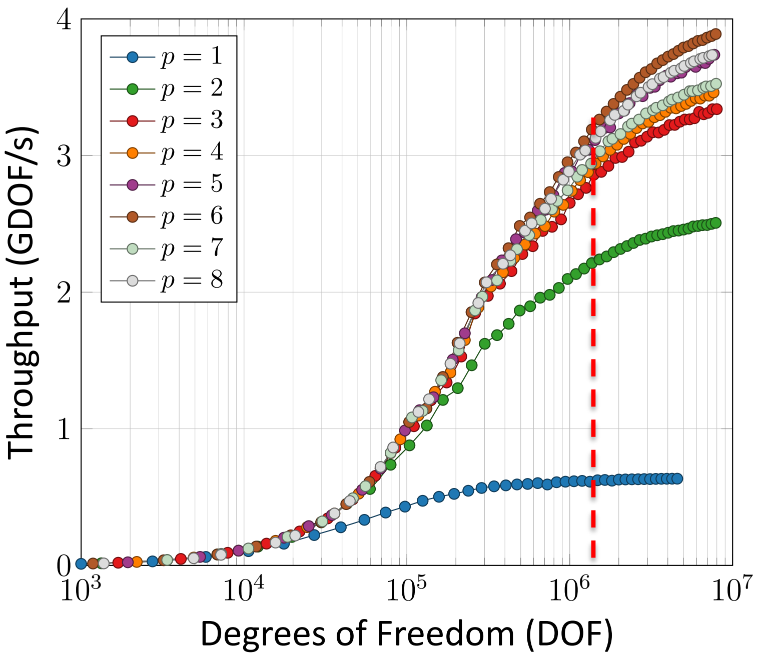
Majority of ops. *Majority of memory refs.*

$n_{0.8}$ on a Single GPU (V100)

John Camier, LLNL (CEED MS37, 2022)

Structured data accesses → higher order FEM outperforms lower order FEM

MFEM BP1 FAST @ Lassen V100



Runtime Kernel Tuning

- On the GPUs, we JIT-compile every kernel to match the data sizes, which allows the compiler to optimize the code

- So, for p-multigrid, we need $A\underline{u}$ for each polynomial order visited in the multigrid relaxtion process (e.g., N=7, 5, 3, 1)

- It is essentially the same operation for any value of N, but the array sizes and, hence, loop bounds differ for each N

  - One-time JIT compiled code for each case allows optimal performance

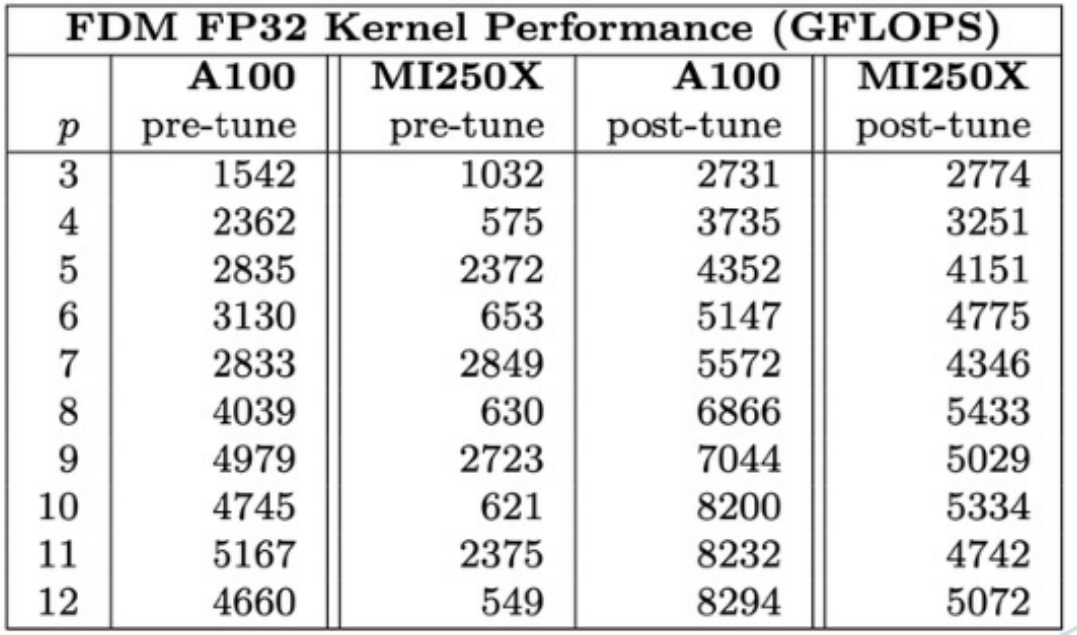
- Moreover, we have 3-12 different kernels for each operation (e.g., diffusion, advection, and interpolation)

  - For each operation, NekRS runs a battery of tests to select the fastest

  - Different data sizes and different platforms may select different kernels

Highly-Tuned Kernels for Tensor Contractions, FP32 and FP64

Tuning Results for FP32 Fast-Diagonalization-Method: T. Warburton



FDM FP32 Kernel Performance (GFLOPS)

	A100	MI250X	A100	MI250X
p	pre-tune	pre-tune	post-tune	post-tune
3	1542	1032	2731	2774
4	2362	575	3735	3251
5	2835	2372	4352	4151
6	3130	653	5147	4775
7	2833	2849	5572	4346
8	4039	630	6866	5433
9	4979	2723	7044	5029
10	4745	621	8200	5334
11	5167	2375	8232	4742
12	4660	549	8294	5072

$$\underline{\tilde{u}}^e = (S_z \otimes S_y \otimes S_x)\Lambda^{-1}(S_z^T \otimes S_y^T \otimes S_x^T)\underline{b}^e$$

From NekRS logfile, Perlmutter, SS10:

```
  Ax: N=7 FP64 GDOF/s=13.2 GB/s=1260 GFLOPS=2183 kv0
  Ax: N=3 FP64 GDOF/s=12.6 GB/s=1913 GFLOPS=1883 kv5

  Ax: N=7 FP32 GDOF/s=25.0 GB/s=1194 GFLOPS=4145 kv4
  Ax: N=3 FP32 GDOF/s=18.0 GB/s=1368 GFLOPS=2693 kv2

fdm: N=9 FP32 GDOF/s=44.9 GB/s= 812 GFLOPS=7452 kv4
fdm: N=5 FP32 GDOF/s=34.1 GB/s= 825 GFLOPS=4301 kv1

flop/s  3.36729e+13  (701 GFLOPS/rank)
```

❑ ***NekRS** picks the optimal kernel at runtime, for each pMG order (e.g., N=7, 5, 3)*

Auto-Tuning Avoids Performance Regression

ellipticBlockPartialAxCoeffHex3D

V1 reduces register pressure by utilizing multiple planes (Peng Wang, NVDIA)

Gflop/s

A100	5	6	7	8	9	10
v0	2253.03	2782.43	3134.98	2996.22	3657.43	3684.28
v1	2943.87	3072.6	4129.65	3582.21	3465.56	3145.26

H100	5	6	7	8	9	10
v0	3015.31	3801.18	6114.79	4261.23	4587.07	5101.18
v1	5493.15	5937.82	7662.32	6683.8	6640.17	6090.34

Notice the performance regression for V1, N=10.

NekRS would not encounter any regression because the autotuner would select to use V0 at runtime.

NekRS also uses auto-tuned communication

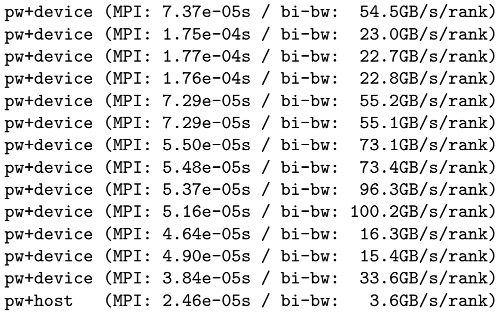
- ❑ ***Runtime tests select the best option for each communication topology and precision***

- ❑ These include
  - ❑ Pack on device + GPUDirect
  - ❑ Pack on device, communicate pairwise via host
  - ❑ Pack on host, communicate pairwise via host
  - ❑ Etc.

- ❑ The tests output also provides useful diagnostics.

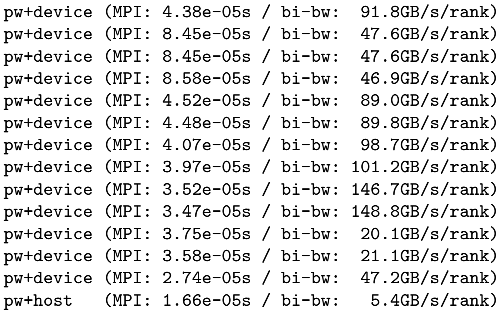
From NekRS logfile, Perlmutter:

SS10:



```
pw+device (MPI: 7.37e-05s / bi-bw:  54.5GB/s/rank)
pw+device (MPI: 1.75e-04s / bi-bw:  23.0GB/s/rank)
pw+device (MPI: 1.77e-04s / bi-bw:  22.7GB/s/rank)
pw+device (MPI: 1.76e-04s / bi-bw:  22.8GB/s/rank)
pw+device (MPI: 7.29e-05s / bi-bw:  55.2GB/s/rank)
pw+device (MPI: 7.29e-05s / bi-bw:  55.1GB/s/rank)
pw+device (MPI: 5.50e-05s / bi-bw:  73.1GB/s/rank)
pw+device (MPI: 5.48e-05s / bi-bw:  73.4GB/s/rank)
pw+device (MPI: 5.37e-05s / bi-bw:  96.3GB/s/rank)
pw+device (MPI: 5.16e-05s / bi-bw: 100.2GB/s/rank)
pw+device (MPI: 4.64e-05s / bi-bw:  16.3GB/s/rank)
pw+device (MPI: 4.90e-05s / bi-bw:  15.4GB/s/rank)
pw+device (MPI: 3.84e-05s / bi-bw:  33.6GB/s/rank)
pw+host   (MPI: 2.46e-05s / bi-bw:   3.6GB/s/rank)
```

SS11:



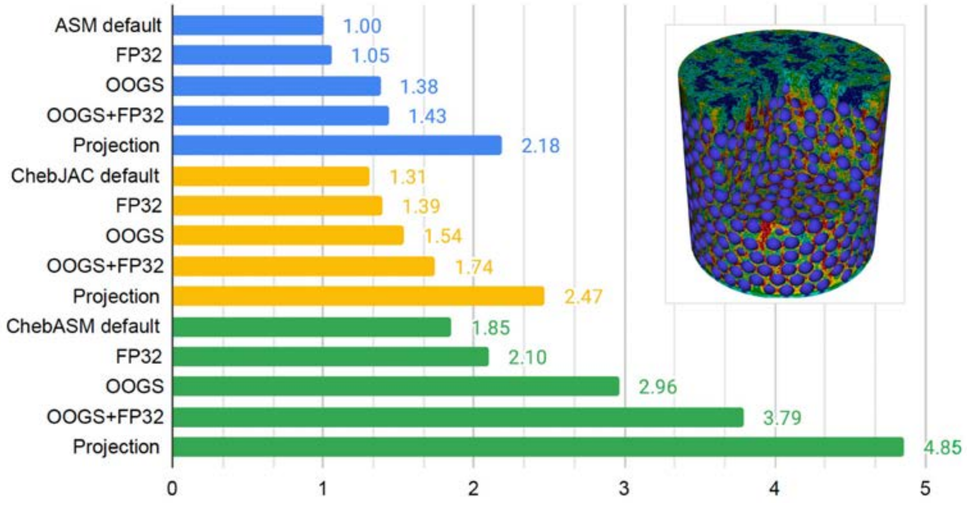
```
pw+device (MPI: 4.38e-05s / bi-bw:  91.8GB/s/rank)
pw+device (MPI: 8.45e-05s / bi-bw:  47.6GB/s/rank)
pw+device (MPI: 8.45e-05s / bi-bw:  47.6GB/s/rank)
pw+device (MPI: 8.58e-05s / bi-bw:  46.9GB/s/rank)
pw+device (MPI: 4.52e-05s / bi-bw:  89.0GB/s/rank)
pw+device (MPI: 4.48e-05s / bi-bw:  89.8GB/s/rank)
pw+device (MPI: 4.07e-05s / bi-bw:  98.7GB/s/rank)
pw+device (MPI: 3.97e-05s / bi-bw: 101.2GB/s/rank)
pw+device (MPI: 3.52e-05s / bi-bw: 146.7GB/s/rank)
pw+device (MPI: 3.47e-05s / bi-bw: 148.8GB/s/rank)
pw+device (MPI: 3.75e-05s / bi-bw:  20.1GB/s/rank)
pw+device (MPI: 3.58e-05s / bi-bw:  21.1GB/s/rank)
pw+device (MPI: 2.74e-05s / bi-bw:  47.2GB/s/rank)
pw+host   (MPI: 1.66e-05s / bi-bw:   5.4GB/s/rank)
```

Pressure Solve Improvements

Main conclusions

❑ overlapping communication/computation yields ~ 15% in pressure time

❑ fp32 in preconditioner can yield 10-15%. Often, the fp32 advantage derives from reduced bandwidth demand on the network. ***Q: Role of strong scaling?***



Algorithmic Impact on Pressure Solve Times, 1568-pebble case.

OUTLINE

- ❑ *Improved Discretizations*
- ❑ *High-performance on each GPU*
- ❑ ***As many GPUs as possible***

Parallelism: Stong-Scaling, Time to Solution, and Energy Consumption

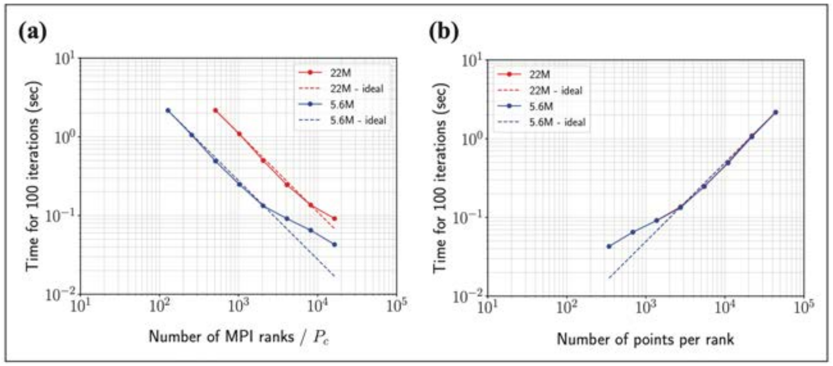


Figure 1. Strong-scale study for BP5-Nek5000 with n= 22 M and 5.6 M. n/P_c is the problem size per core, and strong-scale limit is observed at $n/P_c = 2744$. (a) BP5: time versus P_c. (b) BP5: time versus n/P_c.

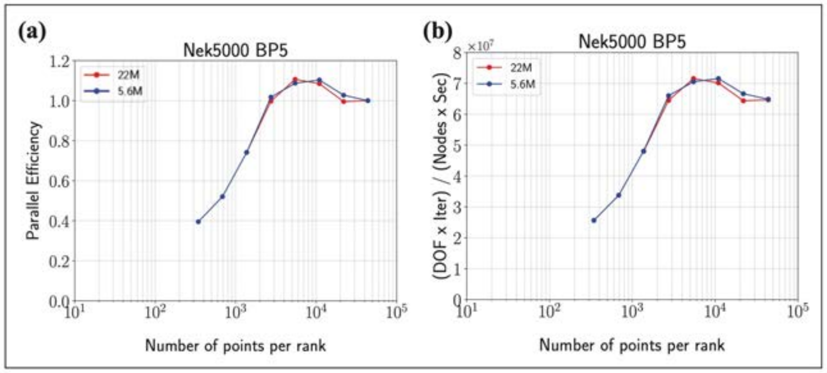


Figure 2. Strong-scale study for BP5-Nek5000. n/P_c is the problem size per core. Order unity parallel efficiency can be achieved for $n/P_c \geq 2744$. (a) BP5: efficiency versus n/P_c. (b) BP5: DOFs versus n/P_c. DOFs: degrees of freedom.

- ***These results suggest the idea of "n-scaling," in which we keep P fixed and alter the problem size, n.***

- ***This approach was taken in our CEED Bake-Off problems so that we could "strong-scale" without having to use enormous processor counts.***

- ***Idea is to fix P and monitor performance as function of (n/P) - performance is weakly dependent on P.***

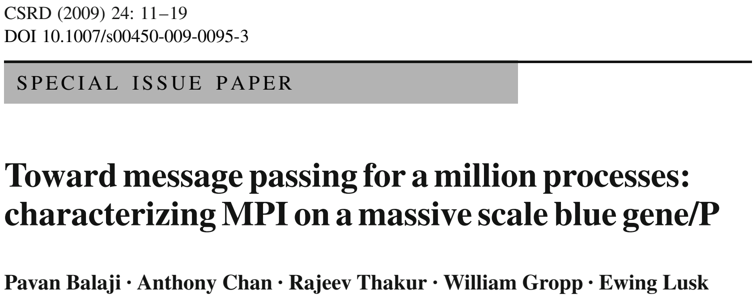
Fischer, Min, Rathnayake, Dutta, Kolev, Dobrev, Camier, Kronbichler, Warburton, Swirydowicz, and Brown. **Scalability of high-performance PDE solvers**. Int. J. of High Perf. Comp. Appl., 34(5):562–586, 2020.

Exascale Challenges - Scalability

- Key point:
  - Performance, $S_P = \eta P S_1$
  - Just definition of η.
- Main things are to:
  - Boost S_1
  - Keep η from falling as P is increased
- Scalability of an application:
  - Nature of problem/algorithm
  - Code (ideally, code doesn't matter - Bake-Offs)
  - Platform
  - Size of problem, n (number of spatial grid points)

P-fold speed-up

P=1 Million. Why not?



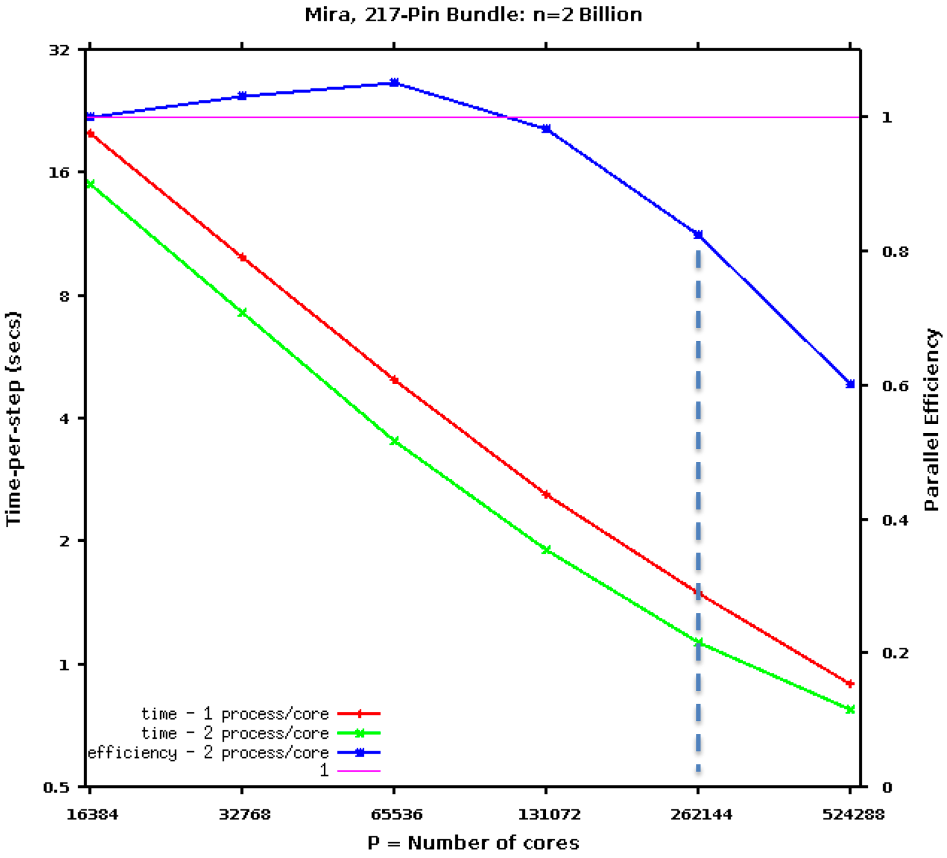
CSRD (2009) 24: 11–19
DOI 10.1007/s00450-009-0095-3

SPECIAL ISSUE PAPER

Toward message passing for a million processes: characterizing MPI on a massive scale blue gene/P

Pavan Balaji · Anthony Chan · Rajeev Thakur · William Gropp · Ewing Lusk

Strong Scaling to a Million Ranks (Mira, BG/Q)

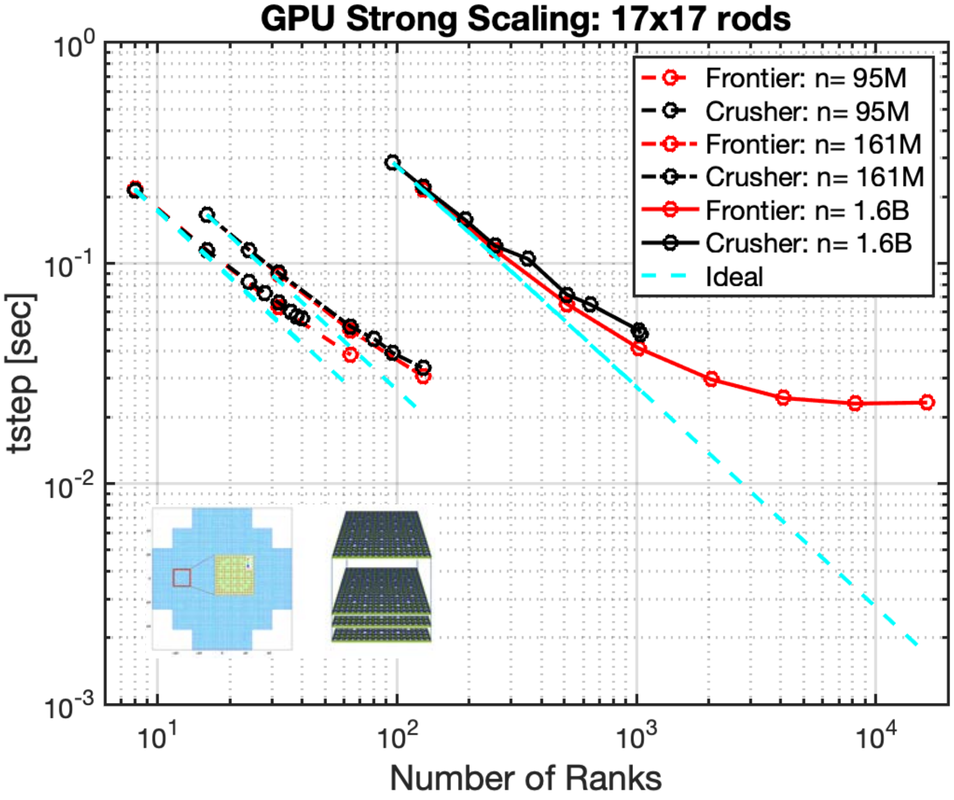


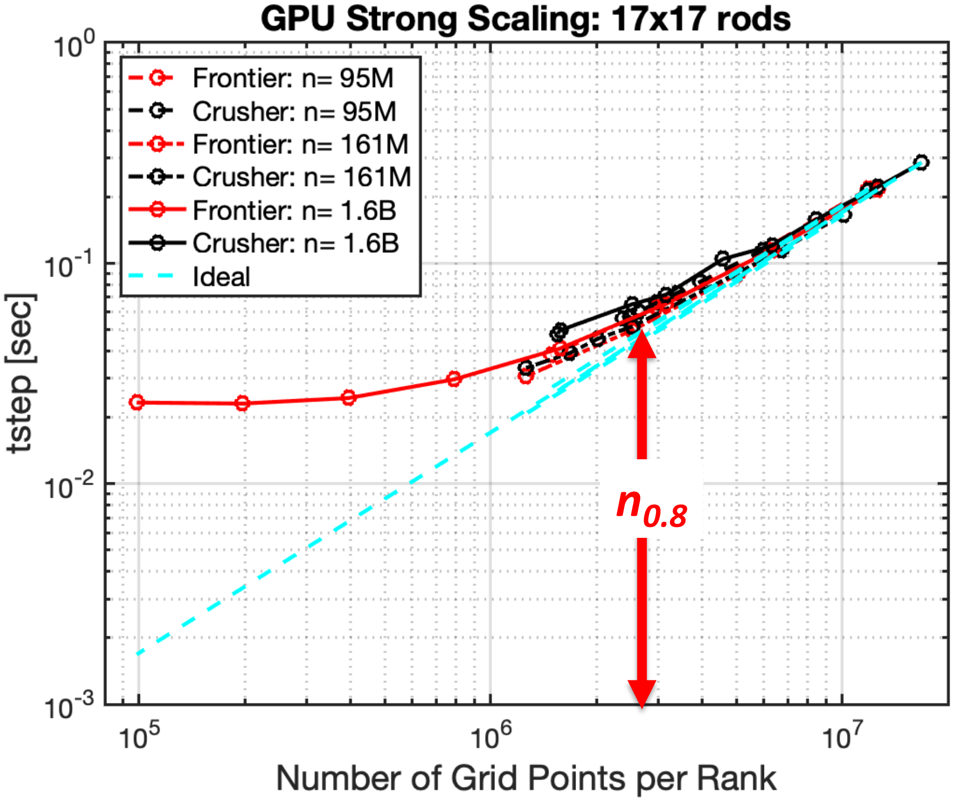
- ❑ Q: Do we use the 1-rank/core or 2-rank/core curve for strong-scale study?

  A: Whatever the user would do…
  (i.e., 2-rank/core, because it's faster)

- ❑ n = 2 billion
- ❑ $n_{0.8}$ = 2 B/(½ M) = 4000 points per rank
- ❑ Follow the practice of "user perspective" in presenting metrics, e.g.,
  - ❑ AMD-250X has 2 GCDs → 2 MPI ranks per 250X
  - ❑ Other architectures similar…

Strong-Scaling Example: ExaSMR Test Case on Frontier and Crusher





- ❑ ***Critical parameter: $n_{0.8}$ = number of points-per-rank to realize 80% efficiency.***
- ❑ ***This is where users will typically run and thus is the performance design point.***

Exascale Challenges - Scalability

- **General rule of thumb for PDEs:**
  - **If you double n, you can double P**
    $\longrightarrow$ key parameter is *size of problem per MPI rank* $= n/P$
- **Bottom line:**

  At strong-scale limit (where users generally run),
  time-to-solution $\sim \frac{W}{0.8}\,\frac{n_{0.8}}{S_1}$

$$\begin{aligned} W &= \text{number of flops per grid point} \\ n_{0.8} &= n/P, \text{ where } \eta \approx 0.8 \\ S_1 &= \text{processing rate (GFLOPS) on a single rank} \end{aligned}$$

- **To reduce time-to-solution, must not let the ratio $\underline{(n_{0.8}/S_1)}$ increase.**
- **It's clear, for example, that GPUs offer significant increases in S_1.**
- **Questions going into this project:**
  - ***How to maximize S_1?*** **(All in approach.)**
  - ***What happens to $n_{0.8}$?***

Addressing Efficiency Fall-Off

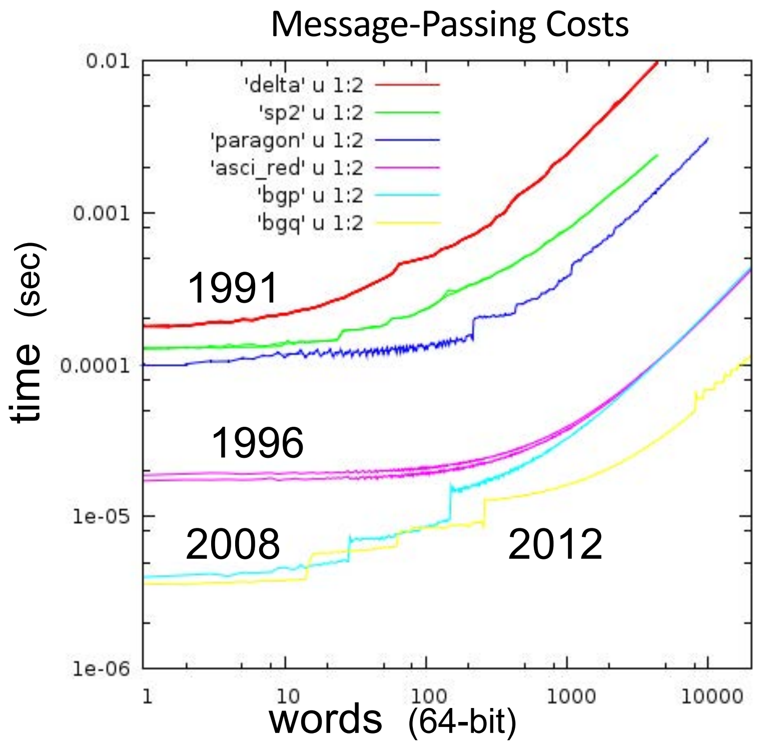
❑ From a User's perspective, for most PDE solvers, efficiency fall-off for CPUs and GPUs is generally different

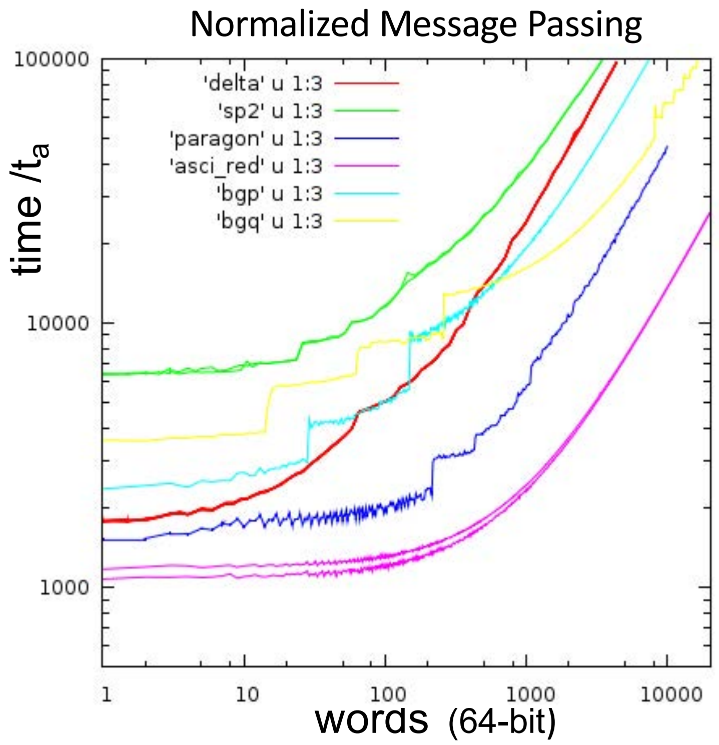
  ❑ CPUs - MPI latency effects (not bandwidth… WHY?)

  ❑ GPUs - GPU scalability ***and*** MPI latency/bandwidth effects

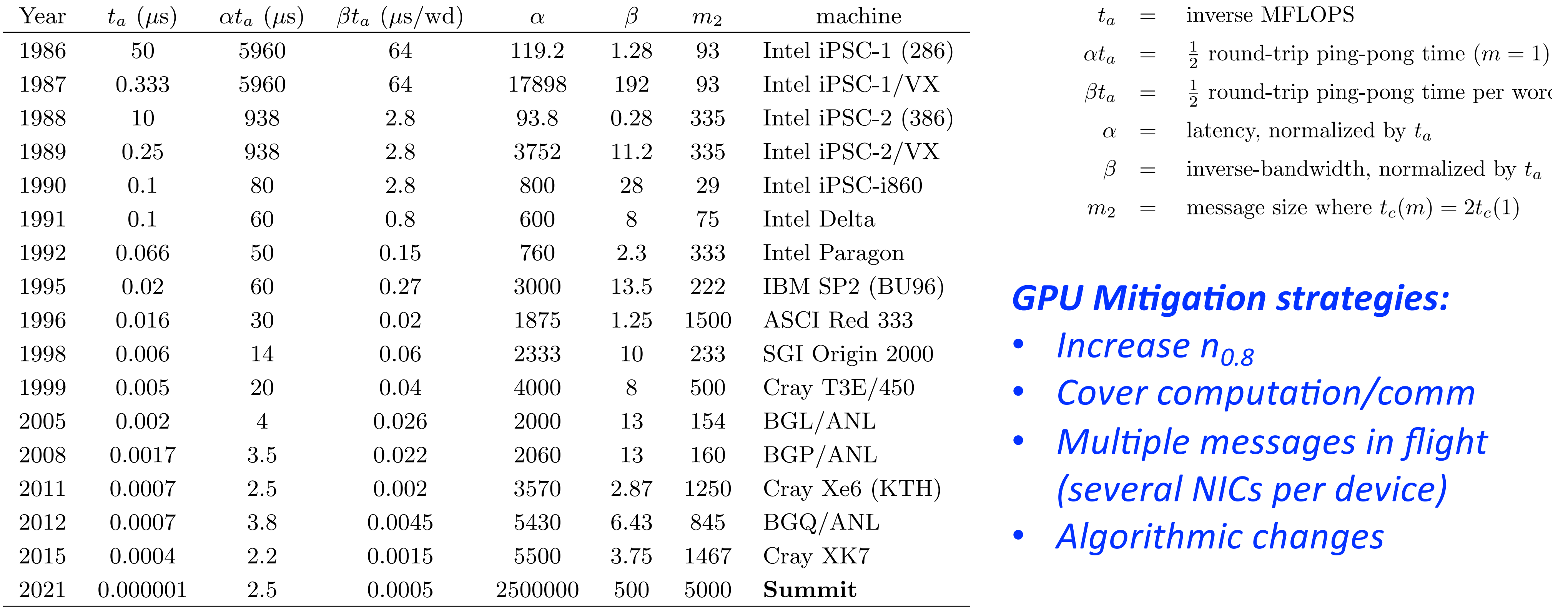
Early Ping-Pong Tests

- Postal model: $t_c(m) = (\alpha + \beta m)\, t_a$


Message-Passing Costs


Normalized Message Passing

35 Years of Ping-Pong Data



Year	t_a (μs)	αt_a (μs)	βt_a (μs/wd)	α	β	m_2	machine
1986	50	5960	64	119.2	1.28	93	Intel iPSC-1 (286)
1987	0.333	5960	64	17898	192	93	Intel iPSC-1/VX
1988	10	938	2.8	93.8	0.28	335	Intel iPSC-2 (386)
1989	0.25	938	2.8	3752	11.2	335	Intel iPSC-2/VX
1990	0.1	80	2.8	800	28	29	Intel iPSC-i860
1991	0.1	60	0.8	600	8	75	Intel Delta
1992	0.066	50	0.15	760	2.3	333	Intel Paragon
1995	0.02	60	0.27	3000	13.5	222	IBM SP2 (BU96)
1996	0.016	30	0.02	1875	1.25	1500	ASCI Red 333
1998	0.006	14	0.06	2333	10	233	SGI Origin 2000
1999	0.005	20	0.04	4000	8	500	Cray T3E/450
2005	0.002	4	0.026	2000	13	154	BGL/ANL
2008	0.0017	3.5	0.022	2060	13	160	BGP/ANL
2011	0.0007	2.5	0.002	3570	2.87	1250	Cray Xe6 (KTH)
2012	0.0007	3.8	0.0045	5430	6.43	845	BGQ/ANL
2015	0.0004	2.2	0.0015	5500	3.75	1467	Cray XK7
2021	0.000001	2.5	0.0005	2500000	500	5000	**Summit**

t_a = inverse MFLOPS

αt_a = $\frac{1}{2}$ round-trip ping-pong time ($m = 1$)

βt_a = $\frac{1}{2}$ round-trip ping-pong time per wor

α = latency, normalized by t_a

β = inverse-bandwidth, normalized by t_a

m_2 = message size where $t_c(m) = 2t_c(1)$

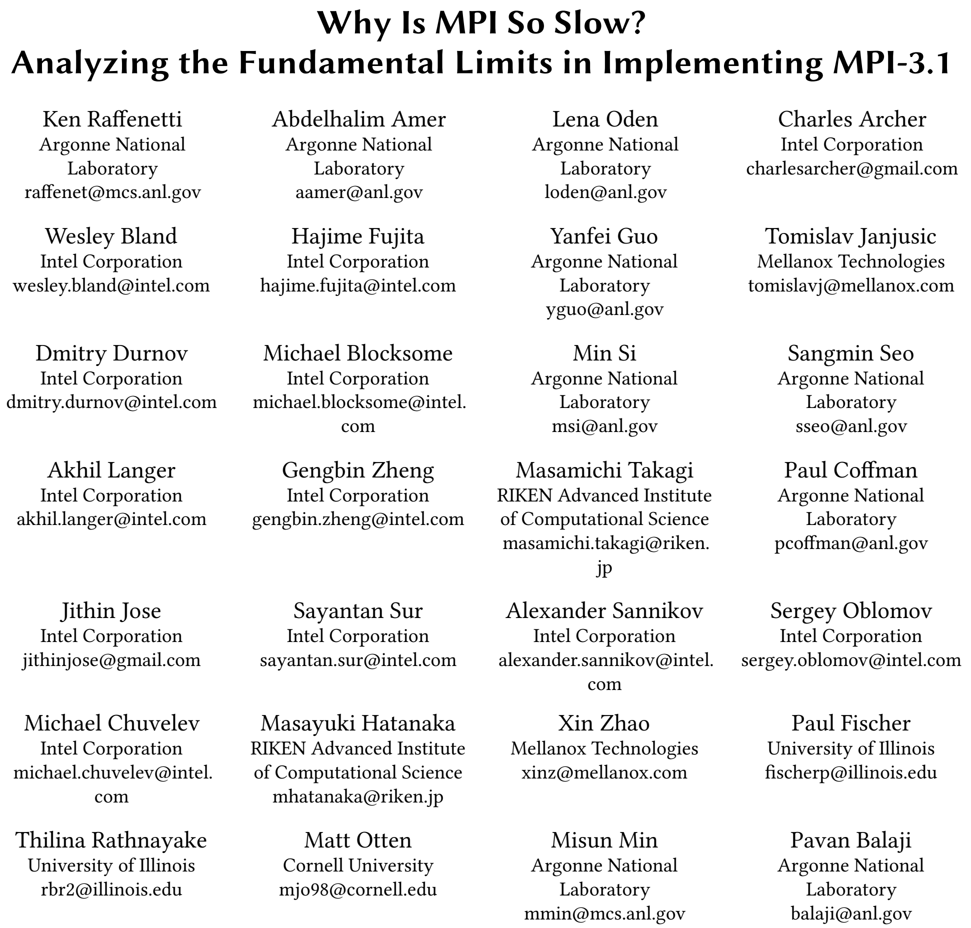
GPU Mitigation strategies:

- *Increase $n_{0.8}$*
- *Cover computation/comm*
- *Multiple messages in flight (several NICs per device)*
- *Algorithmic changes*

Latency-Mitigation Strategies - CPU

- ❑ Low-noise (convex) networks
- ❑ Hardware all-reduce
- ❑ …
- ❑ Not so much covering communication/computation. (WHY)?

- ❑ Looked at several of the issues in an SC17 effort led by Ken Raffenetti (ANL)



Why Is MPI So Slow?
Analyzing the Fundamental Limits in Implementing MPI-3.1

Ken Raffenetti
Argonne National Laboratory
raffenet@mcs.anl.gov

Abdelhalim Amer
Argonne National Laboratory
aamer@anl.gov

Lena Oden
Argonne National Laboratory
loden@anl.gov

Charles Archer
Intel Corporation
charlesarcher@gmail.com

Wesley Bland
Intel Corporation
wesley.bland@intel.com

Hajime Fujita
Intel Corporation
hajime.fujita@intel.com

Yanfei Guo
Argonne National Laboratory
yguo@anl.gov

Tomislav Janjusic
Mellanox Technologies
tomislavj@mellanox.com

Dmitry Durnov
Intel Corporation
dmitry.durnov@intel.com

Michael Blocksome
Intel Corporation
michael.blocksome@intel.com

Min Si
Argonne National Laboratory
msi@anl.gov

Sangmin Seo
Argonne National Laboratory
sseo@anl.gov

Akhil Langer
Intel Corporation
akhil.langer@intel.com

Gengbin Zheng
Intel Corporation
gengbin.zheng@intel.com

Masamichi Takagi
RIKEN Advanced Institute of Computational Science
masamichi.takagi@riken.jp

Paul Coffman
Argonne National Laboratory
pcoffman@anl.gov

Jithin Jose
Intel Corporation
jithinjose@gmail.com

Sayantan Sur
Intel Corporation
sayantan.sur@intel.com

Alexander Sannikov
Intel Corporation
alexander.sannikov@intel.com

Sergey Oblomov
Intel Corporation
sergey.oblomov@intel.com

Michael Chuvelev
Intel Corporation
michael.chuvelev@intel.com

Masayuki Hatanaka
RIKEN Advanced Institute of Computational Science
mhatanaka@riken.jp

Xin Zhao
Mellanox Technologies
xinz@mellanox.com

Paul Fischer
University of Illinois
fischerp@illinois.edu

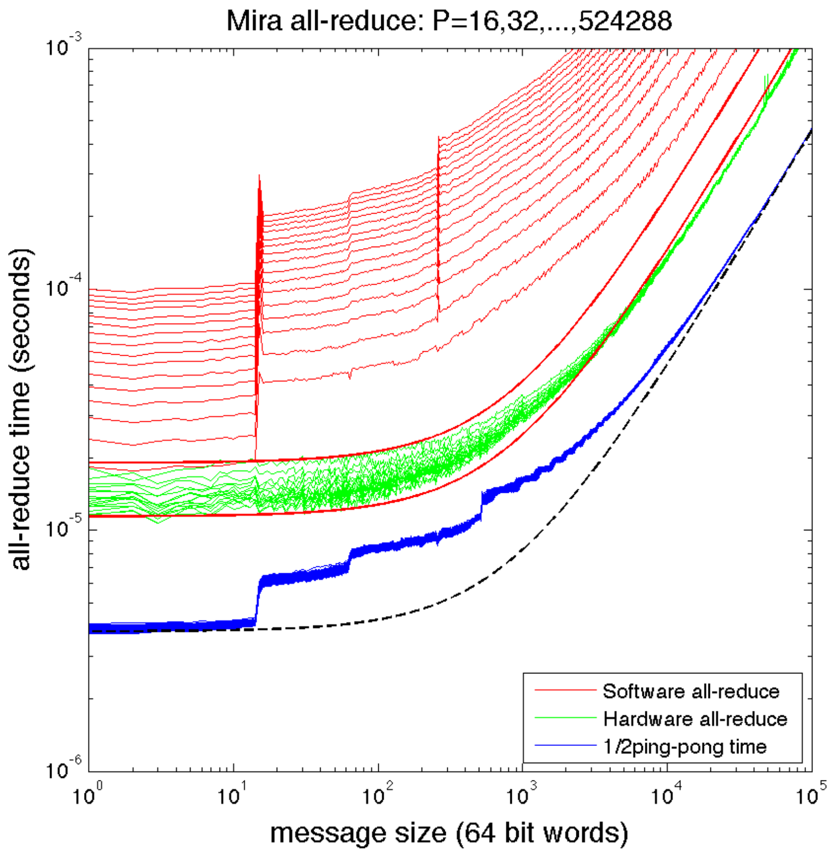
Thilina Rathnayake
University of Illinois
rbr2@illinois.edu

Matt Otten
Cornell University
mjo98@cornell.edu

Misun Min
Argonne National Laboratory
mmin@mcs.anl.gov

Pavan Balaji
Argonne National Laboratory
balaji@anl.gov

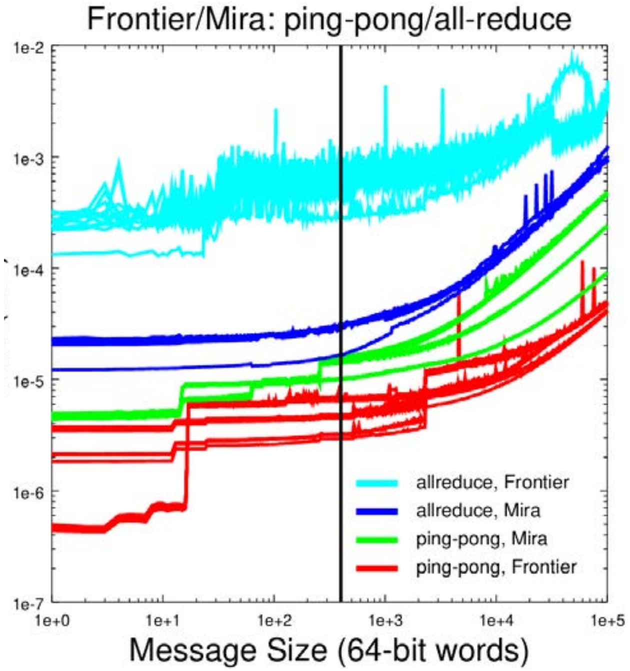
All-Reduce Cost Mitigation

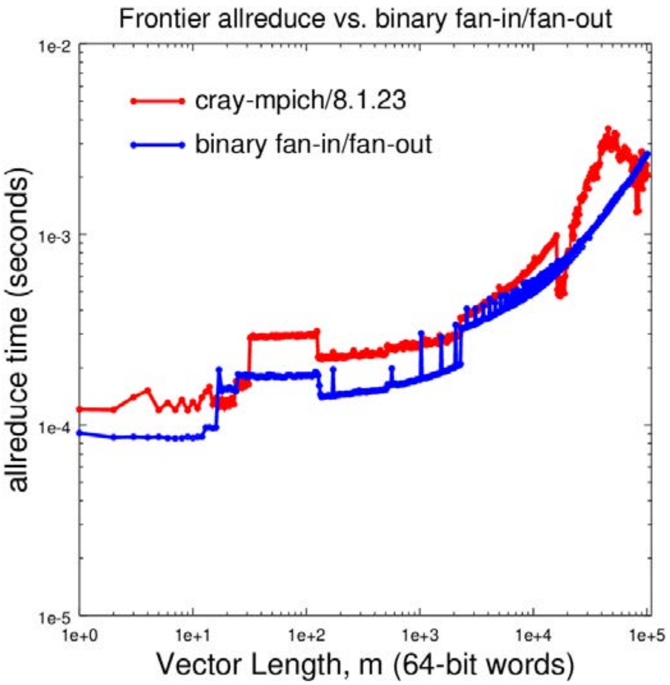


- ❑ *BG/Q (Mira, Sequoia, BG/P, BG/L)*
- ❑ *Isolated convex subnetworks - no traffic competing with User's resources*
- ❑ *18 cores per node - 16 compute, one for System, one for Yield*
- ❑ *All-reduce performed on NIC:*

  → 4 X [½ ping-pong latency time] !!
- ❑ *Even software all-reduce is reasonably fast*

Potential Problems with MPI *(SC23 MPI BOF)*





- Compare Frontier MPI with home-grown f77 all-reduce
- ~ 1.5 X faster than mpich/8.1.23 at several points
  (Discovered while developing a new coarse-grid solver…)

Surprises - Part & Parcel of HPC Since Its Inception - SS10 → SS11 Upgrade

❑ SS11 realized a 1.5X gain in bandwidth

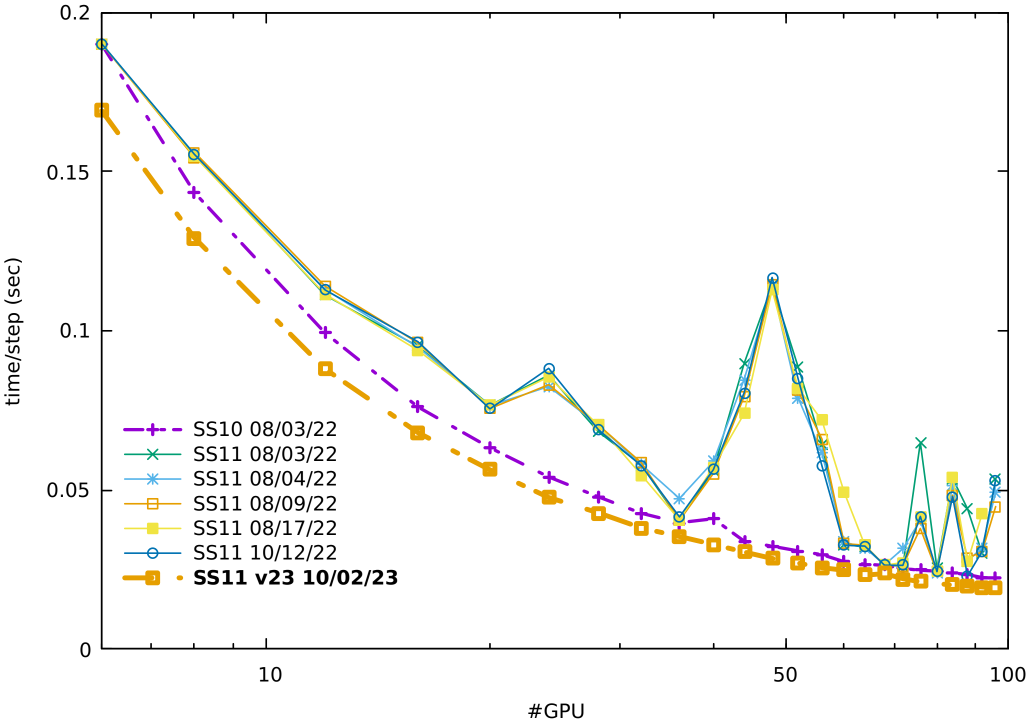
❑ However, flakey but repeatable message-passing costs yielded a 3X overall slowdown in NS solution performance.

❑ Issue: a handful of short messages in lowest levels of p-multigrid

❑ We were worried that Polaris (and other SS11 systems) would be the same.

❑ This issue resolved with later SS11 release

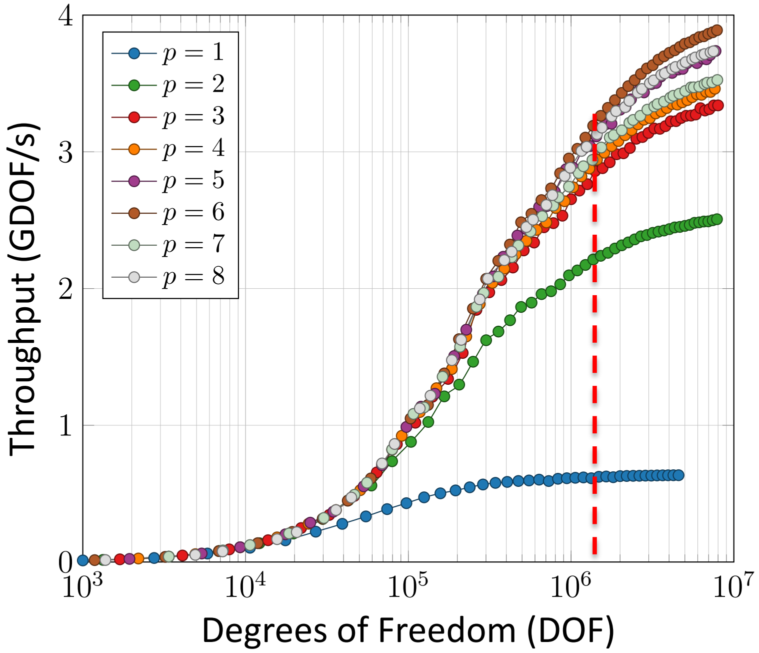
Navier-Stokes Solution Time - Strong Scaling, Perlmutter



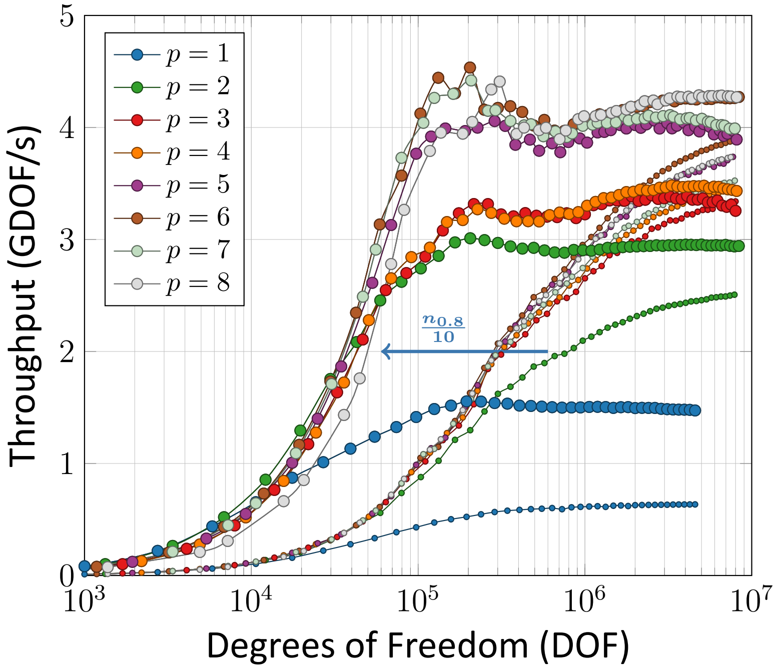
What about improving $n_{0.8}$ on a Single GPU? *John Camier, LLNL (CEED*

- CEED Bake-Off BP1: Throughput vs. Local Problem Size. *(Up and to the left is good.)*
- Can we mitigate kernel-launch overhead?

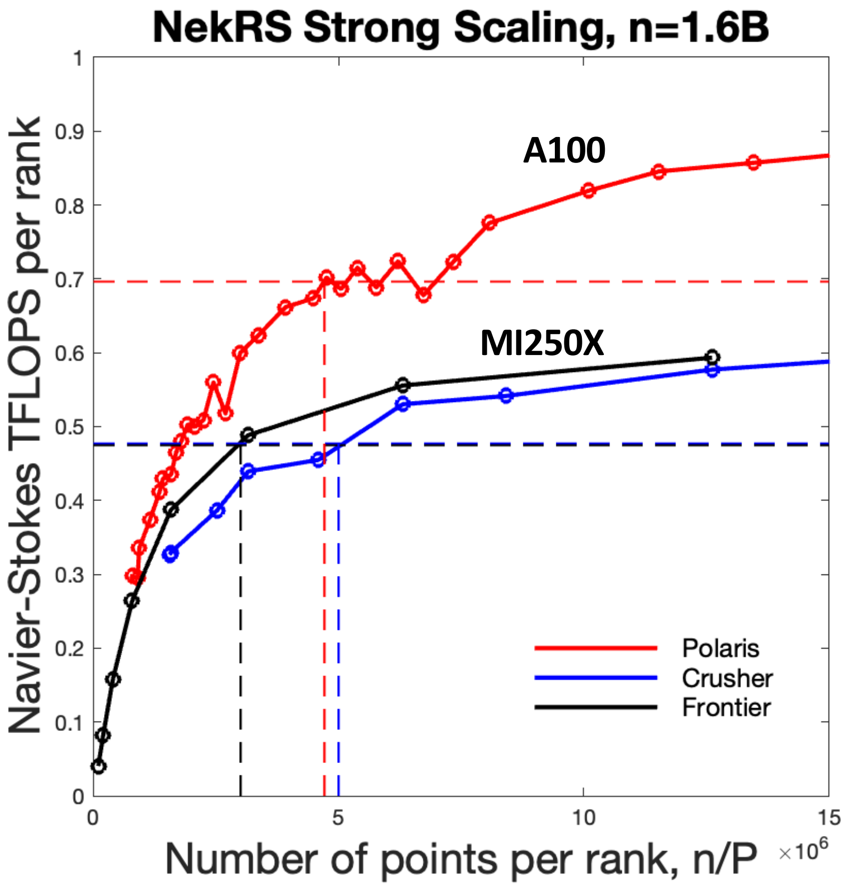
MFEM BP1 FAST @ Lassen V100



MFEM BP1 XFL vs FAST @ V100



Strong-Scaling Example: ExaSMR on Frontier, Crusher, Polaris



- ❑ ***While the A100 has higher peak performance, its $n_{0.8}$ ~ 5M per GPU***

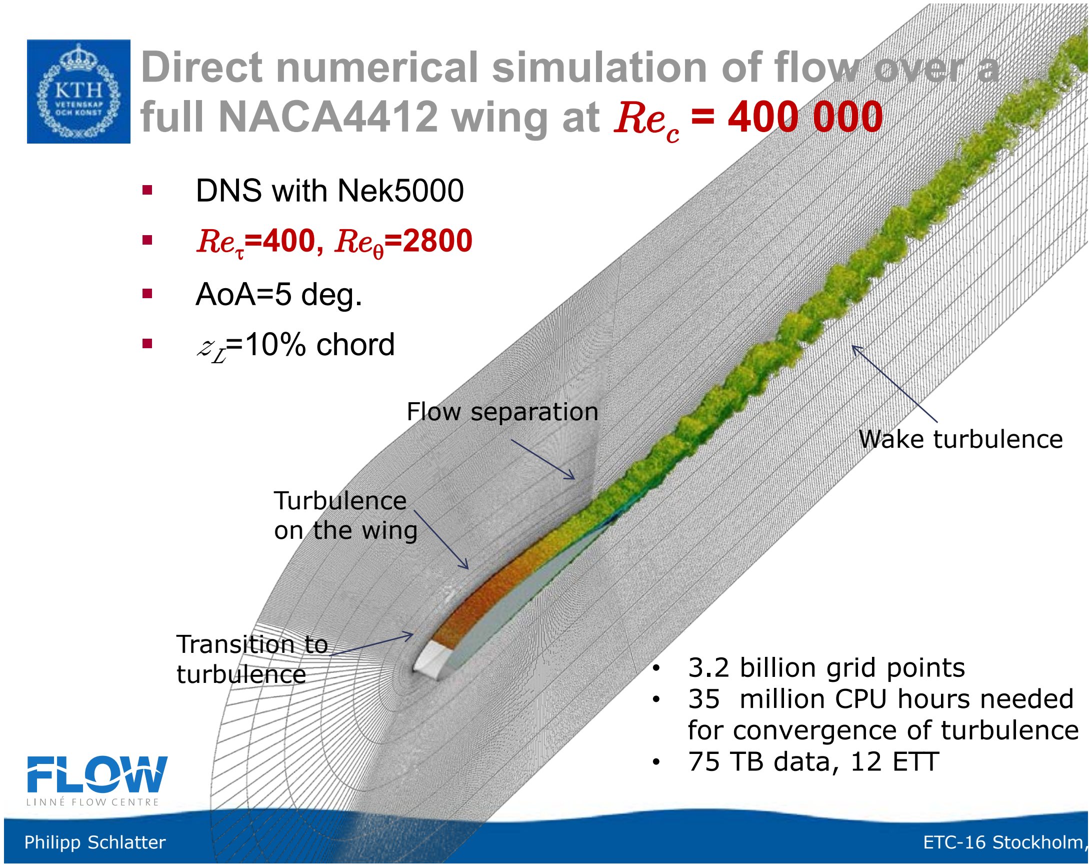
- ❑ ***For Frontier (MI250X), $n_{0.8}$ ~ 3M per GCD***

- ❑ ***At 80% efficiency, time to solution is actually lower (0.84) on Frontier than on Polaris because Frontier can use more ranks.***

- ❑ ***Note that if we try to run on Polaris at ~ 100% efficiency the time to solution will be > 3 x 0.8 = 2.4x longer.***

- 2.4 days, instead of 1 day.

Answering a Common Question: How long will my job take?



Direct numerical simulation of flow over a full NACA4412 wing at Re_c = 400 000

- DNS with Nek5000
- Re_τ=400, Re_θ=2800
- AoA=5 deg.
- z_L=10% chord

- 3.2 billion grid points
- 35 million CPU hours needed for convergence of turbulence
- 75 TB data, 12 ETT

Philipp Schlatter

ETC-16 Stockholm,

❑ Consider this hero calculation from a few years ago.

❑ How many A100s?

❑ How many A100 hours?

❑ How many wall-clock hours?

❑ 1000 A100s
- ❑ Each ~300X a CPU
- ❑ 110K GPU hours
- ❑ 110 wall clock hours

Putting it all together:

❑ ***$n_{0.8}$ ~ 2 M on V100***
~ 3 M on AMD MI250X (single GCD)
~ 4-5 M on A100

❑ ***Did we improve $n_{0.8}$ / S_1 ??***

Summit-Mira Comparison *Ramesh Balakrishnan ANL*

E=3.14M, N=7, n = 1.08B

Mira: *Nek5000*
P=524288 ranks (262144 cores)
n/P = 2060
0.496 s/step (CFL ~ 0.45)
24 hour run (of several)

Nek5000 DNS of flow past a periodic hill at Re=19,000 on ALCF Mira. Ramesh Balakrishnan, ANL

Summit: *NekRS*
P=528 ranks (528 V100s)
n/P = 2.05M
0.146 s/step (CFL ~ 0.45)
24 hour run (of several)

Summary:

At strong-scale limit (80% eff.)

- *NekRS+Summit → **3.4X faster** than Nek5000+Mira*
- *Requires about **10%** of Summit resources vs. **½** Mira*

Extreme Scalability: Full-Core Pebble Bed Simulations

Y.Lan, PF., E. Merzari,M.Min

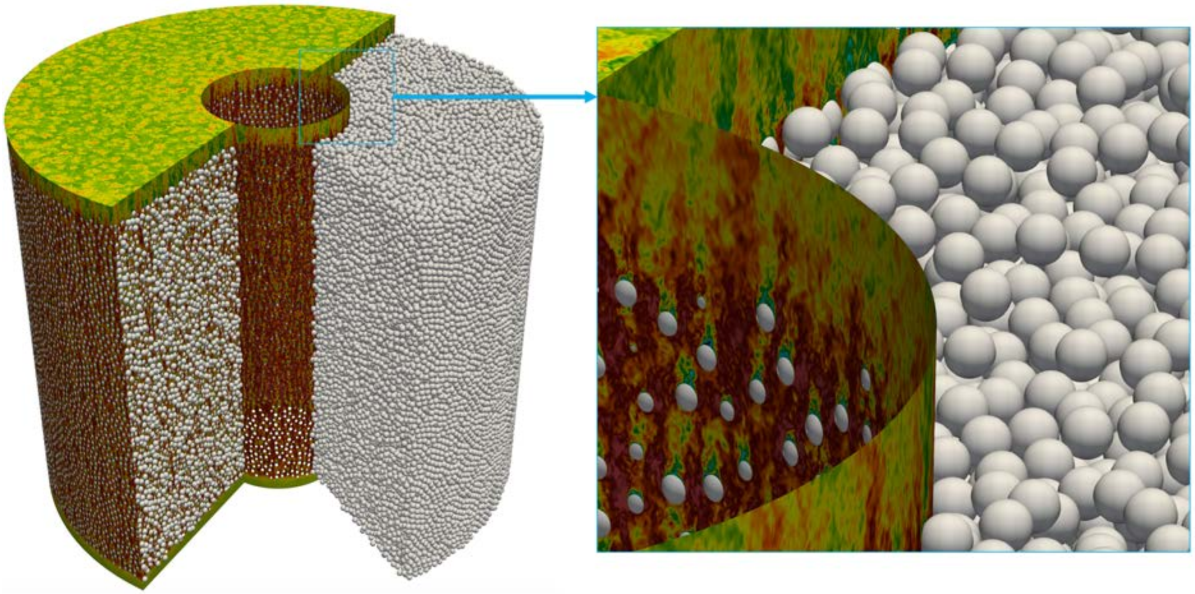


Figure 8: Turbulent flow in an annular packed bed with $\mathcal{N} = 352625$ spheres meshed with $E = 98,782,067$ spectral elements of order $N = 8$ ($n = 50$ billion gridpoints). This NekRS simulation requires 0.233 seconds per step using 27648 V100s on Summit. The average number of pressure iterations per step is 6.

- ❑ *352,625 spherical pebbles*
- ❑ *E=99 M elements*
- ❑ *N=51 B gridpoints*
- ❑ *1.4 TB per snapshot (FP32)*
- ❑ *P=27648 V100s (all of Summit)*

- ❑ *High quality all-hex mesh generated by tessellation of Voronoi facets that are projected onto the sphere or domain boundaries to yield hexahedral elements*

- ❑ *~300 elements / sphere*

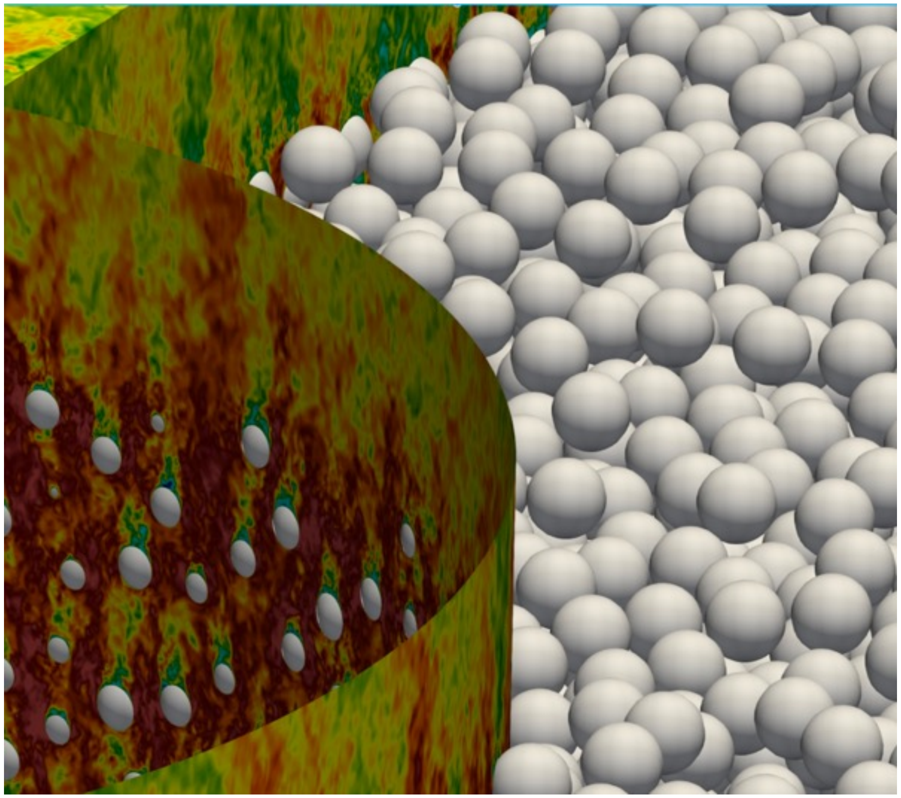
- ❑ *Turbulent flow in the interstitial region between the randomly-packed spheres.*

[1] Min et al., Optimization of Full-core Reactor Simulations on Summit, SC22 (https://ieeexplore.ieee.org/document/10046048)

[2] Lan et al., All-Hex Meshing Strategies For Densely Packed Spheres, The 29th International Meshing Roundtable, 2021

Net Improvements - Full Core Simulation

- ❑ ***Full flow-through in just 6 hours on Summit,*** which is a significant achievement compared to pre-ECP capabilities, both in size and speed.
- ❑ Record problem size on Mira was E=15M elements
- ❑ Here, E=98M on Summit and new runs on Frontier are at E=1.6B elements (~1 trillion gridpoints)



[1] Min et al., Optimization of Full-core Reactor Simulations on Summit, SC22 (https://ieeexplore.ieee.org/document/10046048)
[2] Lan et al., All-Hex Meshing Strategies For Densely Packed Spheres, The 29th International Meshing Roundtable, 2021

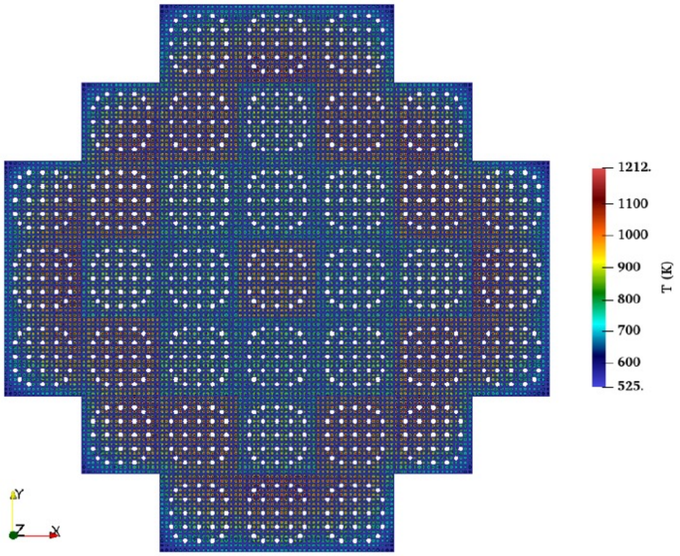
All of Frontier: SMR Full-Core Model

- The neutronics model includes the division of each fuel pin into three radial rings as well as the modeling of gap and cladding regions.

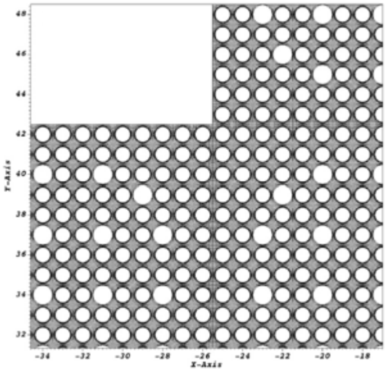
- The assembly model in NekRS was created with a mesh that was tuned to fully resolve the boundary layers for Re = 80,000 for a polynomial order of N=7 (343 points/elem.)

  - Coupled run was conducted with E = 1,098,530,000 element and 3.76 x 10^{11} grid points.

  - Standalone runs were also conducted with 6.03x10^{11} grid points.



Temperature distribution in the core



Example of the fluid mesh

E. Merzari, S. Hamilton, T. Evans, P. Romano, P. Fischer, M. Min, S. Kerkemeier, Y.H. Lan, J. Fang, M. Phillips, T. Rathnayake, E. Biondo, K. Royston, N. Chalmers, and T. Warburton. **Exascale multiphysics nuclear reactor simulations for advanced designs** (Gordon Bell Prize Finalist paper). In Proc. of SC23: Int. Conf. for High Performance Computing, Networking, Storage and Analysis. IEEE, 2023.

ExaSMR: 9000-Nodes Frontier Runs (72,000 GCDs)

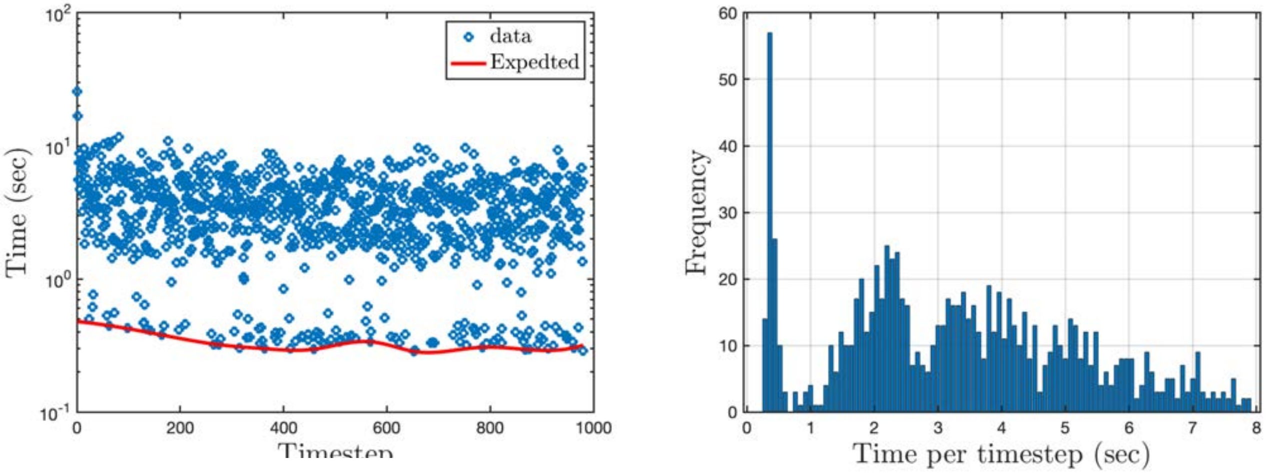
E. Merzari (PSU/ANL), Y. Lan, M. Min

❑ Time per-step, 300 B points, on 72,000 GCds ~ ***0.3 sec/step***.

❑ Except, with system noise, sometimes ***10 sec/step!***

❑ Many (difficult) trials isolated the issue to congestion in modestly communication-intensive routines.

April, 2023: NekRS Default



❑ ***Explored several strategies to reduce network congestion.***

- Turning off GPU direct was most effective.

E. Merzari, S. Hamilton, T. Evans, P. Romano, P. Fischer, M. Min, S. Kerkemeier, Y.H. Lan, J. Fang, M. Phillips, T. Rathnayake, E. Biondo, K. Royston, N. Chalmers, and T. Warburton. Exascale multiphysics nuclear reactor simulations for advanced designs (Gordon Bell Prize Finalist paper). In Proc. of SC23: Int. Conf. for High Performance Computing, Networking, Storage and Analysis. IEEE, 2023.

U.S. DEPARTMENT OF ENERGY | Office of Science

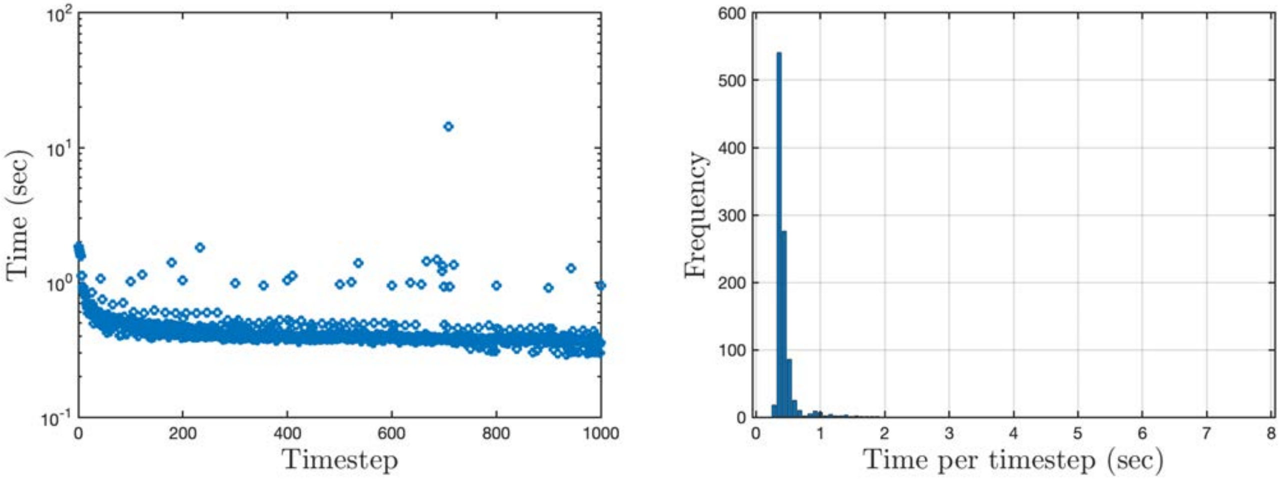
Argonne NATIONAL LABORATORY

ExaSMR: 9000-Nodes Frontier Runs (72,000 GCDs)

E. Merzari (PSU/ANL), Y. Lan, M. Min

- □ Time per-step, 300 B points, on 72,000 GCds ~ ***0.3-0.4 sec/step***.
- □ With no GPU-direct, significant reduction in network noise.
- □ ***390 GFLOPS/rank***
  → 28 PFLOPS total

July, 2023: ***No GPU-direct***



E. Merzari, S. Hamilton, T. Evans, P. Romano, P. Fischer, M. Min, S. Kerkemeier, Y.H. Lan, J. Fang, M. Phillips, T. Rathnayake, E. Biondo, K. Royston, N. Chalmers, and T. Warburton. Exascale multiphysics nuclear reactor simulations for advanced designs (Gordon Bell Prize Finalist paper). In Proc. of SC23: Int. Conf. for High Performance Computing, Networking, Storage and Analysis. IEEE, 2023.

Summary: Scaling CFD to Exascale and Beyond

Most important ideas:

- ❑ Runtime selection of kernels and communication strategies
  - ❑ Libraries and MPI could follow the same strategy?

- ❑ FP32 (or lower), where possible
  - ❑ Gating issue is ill-conditioning of the governing systems

- ❑ Efficient discretizations
  - ❑ Minimal number of unknowns for given accuracy
  - ❑ Minimal number of memory accesses per unknown

- ❑ Know your hardware :)
  - ❑ Leading edge HPC systems are finicky

Summary: Scaling CFD to Exascale and Beyond

Possible avenues for improved scaling:

- ❑ Extreme kernel fusion (a la John Camier…)
- ❑ Hardware-supported all-reduce (a la BG/L, BG/P, BG/Q)