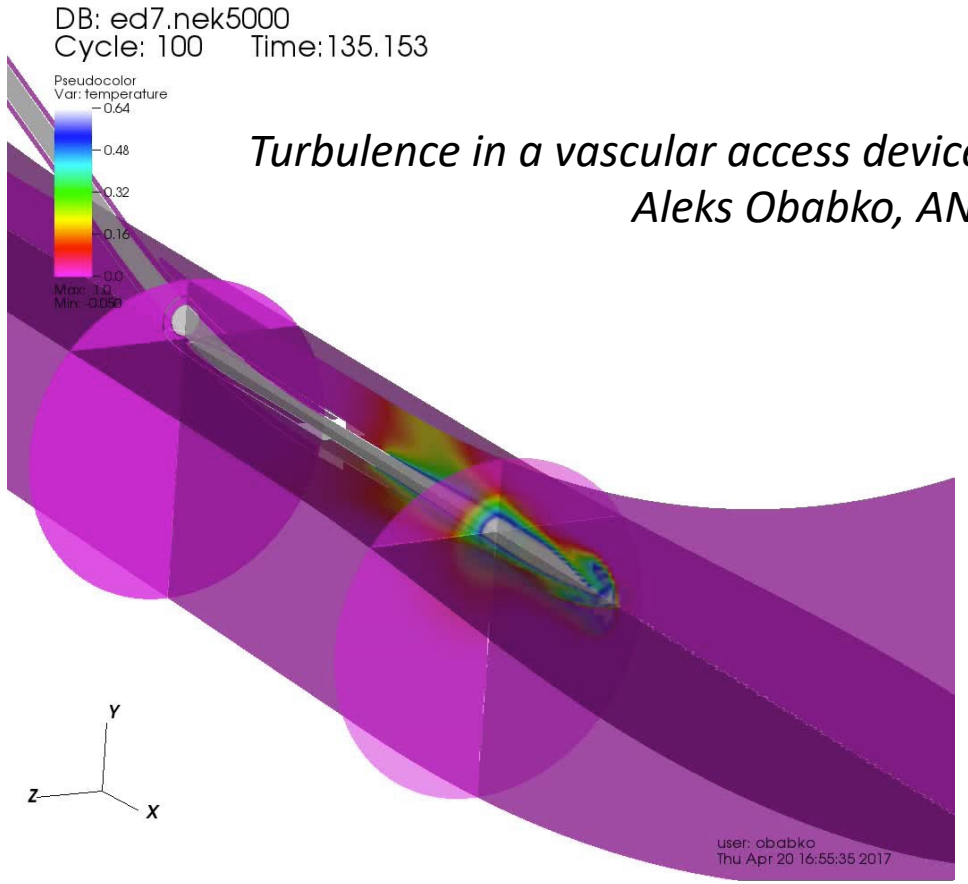
Exascale: A User's Perspective

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

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



Turbulence in a vascular access device.
Aleks Obabko, ANL

Supported by the Center for Efficient Exascale Discretizations (CEED) as part of DOE's Exascale Computing Project.

Example: Compressed Turbulence (Nek5000-CPU)

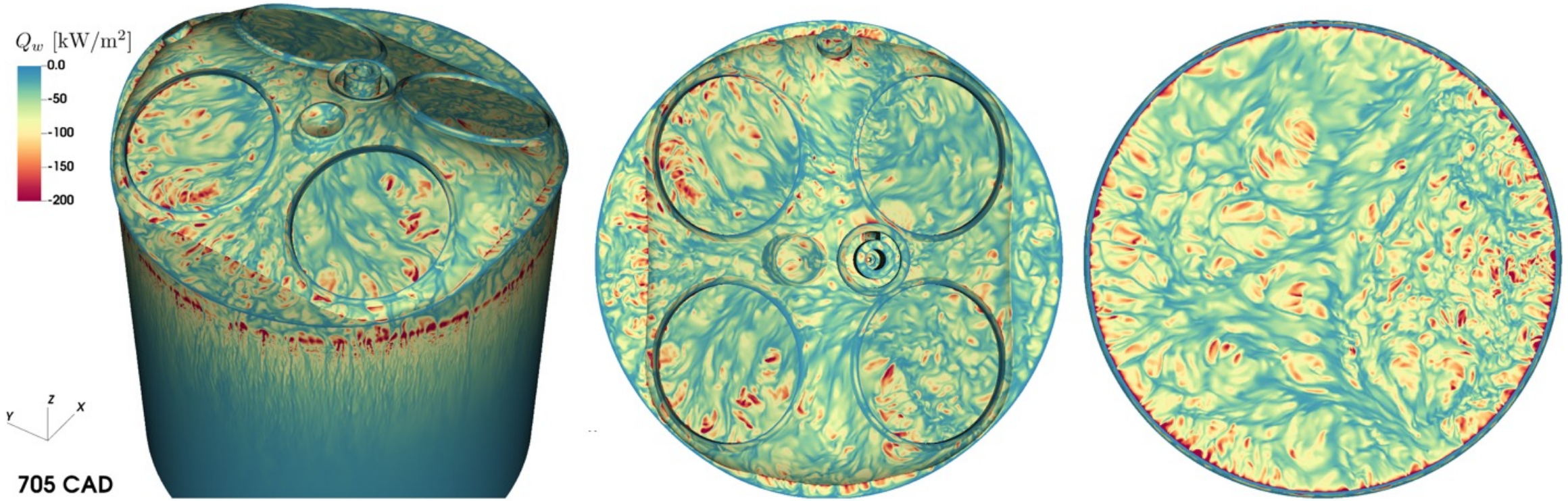
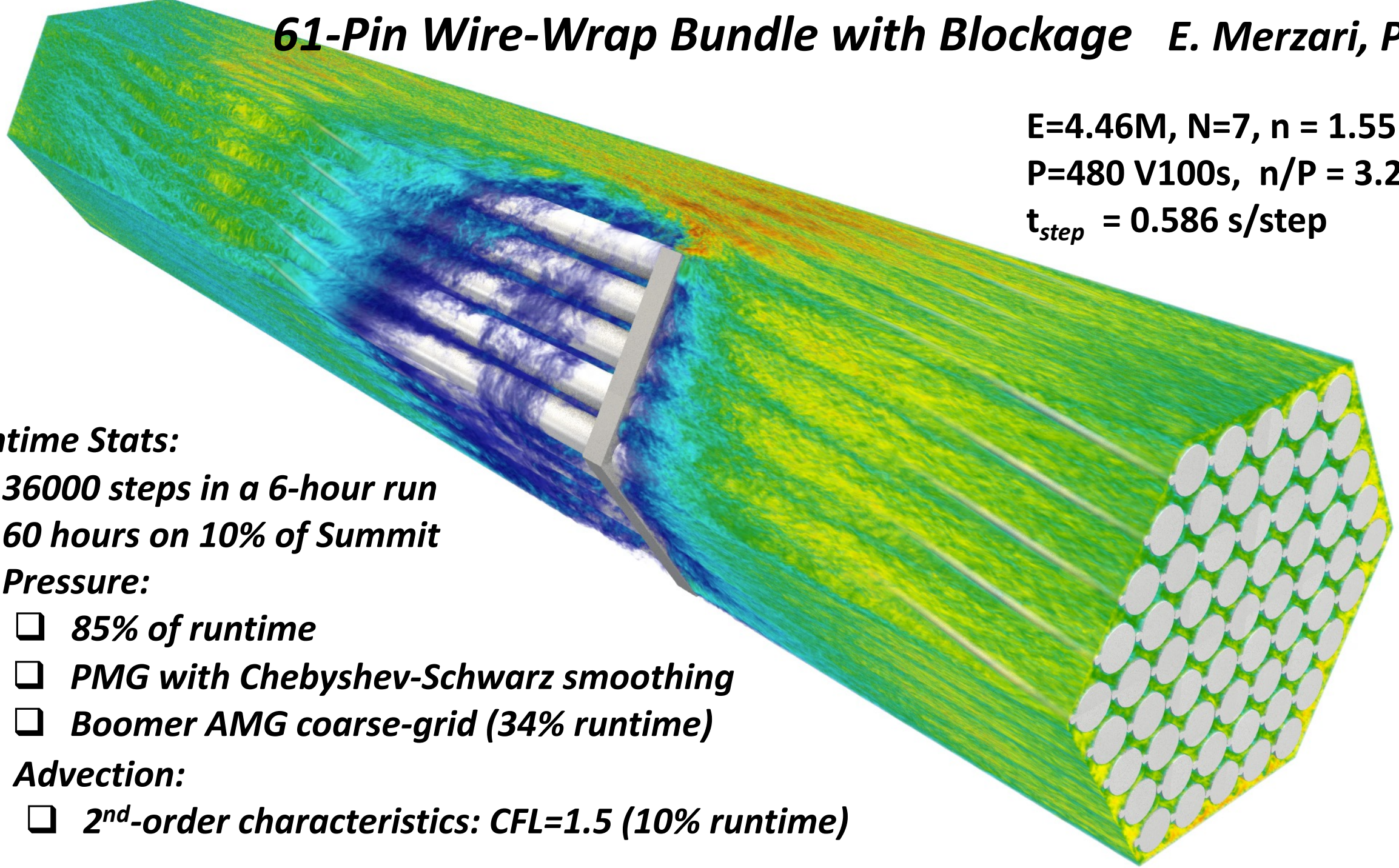


FIGURE 2.9 DNS of compression in an optical engine. Iso-contours of heat flux along the cylinder walls at 15^o bTDC, left-to-right: bird's eye view, cylinder head, piston.

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.*

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)***

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)

Summit: ***NekRS***
P=528 ranks (528 V100s)
n/P = 2.05M
0.146 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

Summary:

At strong-scale limit (80% eff.)

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

(This result not a foregone conclusion…2020 BP Paper.)

Objective:

- ***Develop a fast, efficient, scalable code for simulating turbulence in complex domains on exascale platforms.***

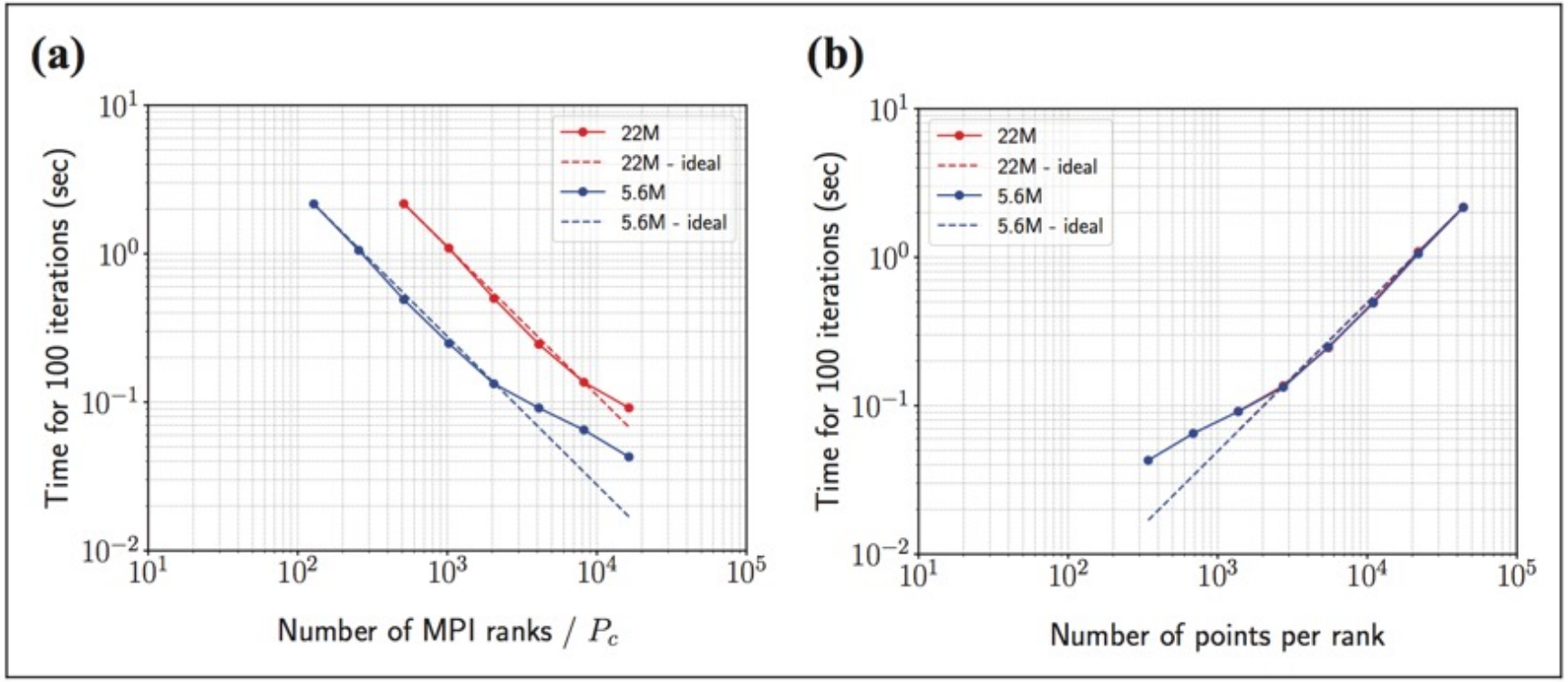
- ***Approach:***
  - ***High-order spectral element discretizations:***
    - ***reduced n for fixed Re, or***
    - ***increased Re for fixed n*** ***[costs scale as O(n)]***
  - ***Fast GPU kernels on each node*** ***[Warburton group, CEED]***
  - ***Strong-scale as far as possible***
    - ***largest possible P for fixed n → n/P as small as possible***
    - ***design to minimize communication, coarse-grid solve overhead***

Nek5000/CEM/RS Design Principles

- Efficient high-order discretizations
  - Minimize number of gridpoints and data movement for a given engineering-level accuracy
  - Cost per gridpoint equivalent to or lower than low-order discretizations
  - Fast implementations on each node (particular focus on GPUs for NekRS)
  - **Strongly scalable**, i.e., use as many nodes as possible for reasonable efficiency (e.g., parallel efficiency $\eta \sim 80\%$) — ***important for speed.***

- Support broad range of physics needed for engineering applications
  - Large number of boundary conditions
  - Turbulence models (LES, uRANS of multiple flavors)
  - Compressible flow (low-Mach), combustion
  - Complex meshes, nonconforming meshes, moving geometries (ALE)
  - State-of-the-art Lagrangian particle tracking with 1-way, 2-way, and 4-way [Zwick&Balachandar '19]
  - Maxwell, Poisson-Nernst-Planck (e.g., ion channels, molten salt)

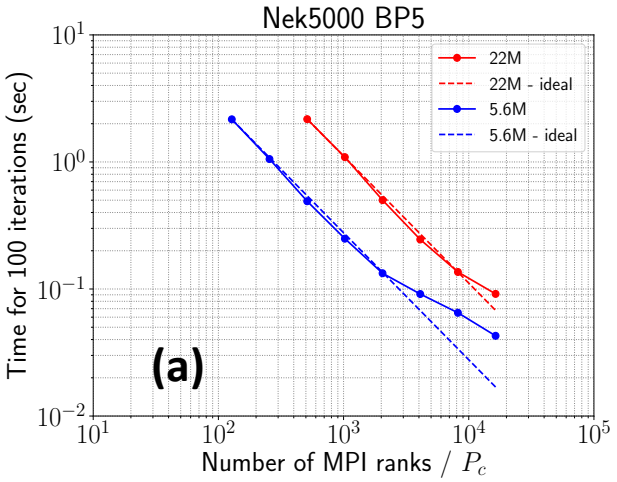
Parallelism: Stong-Scaling, Time to Solution, and Energy Consumption

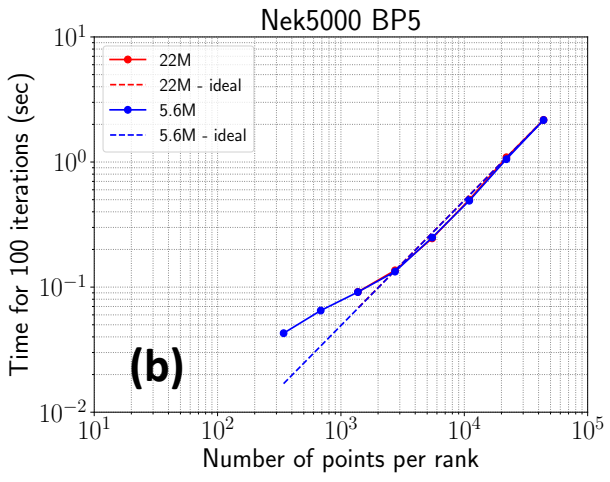


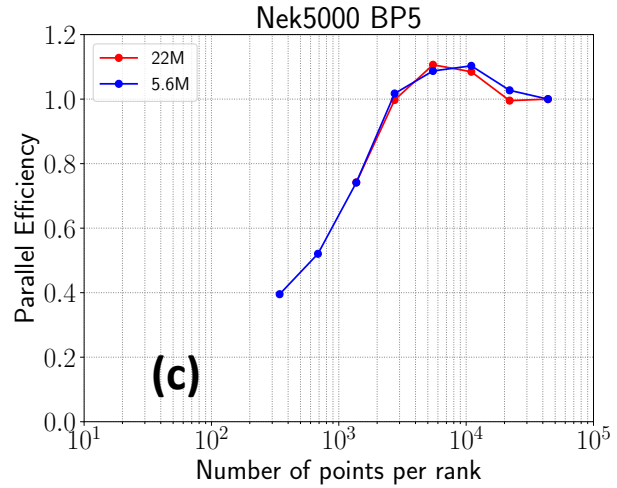
Observations:

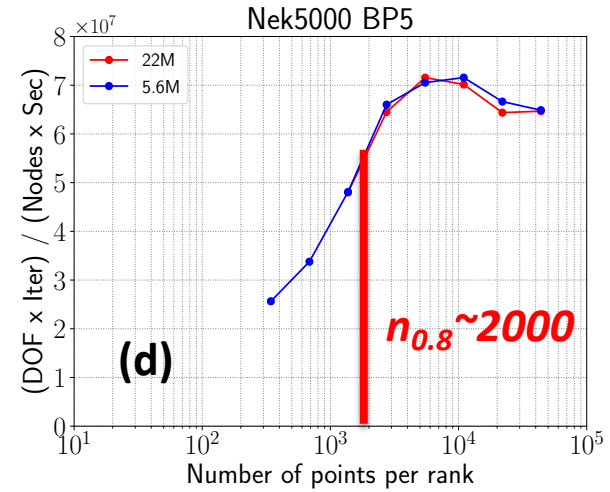
1. Time-to-solution goes down with increasing P, particularly for $\eta = 1$.
2. For $\eta = 1$, energy consumption ~ P x t_{sol} = *constant* — no penalty for increased P.
3. The red curve can use more processors than the blue. ***WHY?***
4. Why (for a problem of any size), do we find $\eta < 1$?
 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

Same Data, Strong-Scale vs. P or n/P and Efficiency or MDOFS vs n/P









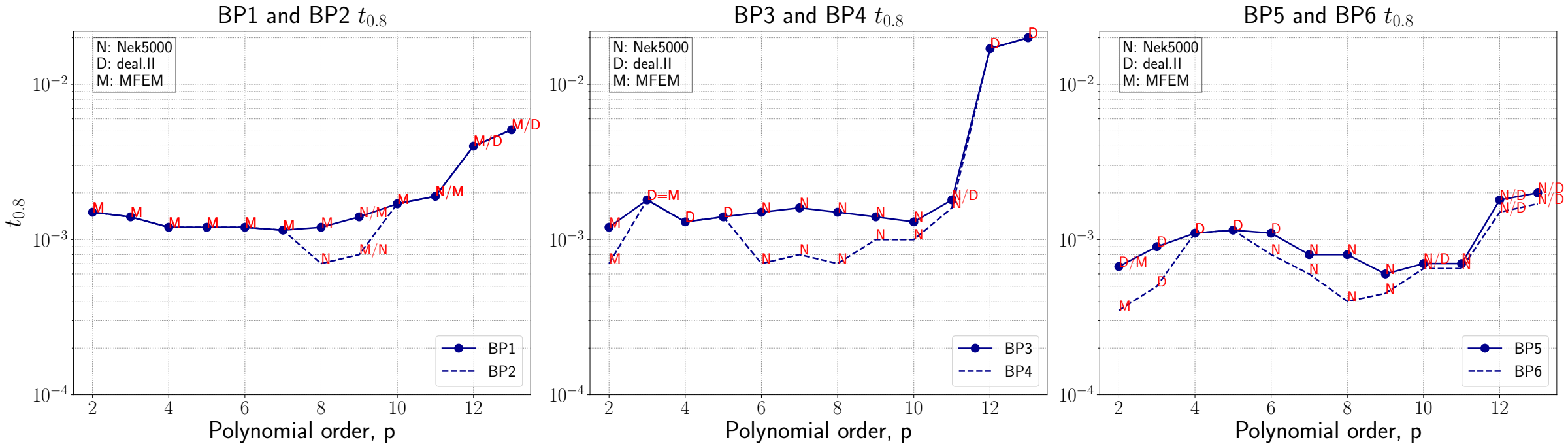
- Plotting MDOFS vs. n/P is more universal than time vs. P:
  - Allows code-to-code comparison.
  - Identifies $n_{0.8}$, the local problem size, (n/P), that realizes $\eta = 0.8$.

- Why do we care about $n_{0.8}$?
  - Min time-to-solution (at $\eta = 0.8$) is

$$t_{0.8} = \frac{W\, n_{0.8}}{MDOFS}$$

- HPC users run at this point
  - Design for performance at this point
  - Analyze system-level approaches to reducing $n_{0.8}$

BP1/3/5 Bake-Off Problems on BG/Q: BP5=SEM Poisson Solve



- Times are the minimum-time-per iteration at $\eta = 0.8$
  - Idea of the bake-off is to compare performance of multiple codes (here, Nek, MFEM, and deal.ii) over a large range of runtime parameters: polynomial order, number of elements, problem types, and to resolve any major discrepancies, i.e., to help all teams realize the fastest possible performance.
  - For BP5, the SEM Poisson solve, Nek5000 is the fastest (has the lowest $t_{0.8}$) for its target operating range of N=7—13.

Outline:

0. *Introduction (preceding slides)*

1. *Mathematical Background*
   - ❑ *Discretization - SEM*
   - ❑ *Navier-Stokes timestepping*
   - ❑ *Pressure Poisson Problem & Preconditioning*

2. *Parallel Computing Concerns*
   - ❑ *Scalability ⇐⇒ Speed*

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

Influence of Scaling on Discretization

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 $t_{\text{final}} \approx L/\lambda$.

- Dispersion errors accumulate linearly with time:

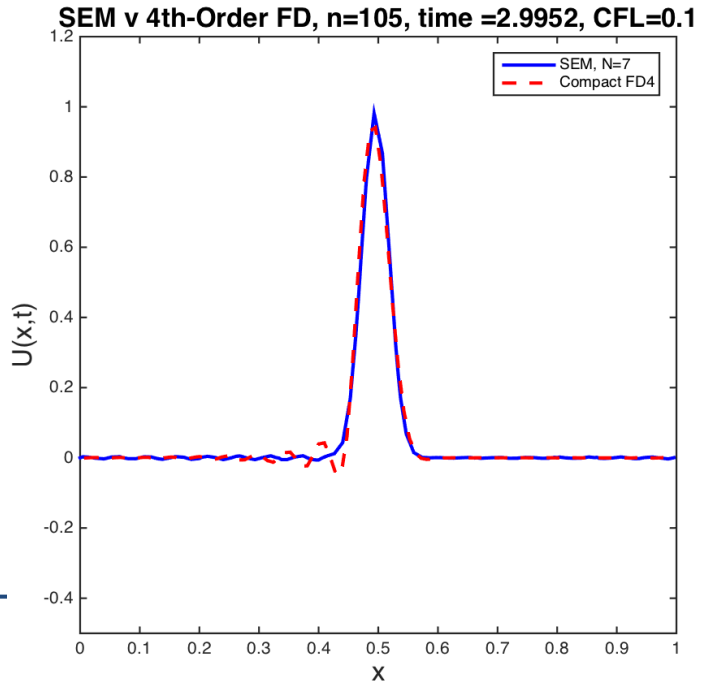
  error $\sim$ **|correct speed - numerical speed|** $* t$ (for each wavenumber, k)

  error$(t_{\text{final}}) \sim (L/\lambda) *$ **|numerical dispersion error|**

- For fixed final error, ϵ_f, require:

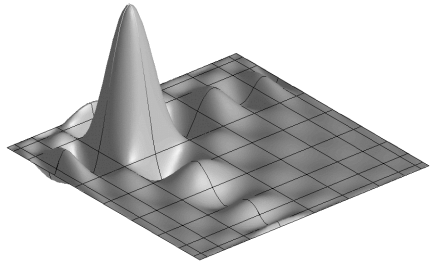
  numerical dispersion error $\sim (\lambda/L)\epsilon_f \ll 1$.

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

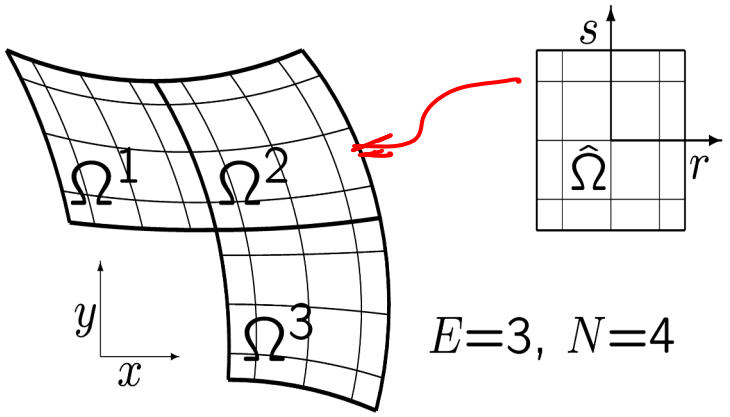


Spectral Element Method — A. Patera '84

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



2D basis function, N=10



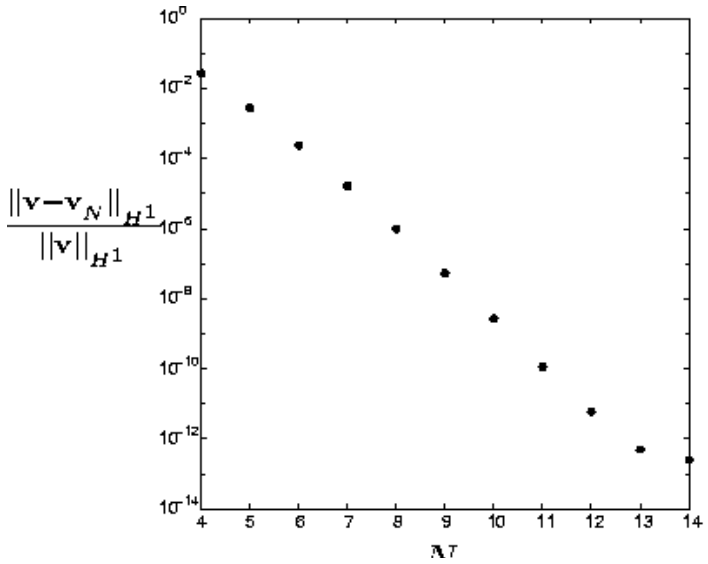
$E=3,\ N=4$

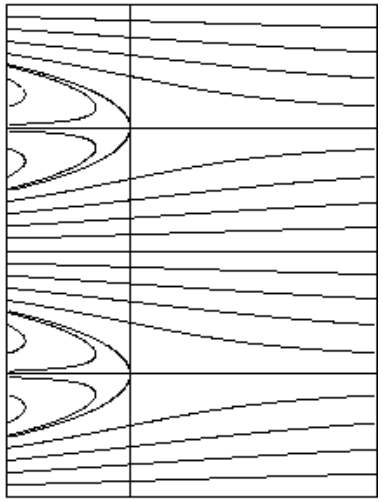
Spectral Element Convergence: Exponential with N

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

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

Exact Navier-Stokes Solution (Kovazsnay '48)



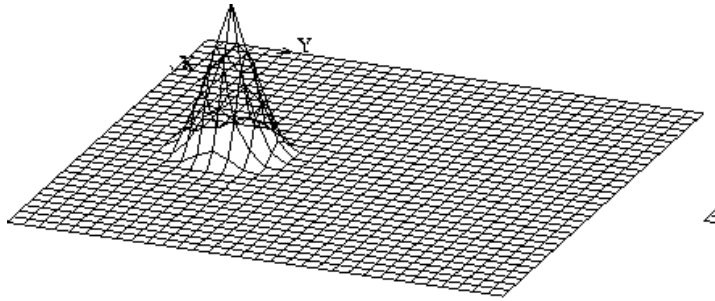


$$
\begin{aligned}
v_x &= 1 - e^{\lambda x}\cos 2\pi y \\
v_y &= \frac{\lambda}{2\pi} e^{\lambda x}\sin 2\pi y \\
\lambda &:= \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}
\end{aligned}
$$

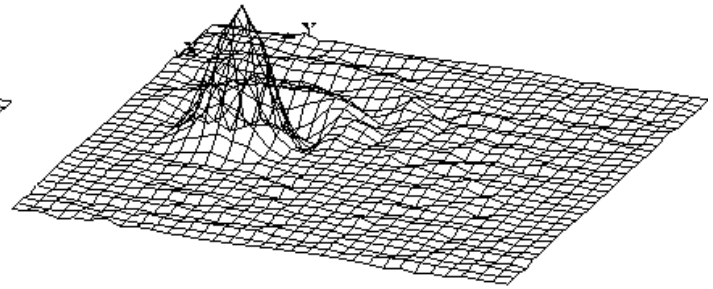
Excellent Transport Properties, even for Nonsmooth Solutions

- *Convection of nonsmooth data on. a 32x32 grid*
- *K1 x K1 spectral elements of order N*

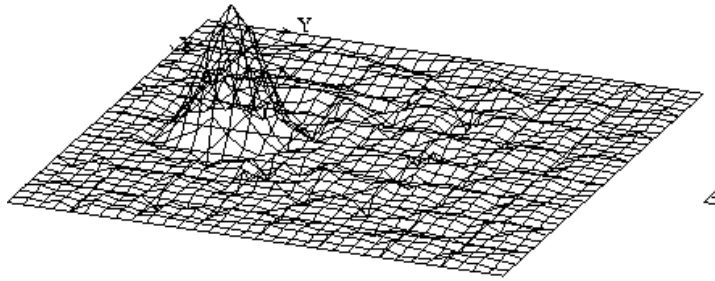
(cf. Gottlieb & Orszag, Spectral Methods, 1977)



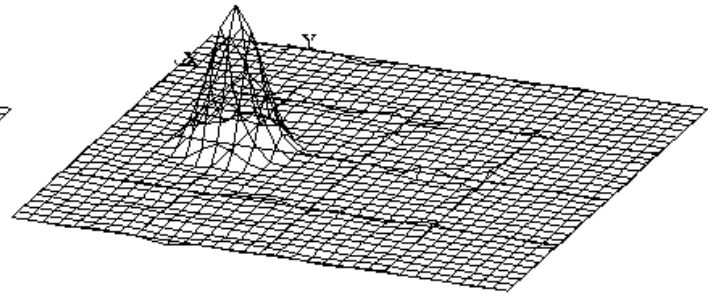
Initial Condition



$K_1 = 16,\ N = 2$



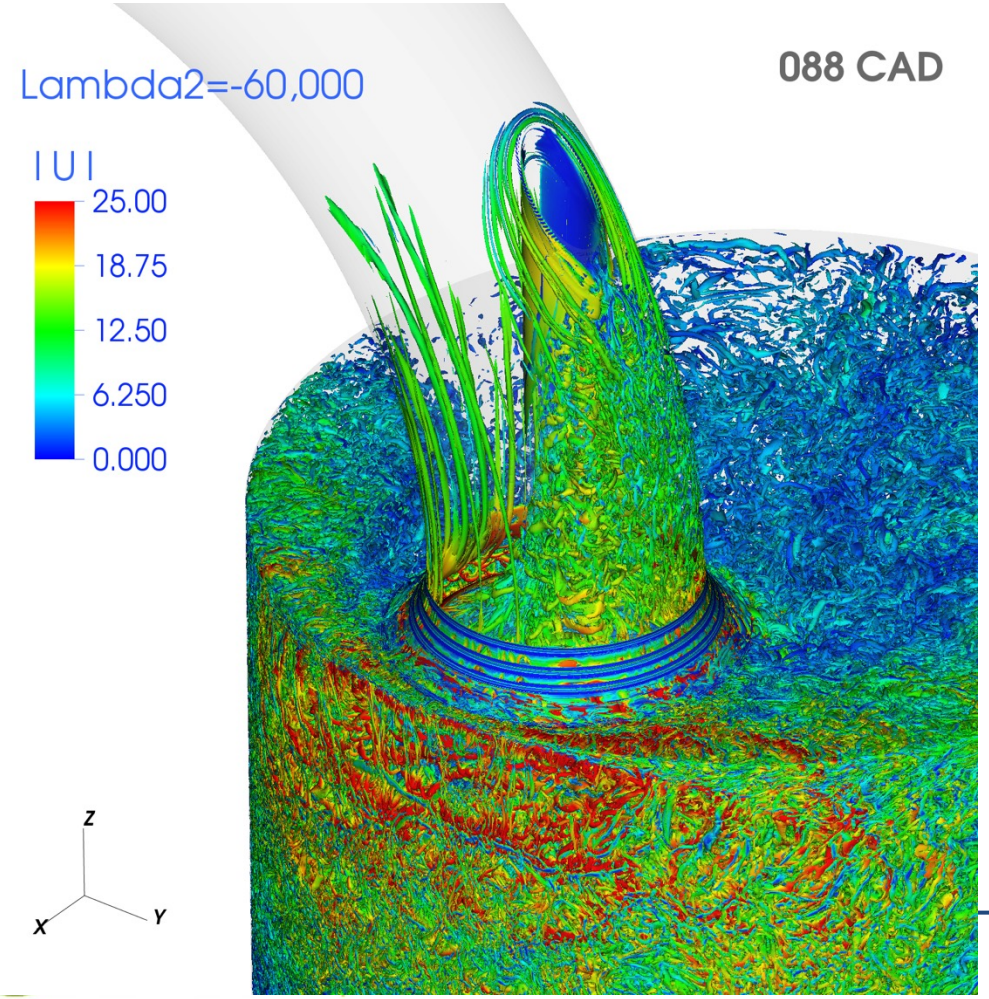
$K_1 = 8,\ N = 4$

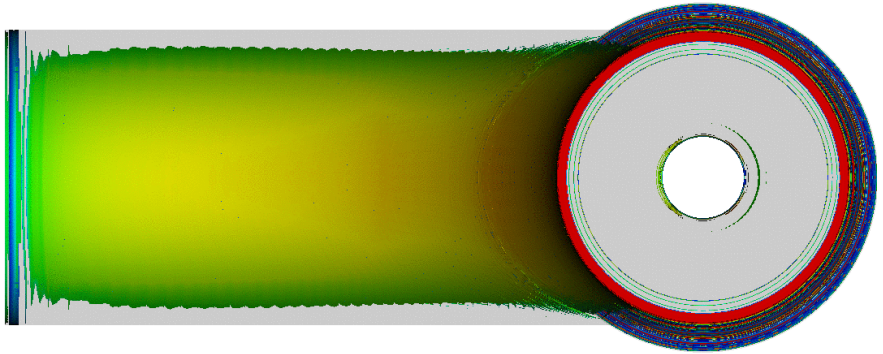


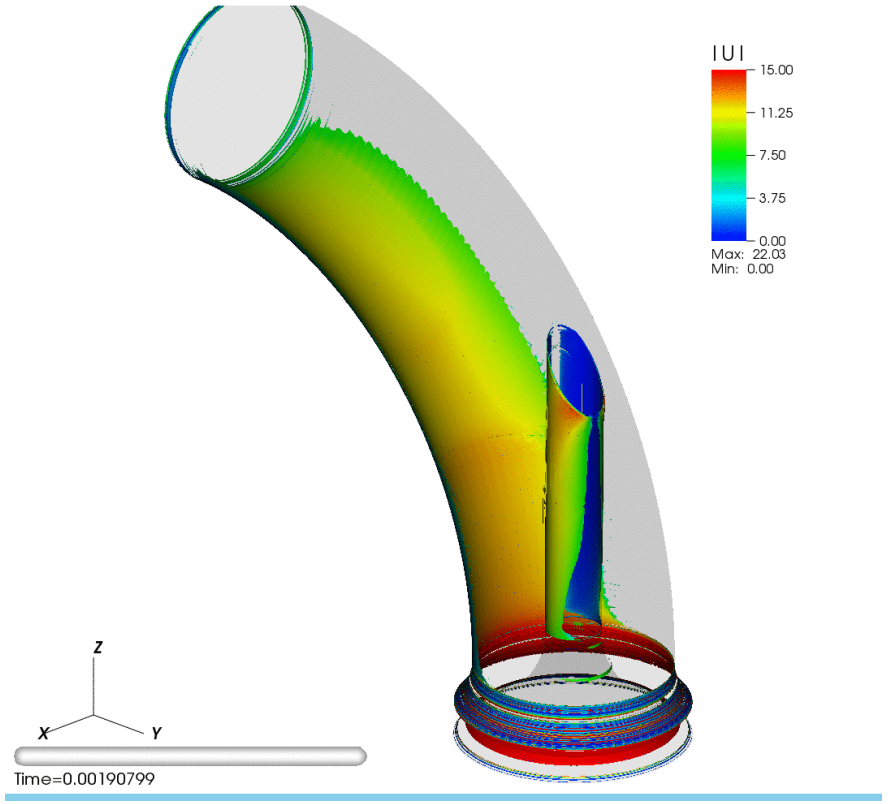
$K_1 = 4,\ N = 8$

High-Order is Efficient for Tracking Small-Scale Structures

- Flow in a model IC engine
- Re=45,000

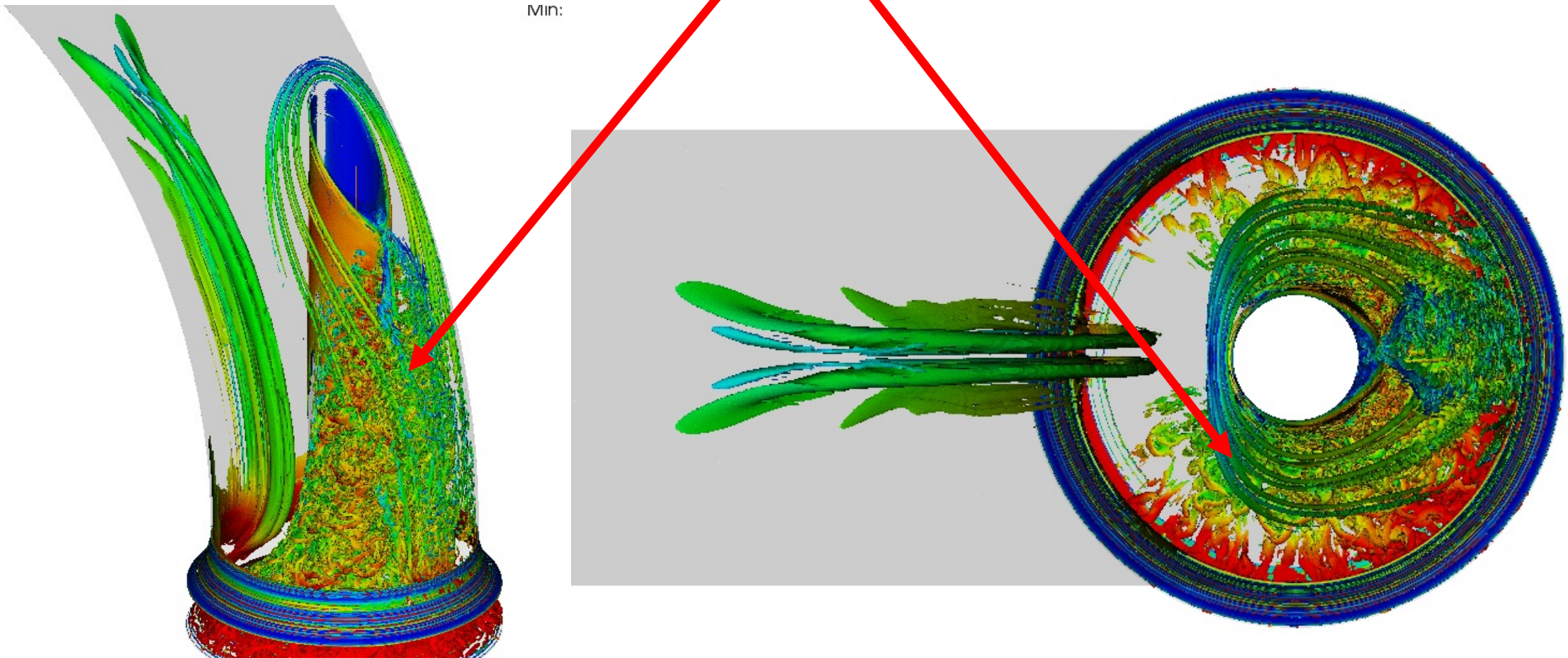






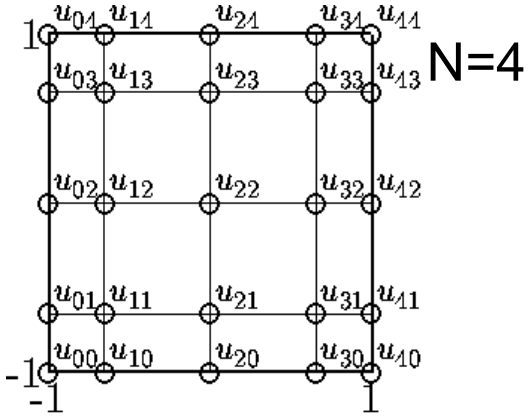
Vortex Breakdown at Re_D = 45,000

- *These are well-resolved calculations performed on ANL's Theta*
- *Note the fine filamental horseshoe vortices around the base of the valve stem that ultimately breaks down into a chain of hairpin vortices*
- *Although turbulent, the flow is not random!*



Costs

- **Cost dominated by iterative solver costs, *proportional to***
  - iteration count
  - matrix-vector product + preconditioner cost



- **Locally-structured tensor-product forms:**
  - minimal indirect addressing
  - ***fast matrix-free operator evaluation***
  - low-cost local operator inversion via fast diagonalization method *(Lynch et al.'64)*

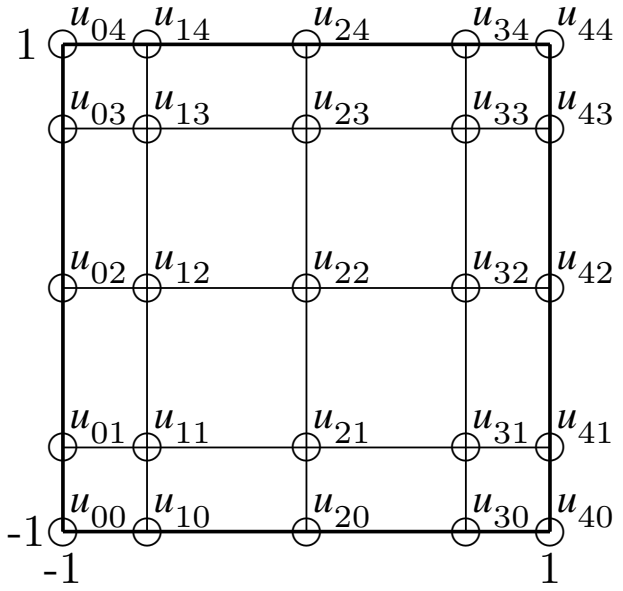
Fast Operator Evaluation: Matrix-Matrix Products / Tensor Contractions

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

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

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

$$= \sum_{p} \hat{D}_{ip} u_{pj} \qquad \begin{cases} \text{matrix-matrix product} \\ \text{tensor contraction} \end{cases}$$



- $2N^2$ reads, $2N^3$ ops
- 3D: N^3 reads ($\underline{u}$), $2N^4$ ops
- 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.

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*

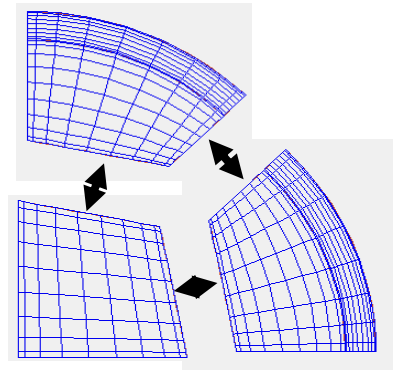
- Spectral element coefficients stored in local ($\underline{u}_L$) form, not global ($\underline{u}$)
- Example: *Application of SEM Laplacian*

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

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

local work (matrix-matrix products)

nearest-neighbor (gather-scatter) exchange



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

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

Matrix free form :
- $7N^3$ memory ref.
- $12N^4 + 15N^3$ ops.

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

Majority of ops. *Majority of memory refs.*

Impact of Order on Costs

- To leading order, cost scales as number of gridpoints, regardless of approximation order.

- Consider Jacobi PCG as an example:

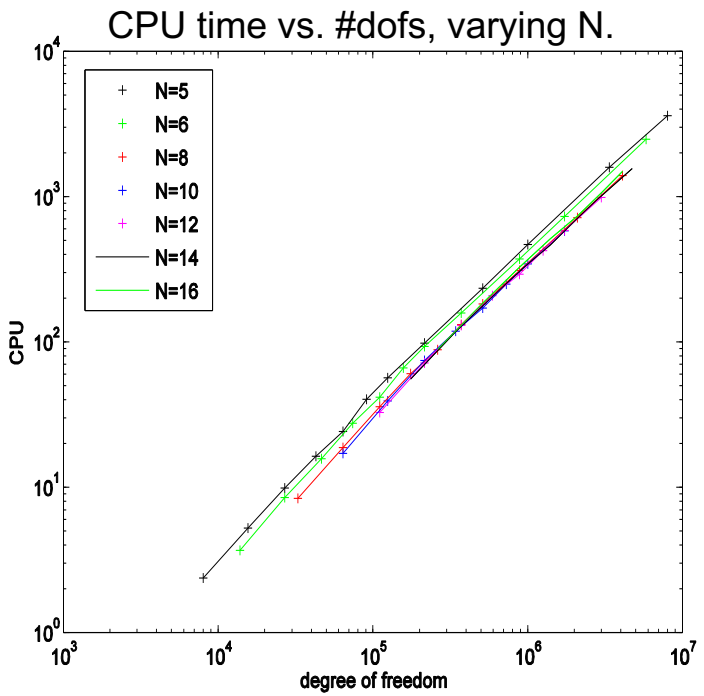
$$\underline{z} = D^{-1}\,\underline{r}$$
$$\underline{r} = \underline{r}^{t}\,\underline{z}$$
$$\underline{p} = z + \beta\,\underline{p}$$
$$\underline{w} = A\,\underline{p}$$
$$\sigma = \underline{w}^{t}\,\underline{p}$$
$$\underline{x} = \underline{x} + \alpha\,\underline{p}$$
$$\underline{r} = \underline{r} - \alpha\,\underline{p}$$

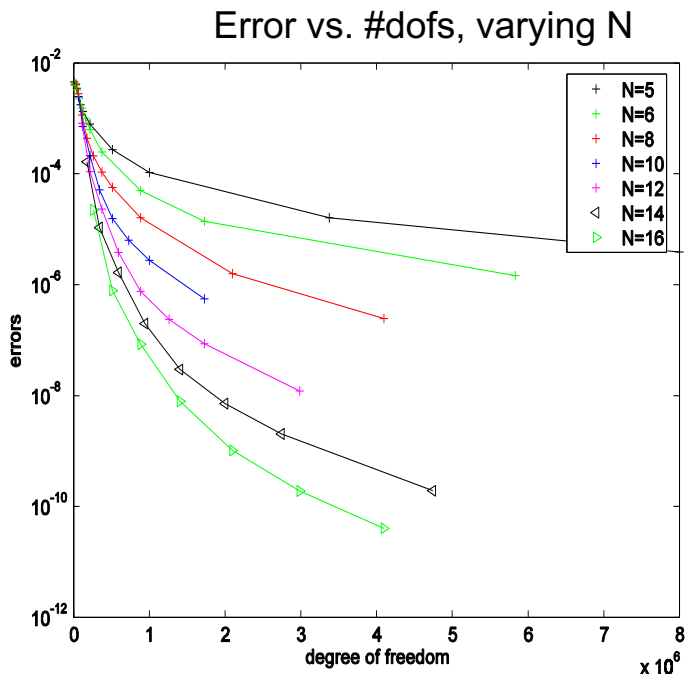
- Only one operation depends on order—the remaining, memory-bound, depend on number of gridpoints, n.
  - Reducing n is an effective way to reduce data movement.

- For incompressible Navier-Stokes, however, preconditioning is the dominant cost.

Runtime is Weakly Dependent on Polynomial Order, N

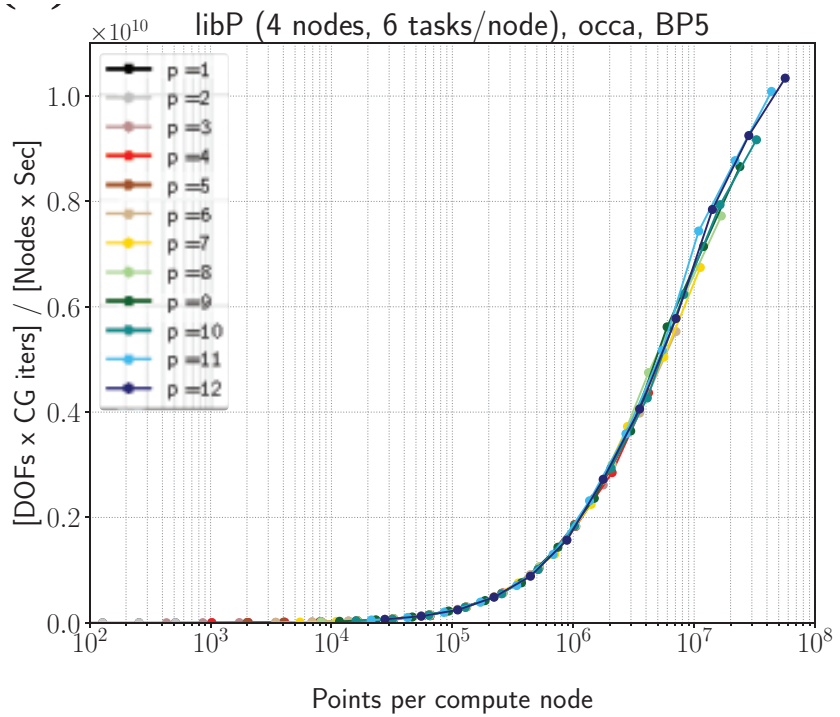
Electromagnetics Example: NekCEM (M. Min)





Periodic Box; 32 nodes, each with a 2.4 GHz Pentium Xeon

Jacobi PCG-BP5 (T. Warburton)



24 NVIDIA V100s

Main Conclusion: *properly implemented high-order methods are not more expensive per DOF than their low-order counterparts*

Stability

We address stability of the advection operator in several ways:

- *Filtering:*
  - *Low-pass filter, F, as a post-processing step, or*
  - *High-pass filter (HPF) on the rhs of the Navier-Stokes equations*
- *Dealiased advection operator*
- *As an alternative, we could consider DG + upwinding for the hyperbolic substep, as is done in MFEM/LAGHOS, libP, and several of the more recent high-order codes from the deal.ii group and others in Europe.*
- *I'll discuss this topic off-line with anyone who is interested.*

Choice of Quadrature for Advection-Diffusion Terms

$$(v, u_t)_L = -(v, \mathbf{c} \cdot \nabla u)_M + \nu(\nabla v, \nabla u)_N$$

Accuracy — Stability — Stability

- Let $(v, u)_N$ denote the discrete L^2 inner product using Gauss-Lobatto-Legendre quadrature on $N + 1$ points in each direction of the reference domain, $\hat{\Omega} := [-1, 1]^d$.
- Let $(v, u)_M$ denote the discrete L^2 inner product using Gauss-Legendre quadrature on M points in each direction.
- Let $(v, u)_L$ denote the discrete L^2 inner product using Gauss-Legendre quadrature on M points in each direction.

If we take $L = N$, we get a diagonal mass matrix—inversion cost is effectively zero.

We need $M > N$ to guarantee that $(v, \mathbf{c} \cdot \nabla u)$ is skew-symmetric (hence, stable), but if were using DG+upwind we might not need dealiasing.

Impact of Inexact Quadrature on the u_t Term — Ainsworth '14

MARK AINSWORTH

Degree	Centred DG	Finite Element	Spectral Element
1	$\frac{i\Omega^3}{48}$	$\frac{i\Omega^5}{180}$	$\frac{i\Omega^3}{6}$
2	$-\frac{i\Omega^7}{16800}$	$-\frac{i\Omega^5}{4320}$	$\frac{i\Omega^5}{1080}$
3	$\frac{i\Omega^7}{806400}$	$\frac{i\Omega^9}{3175200}$	$\frac{i\Omega^7}{75600}$
4	$-\frac{i\Omega^{11}}{1005903360}$	$-\frac{i\Omega^9}{254016000}$	$\frac{i\Omega^9}{31752000}$
5	$\frac{i\Omega^{11}}{120708403200}$	$\frac{i\Omega^{13}}{479480601600}$	$\frac{i\Omega^{11}}{838252800}$

TABLE 1. Leading terms for the relative error in the approximation of the Floquet multiplier for Centred Discontinuous Galerkin, Finite Element and Spectral Element schemes applied to (1)

Our Strategy: *Use inexpensive diagonal mass matrix. If you want more accuracy, increase N. (Maday & Ronquist, 90)*

Incompressible Navier-Stokes and Thermal Transport (P_N-P_N)

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

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

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T = \frac{1}{Pe}\nabla^2 T + q(\mathbf{x})$$

- kth-Order Time Advancement (k=1, 2, or 3)
  - Advection step: $\hat{\mathbf{u}} = \mathbf{advect}(\mathbf{u}^{n-1},\ldots,\mathbf{u}^{n-k})$
  - Pressure correction: $\hat{\hat{\mathbf{u}}} = \hat{\mathbf{u}} - \Delta t\nabla p^n$, where p^n satisfies

$$-\nabla^2 p^n = -\frac{1}{\Delta t}\nabla\cdot\hat{\mathbf{u}}$$

$$\frac{\partial p^n}{\partial n} = \frac{\mathbf{n}\cdot\hat{\mathbf{u}}}{\Delta t} - \frac{1}{Re}\mathbf{n}\cdot\nabla\times\nabla\times\left(\sum_{j=1}^{k}\alpha_j\mathbf{u}^{n-j}\right) \text{ on } \partial\Omega.$$

  - Diffusion step: $\left[-\frac{\Delta t}{Re}\nabla^2\mathbf{u}^n + \beta_0\mathbf{u}^n\right] = \hat{\hat{\mathbf{u}}}$ (+ BCs).

- Pressure-Poisson step is intrinsically the stiffest substep—requires multilevel preconditioning.

- Diffusion step yields 3 diagonally-dominant Helmholtz problems that are readily solved with Jacobi-preconditioned CG.

Solvers

- Pressure Poisson solve – $A\underline{p}^n = \underline{b}^n$
  - Every timestep, conditioning not improved as $\Delta t \rightarrow 0$
  - Almost all Neumann conditions
  - Intrinsically the most expensive step for incompressible NS (fastest timescale).
  - Solving nearby problems on each step
    - leverage by projecting onto prior solution space
    - very often just a few iterations/step

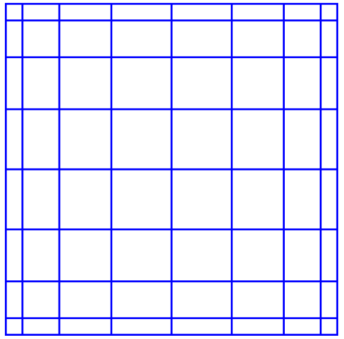
$$A\delta\underline{p}^n = \underline{b}^n - A\bar{\underline{p}}^n,$$

$$\bar{\underline{p}}^n := \sum_j \underline{x}_j \underline{x}_j^T \underline{b}^n, \quad \underline{x}_j^T A \underline{x}_i = \delta_{ij}$$

$$\underline{x}_j \in \mathrm{span}\{\underline{p}^{n-1} \ldots \underline{p}^{n-k}\}$$

What Makes These SEM Problems Different?

- Unstructured, high-aspect-ratio elements (e.g., boundary layer elements)
- High-aspect-ratio subcells because of tensor-product Gauss-Lobatto-Legendre (GLL) nodes
- Many different geometric configurations – no single solver is best for all cases.



- Interested in solving problems ***at the strong-scale limit***.

- Coarse-grid solve costs do not go to zero as 1/P:

$$\frac{T_{cMG}}{T_{aMG}} = \frac{8\,\alpha\,\log_2(n/P) + 30\,\beta\,(n/P)^{\frac{2}{3}} + 8\,\alpha\,\log_2 P}{50n/P}$$

[1]*Fischer, Heisey, Min, Scaling Limits for PDE-Based Simulation, AIAA 2015*

Consequences of Strong-Scaling

- Suppose wall-clock time complexity is: $t_{wc} = W/P + C = 80 + 20$ seconds
- If an improved algorithm leads to C' = 5 seconds, it appears we have a 15 second gain.
- Not quite...
  - Users will increase P to P' = 4P (same efficiency) to yield a runtime of

$$t'_{wc} = W/4P + C' = 25 \text{ seconds}$$

- Same energy and node-hour consumption!
- 4X Faster time to solution.

Possible Solution Strategies

- Krylov Subspace Projection (KSP): mandatory for robustness
  - CG if preconditioner is SPD
  - flexCG
  - GMRES (if # iterations per step is small)

- Must be multilevel because of the problem scale: $n = 10^7 – 10^{11}$

- Preconditioners:
  - Overlapping Schwarz
  - p-Multigrid (PMG): variety of smoothers
  - FEM-SEM equivalence: sparse low-order operator, $A_{FEM} \sim A_{SEM}$
    – requires scalable AMG

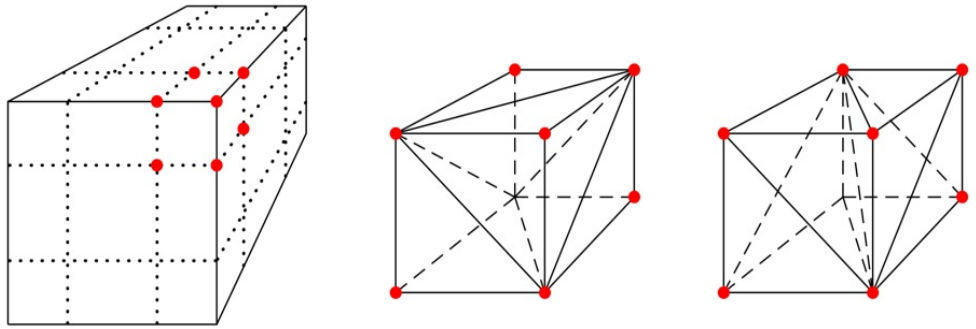
Some PMG Smoothers

- Jacobi: suffers from subcell aspect ratios in d>1 space dimensions (Ronquist'88)
- Line smoothers (Heinrichs '88)
- Block-Smoothers via fast-diagonalization method (Pahl 93, Couzy 95)
- Multilevel Weighted Additive Schwarz–low cost but high iteration counts (F97,Lottes & F 05)
- Chebyshev-Jacobi (Adams et al.'03, Gandham'15, Karakus et al.'19)
- Chebyshev-Schwarz (Phillips'22)

On platforms where communication costs are high, there is an advantage to having better smoothers so that the coarse-grid solve is invoked less frequently.

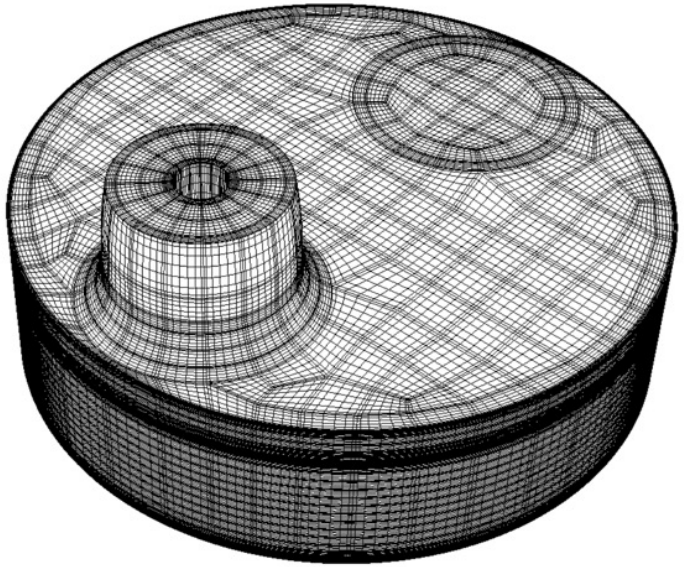
Robust Preconditioner: FEM-SEM + AMG

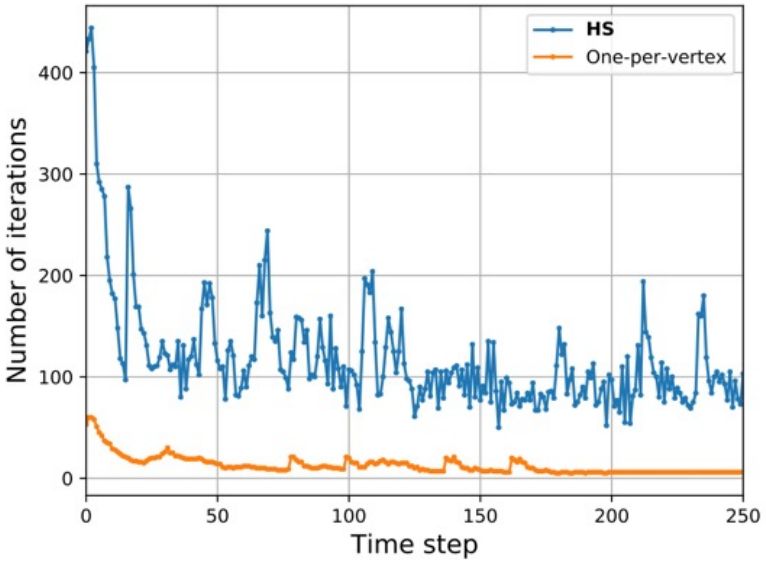
- Orszag (JCP 1980) pointed out that, for Poisson, sparse low-order finite difference operators were spectrally equivalent to their spectral counterparts on the same nodal point distributions, with a bounded condition number of $\pi^2/4$
  - Deville & Mund; Canuto & Quarteroni; F. et al.,Parter et al. extended this to FEM discretizations.
  - More recent developments in Canuto et al. ‘10, Bello-Maldonado & F. ’19, Pazner ‘19.
- Main idea:
  - Tesselate GLL points with tets – ***choice of tessellation is important***
  - Form sparse FEM A_{FEM} matrix
  - Use a single AMG V-cycle (via hypre or other scalable option)
- Observations:
  - Low iteration count
  - Cost per iteration is high, compared to ASM-pMG
  - A winner in some cases where ASM-pMG stalls
  - **Requires robust & fast AMG.**

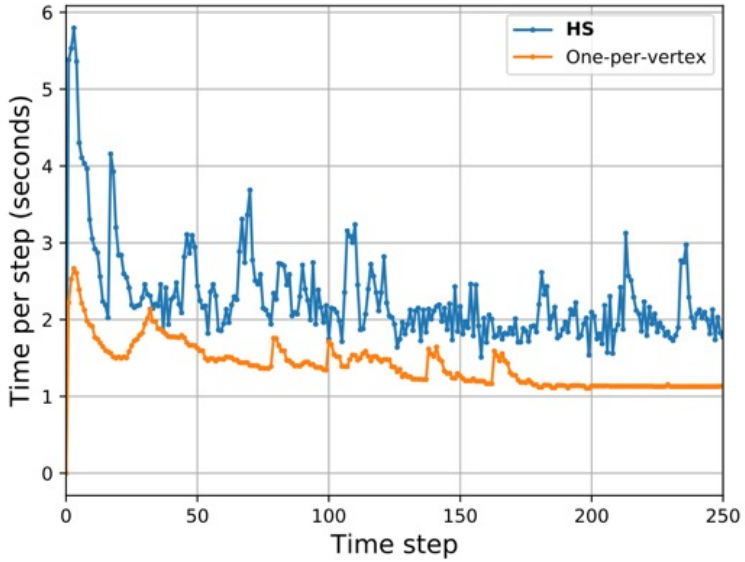


FEM vs HSMG Convergence

- Piston with moving valve example:
  - E=6784, N=7.
  - CFL=4
  - Significant reduction in iteration count
  - FEM preconditioning: ~60% reduction in solution time
  - Single FEM V-Cycle with Hypre as GMRES preconditioner
  - Timings on Mira (CPU)

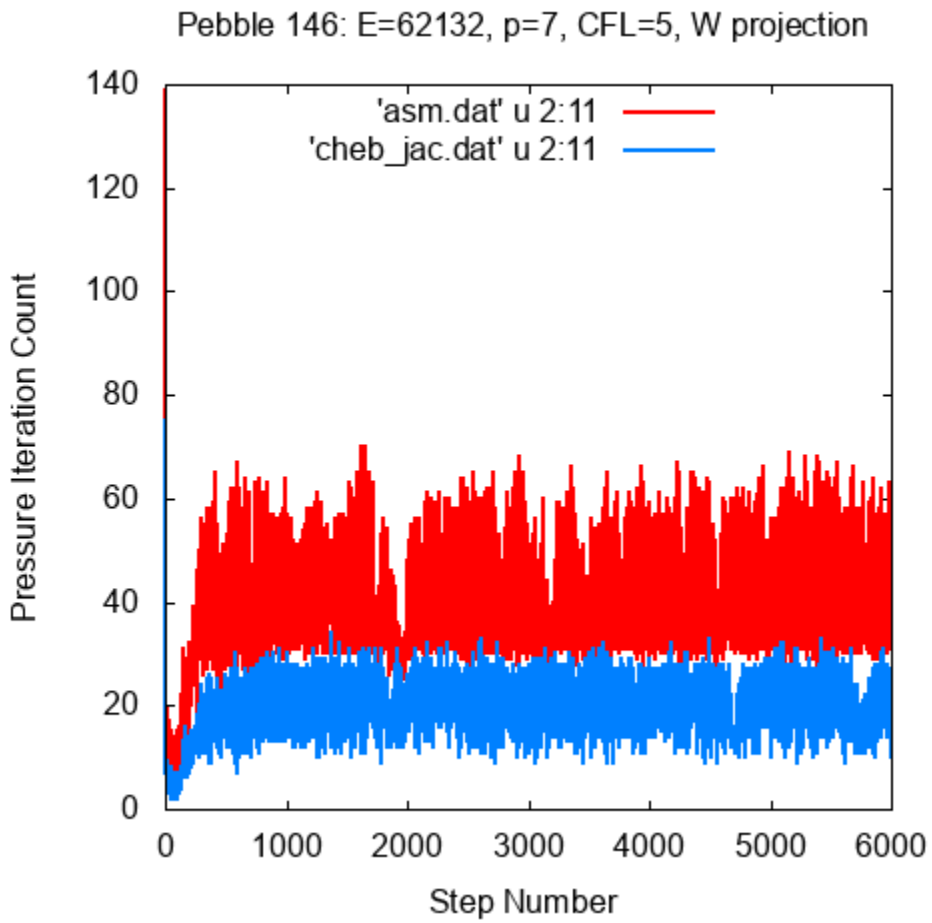


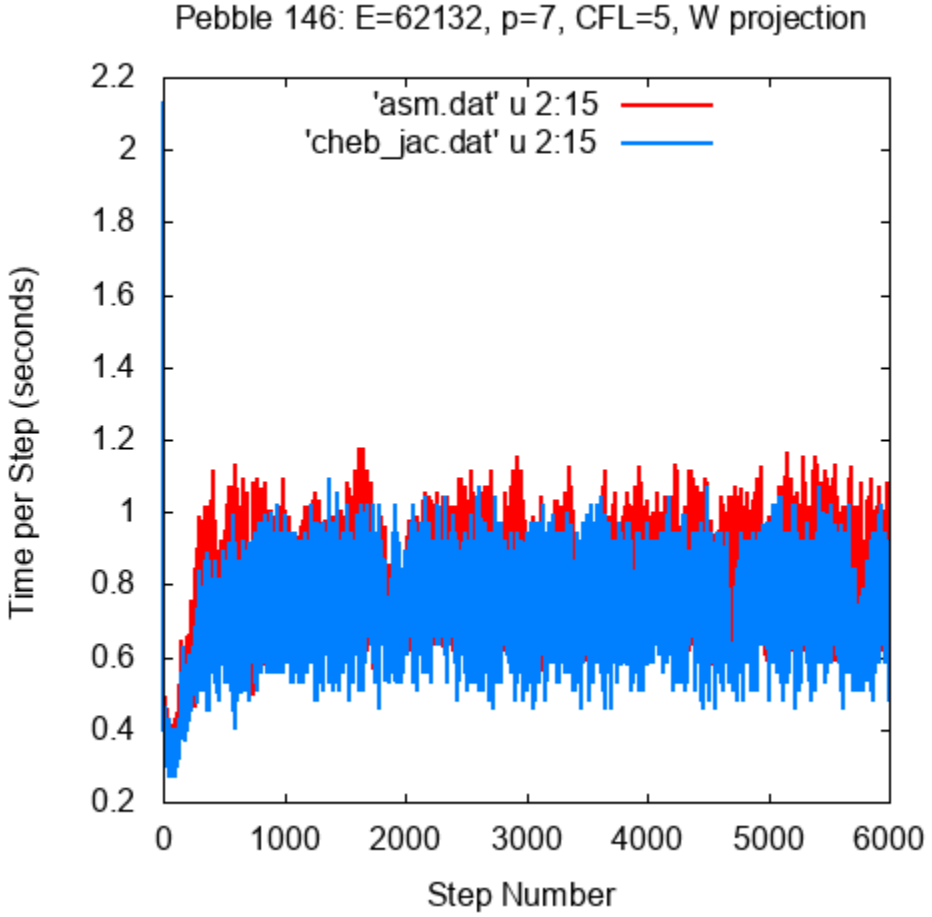




Recent Schwarz Developments

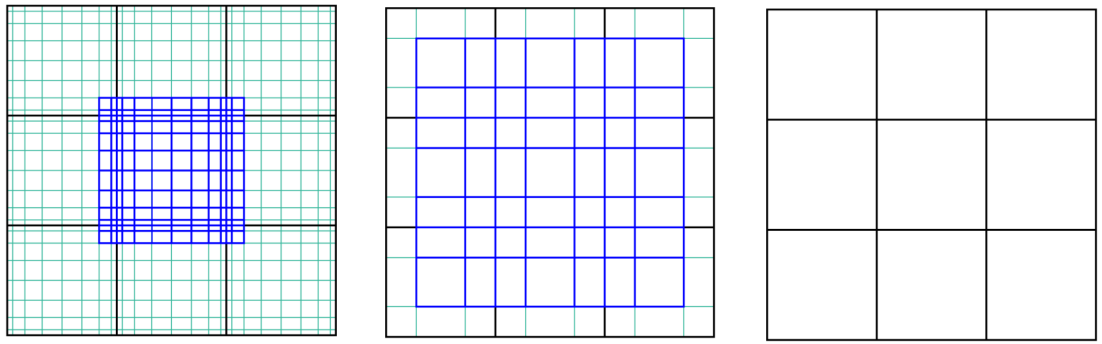
Malachi Phillips (UIUC)



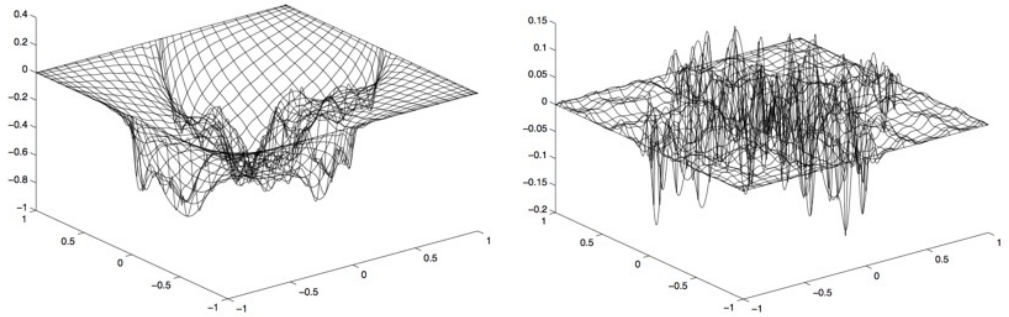


Schwarz v. Chebyshev Jacobi

- We know that weighted overlapping Schwarz is a good local smoother (and, not expensive).
- The additive variant puts more pressure on the coarse-grid solve than a multiplicative version with several rounds of smoothing at each level.
- Chebyshev-Jacobi is remarkably robust.
- A natural idea is to apply Chebyshev acceleration to the Schwarz smoothing cycle.



N=7 N=3 N=1



(e) Application of WM_{Schwarz} (f) Coarse solve after (e)

Figure 3: Error plots for the hybrid Schwarz preconditioner and coarse solve, with $N_C = N/2$ and $(E, N) = (4, 16)$, applied to a random initial guess.

Local Schwarz smoothing is fast via fast-diagonalization

$$M\underline{z} = \underline{r} \implies \underline{z} = \sum_{e=1}^{E} R_e^T A_e^{-1} R_e \underline{r},$$

$$A_e = R_e A R_e^T \approx [B_s^e \otimes A_r^e + A_s^e \otimes B_r^e]$$

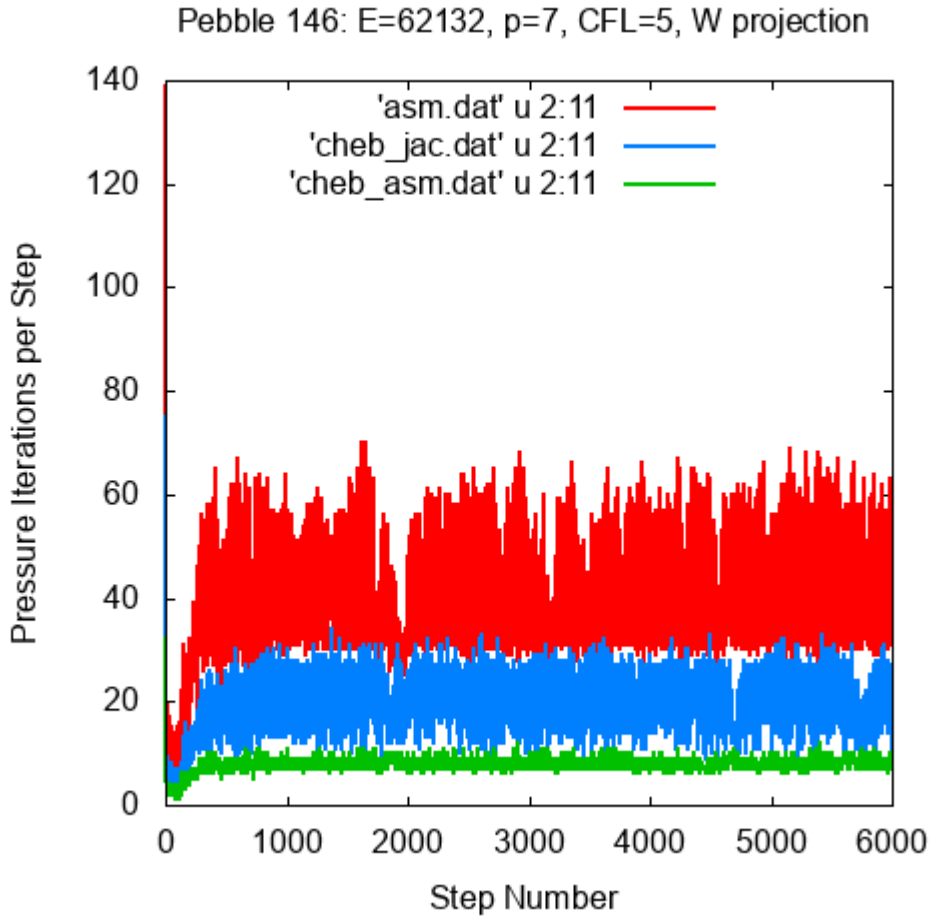
$$\underline{z}_e = (S_s \otimes S_r)[I_s \otimes \Lambda_r + \Lambda_s \otimes I_r]^{-1}(S_s^T \otimes S_r^T) R_e \underline{r}$$

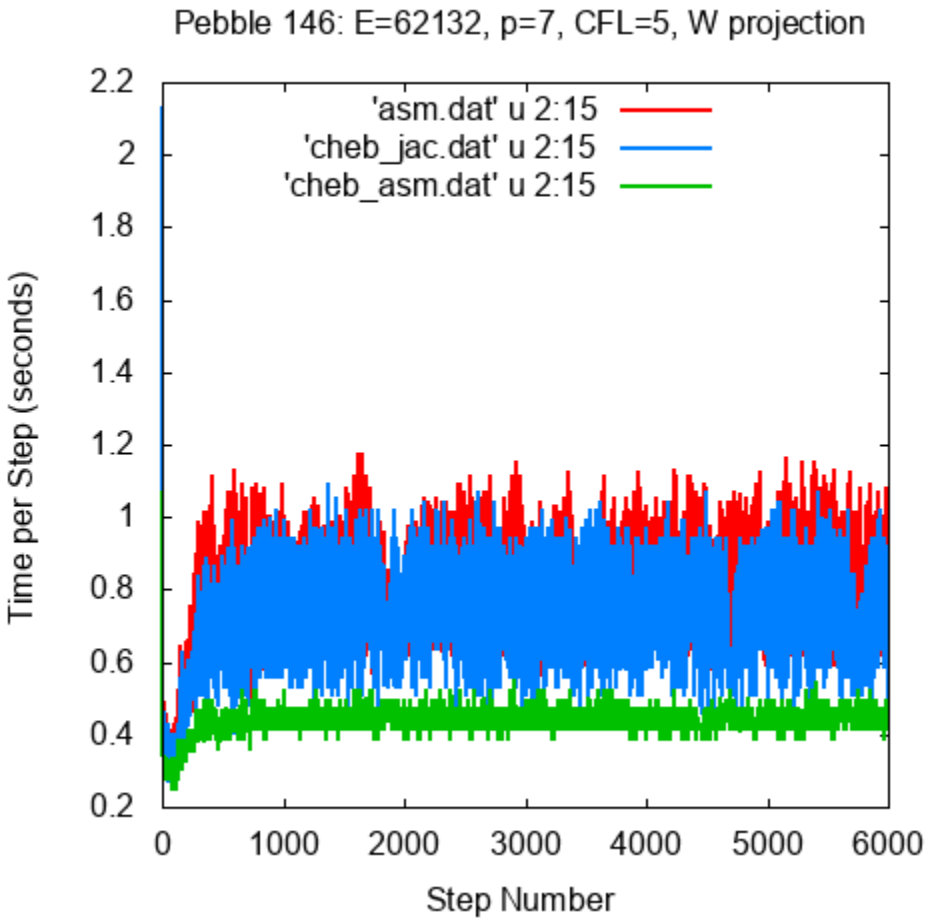
$$A_r^e S_r = S_r \Lambda_r, \quad A_s^e S_s = S_s \Lambda_s,$$

Extended to more general operators: Pazner&Persson; Pablo Brubeck, …

PBR146 Results

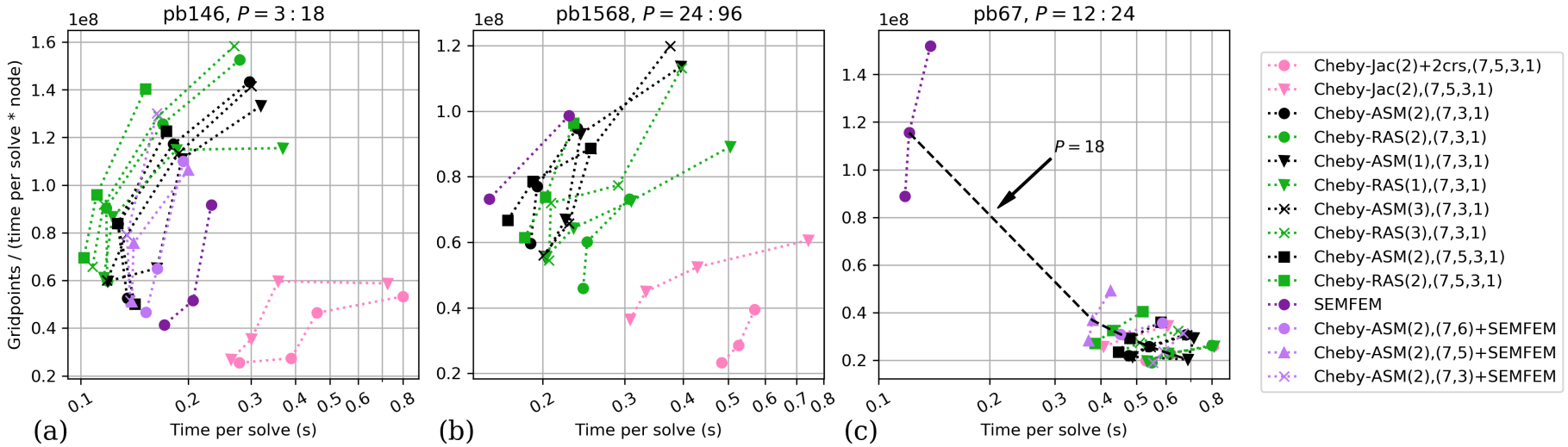
Malachi Phillips (UIUC)





Chebyshev-RAS vs. SEMFEM

Phillips et al'22



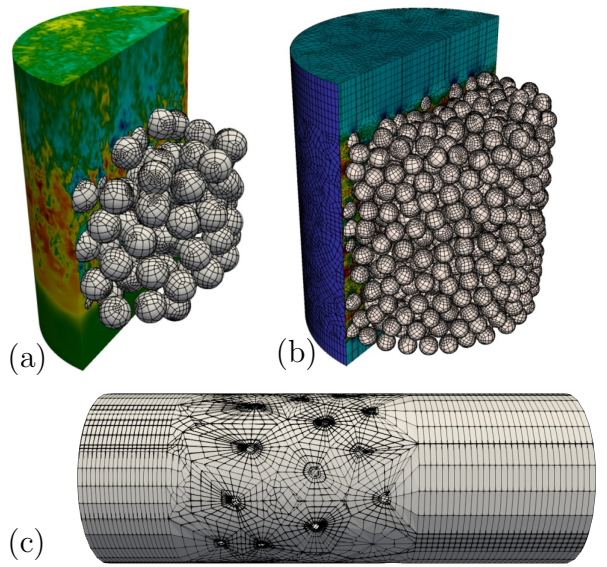
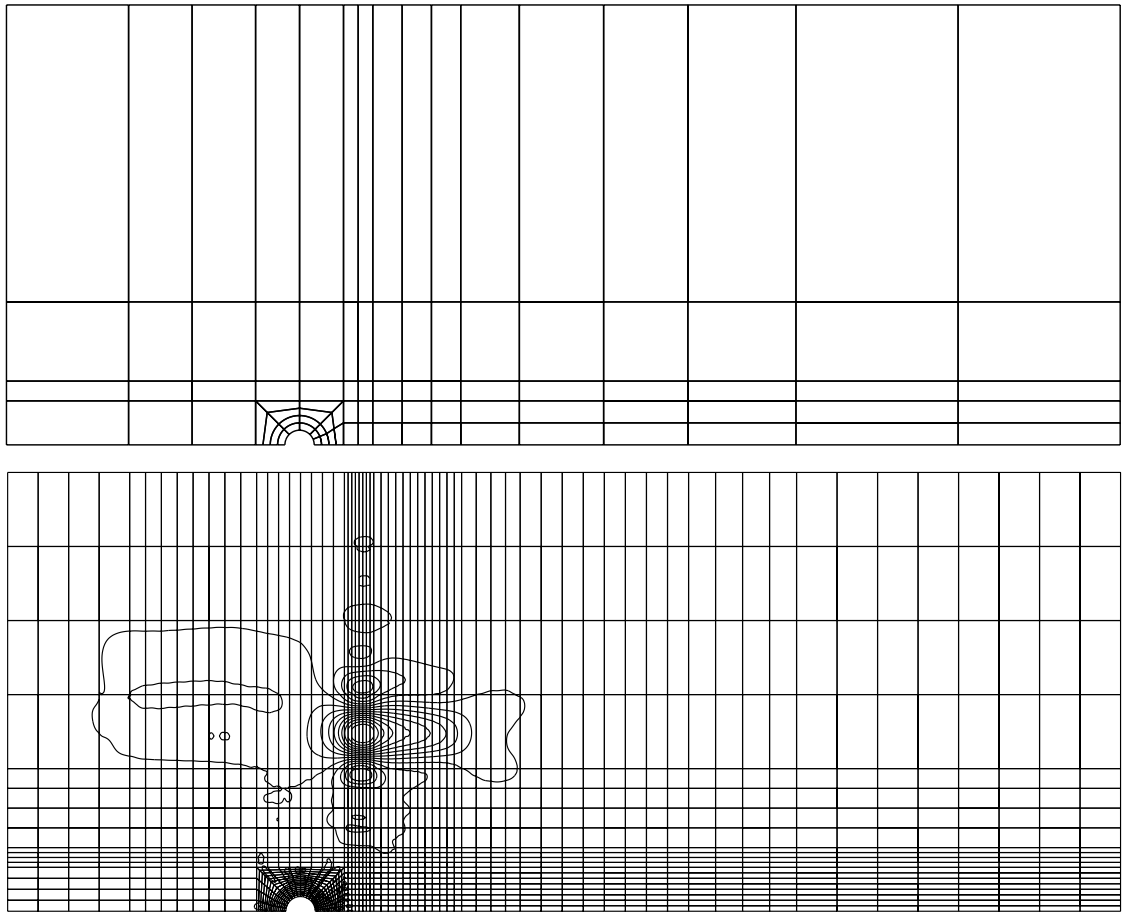


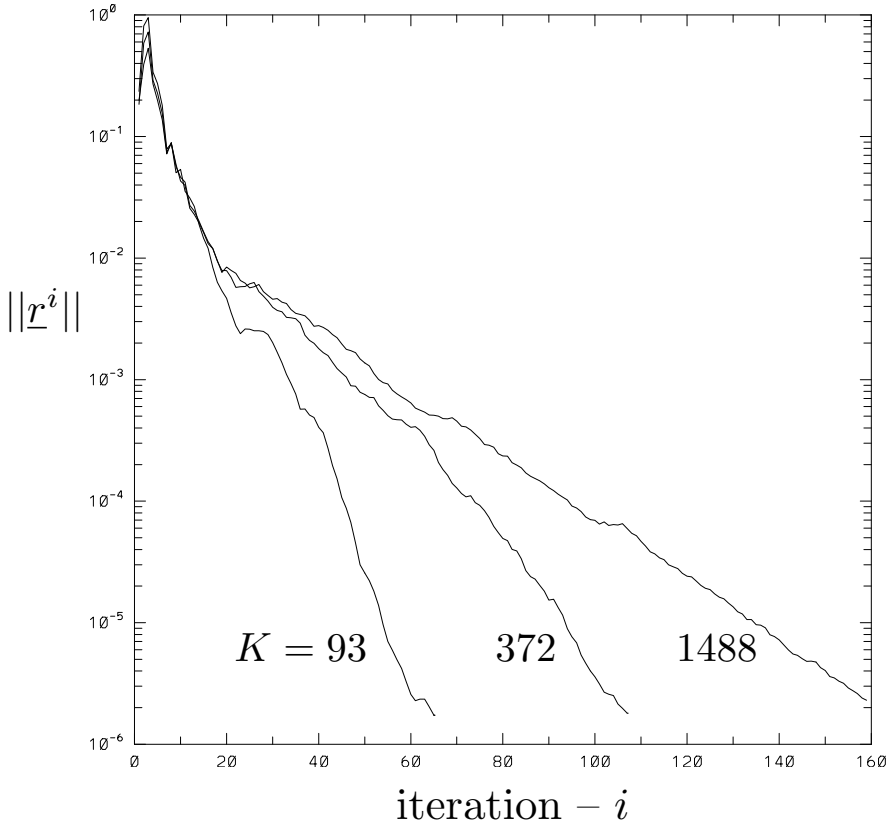
Figure 7: Strong scaling results on Summit for the Navier-Stokes cases of Fig. 3a,b,c. Iso-processor count line illustrated in (c). A user running on a specified number of processors should use the lowest time-to-solution preconditioner along this line.

For Case c, SEMFEM is a clear winner, but Cheby-ASM or Cheby-RAS are generally the productionoptions.

Impact of High-Aspect-Ratio Elements

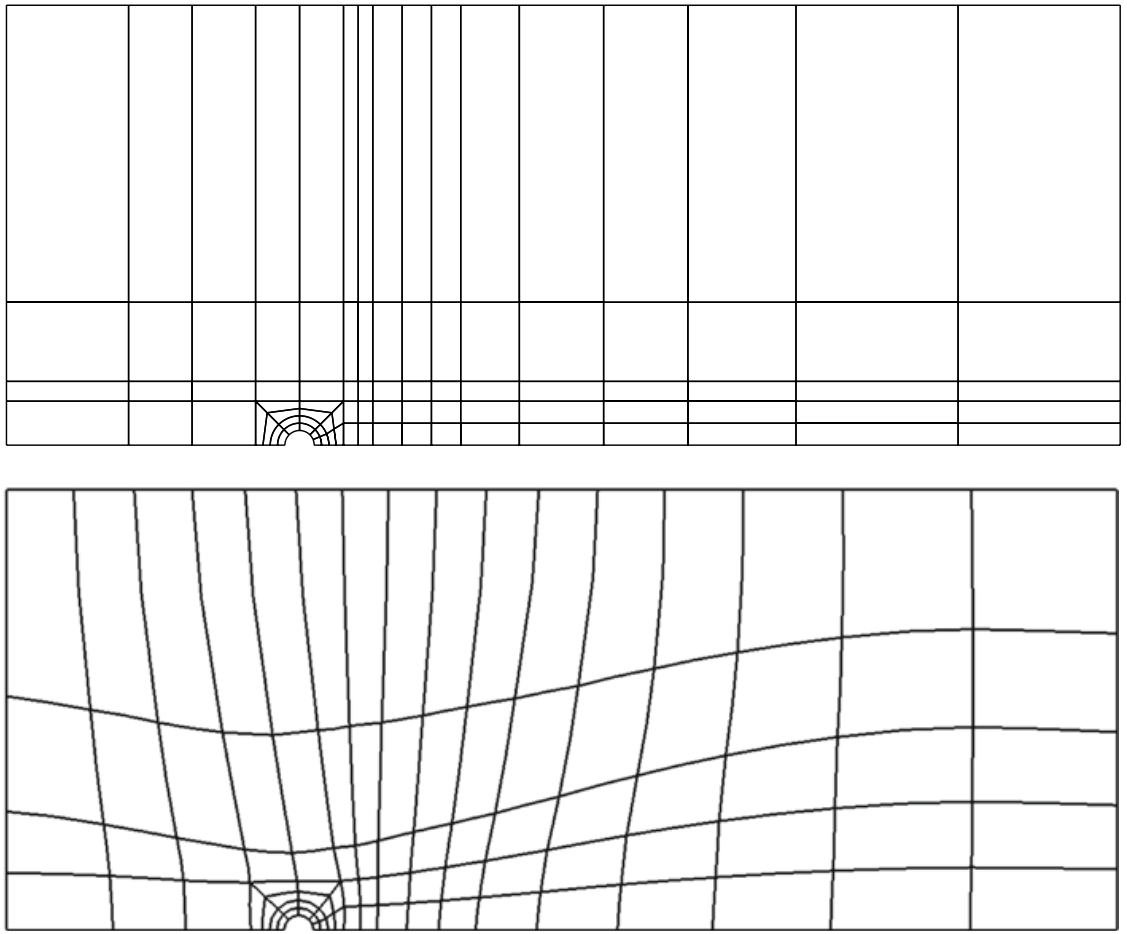
- High-aspect-ratio elements are a continual source of difficulty.





Impact of High-Aspect-Ratio Elements K. Mittal '18

- Sometimes can be repaired by mesh smoothing

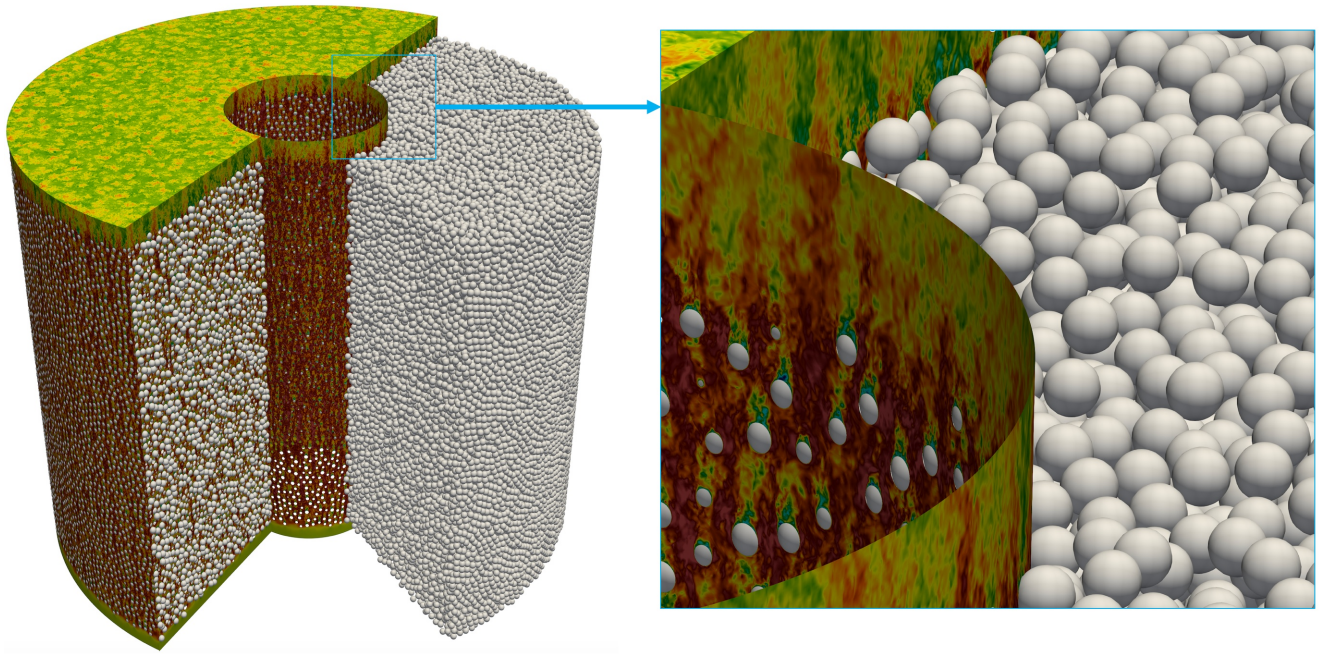


Case	E_s	N_{orig}	N_{smooth}	ΔN%
Half-cylinder	93	44	25	43.2
Half-cylinder	372	37	21	43.2
Half-cylinder	1488	73	39	46.6
LPT	532	6.4	5.6	12.9
Cylinder	1472	7.1	5.3	21.2
Oscillating flow	83598	21.3	20.4	3.9
Hemi-sphere	2072	4.19	3.9	6.8
Piston cylinder	6784	22.7	19.3	15.0

Ketan Mittal & PF, Mesh Smoothing for the SEM, J. Sci. Comp., 2018

Extreme Scalability: 352,000-Pebble Bed – 27648 V100s

Y.Lan, M.Min, E. Merzari



Pre-tuning timing breakdown: .597 s/step

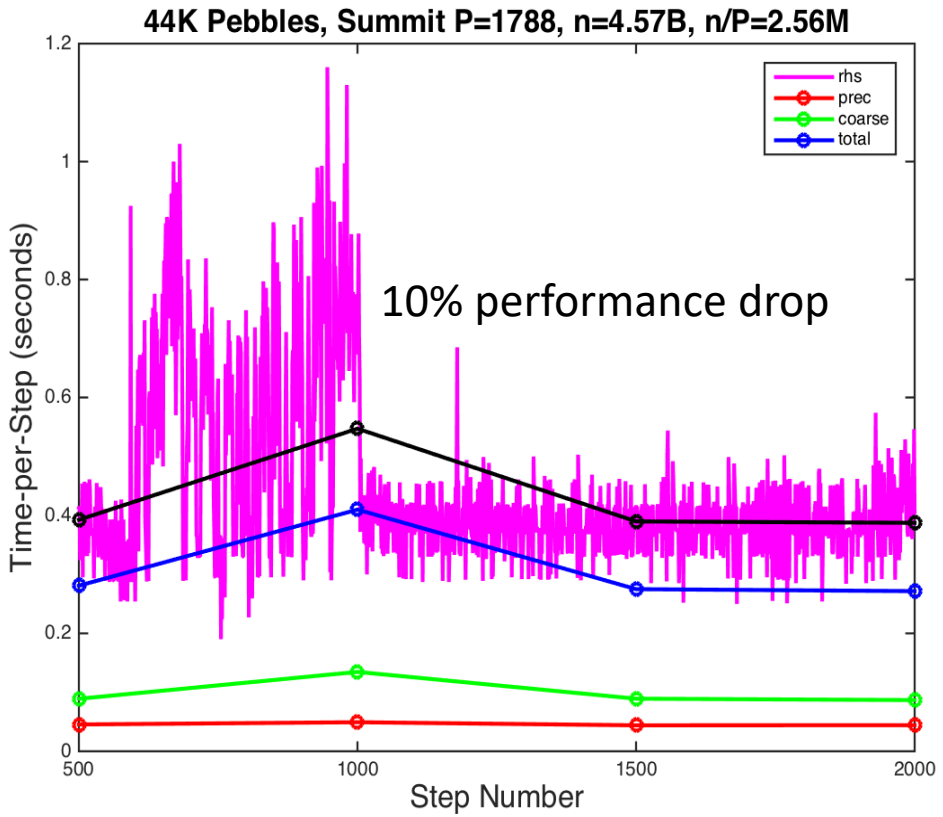
Operation	time (s)	%
computation	1.19+03	100
advection	5.82+01	5
viscous update	5.38+01	5
pressure solve	1.08+03	90
precond.	9.29+02	78
coarse grid	5.40+02	45
projection	6.78+00	1
dotp	4.92+01	4

Q: How to control coarse grid costs without control over the coarse grid solver?

(NB: "coarse" grid = 100M unknowns…)

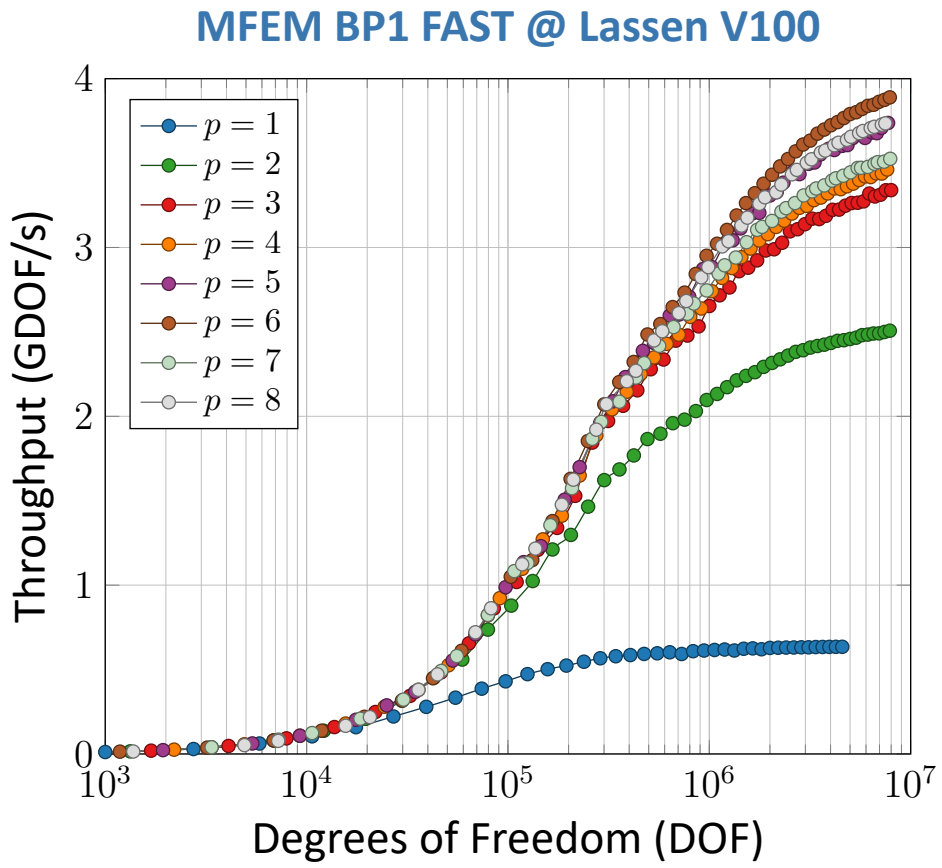
Issues that make the coarse-grid solve challenging

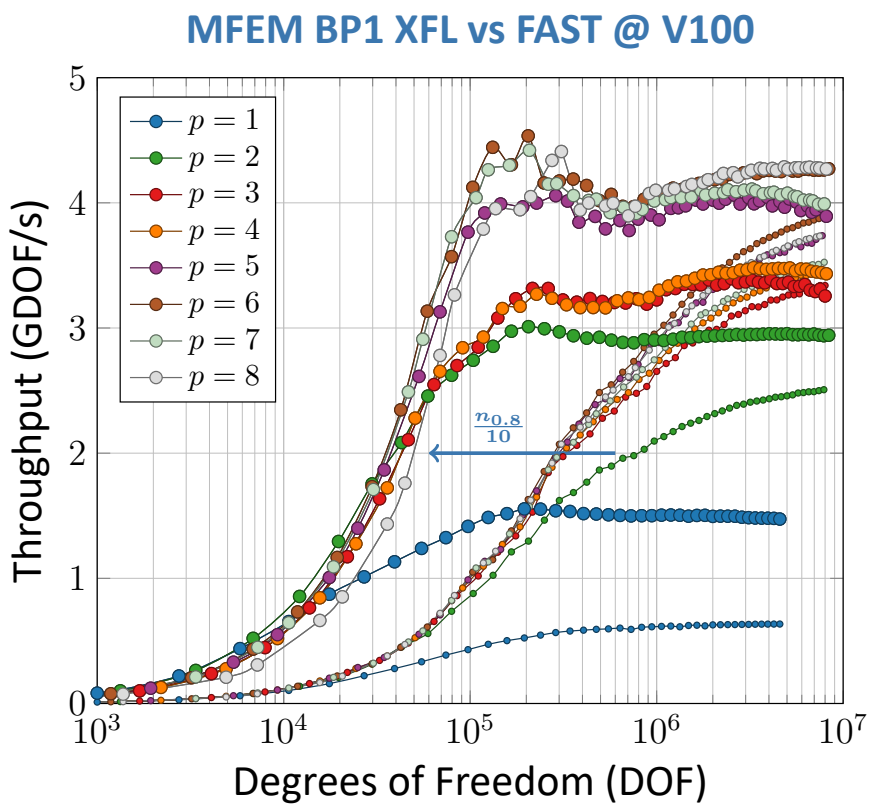
- Communication intensive -
  - small amount of work per rank
  - all-to-all communication
    (Green's functions cover entire domain)
- Too little work to keep GPU happy
  - kernel launch overhead
- System noise (network/node)



Issues that make the coarse-grid solve challenging

- Kernel launch overhead - Jean-Sylvain Camier (CEED MS37)





Number of rows and nonzeros in AMG

(E=580,000)

- Key observations:
  - n_{dofs} < P → idle some processors. OK.
  - Number of nonzeros does not drop as rapidly as number of rows
  - Stencil width grows at lower levels

    → 100s of nonzeros per row

    → More messages per processor

    → Alternative message exchange strategy at lower levels.

    → Rewrite ***gs()***

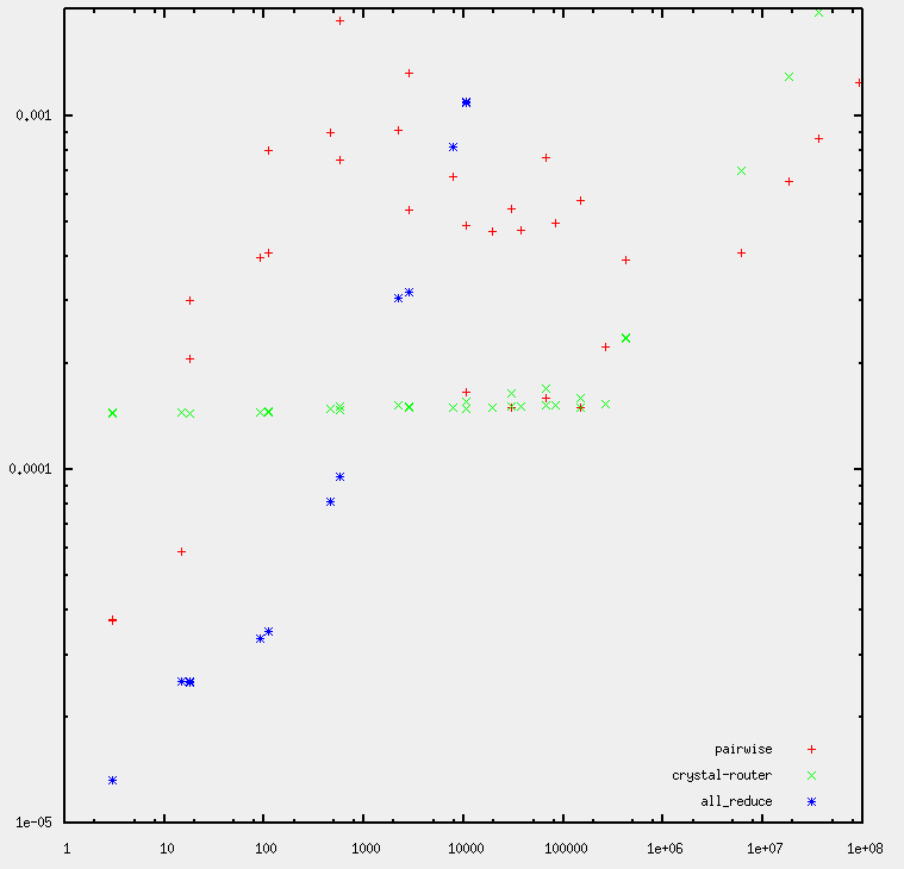
    3 exchange strategies:

    pairwise, all_reduce, crystal-router [Fox et al.,88]

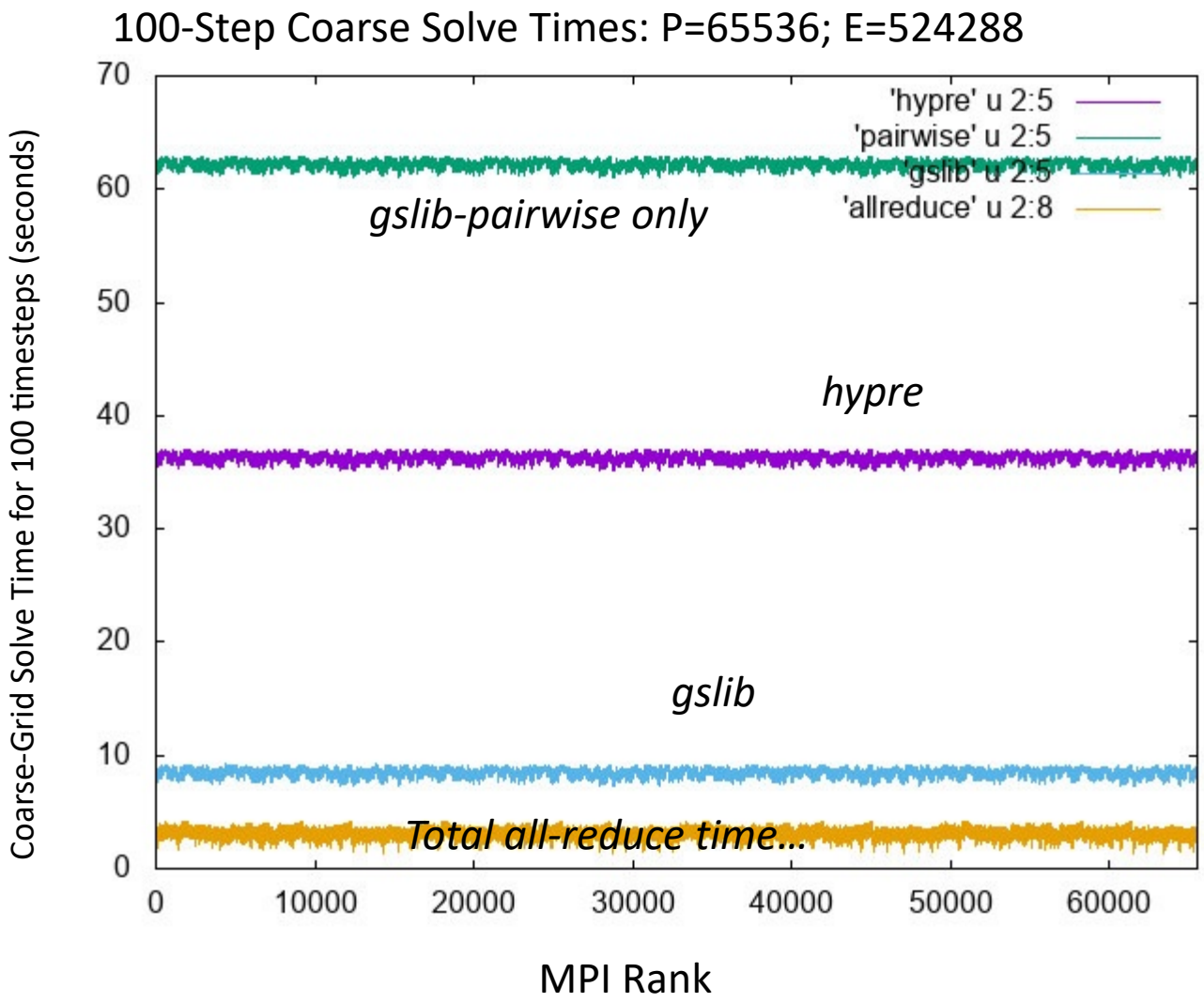
Level	n_{dofs}	*nnz*
0	*665820*	
1	*304403*	*15668640.*
2	*204979*	*20863046.*
3	*96379*	*11293784.*
4	*38094*	*5095546.*
5	*16123*	*2051300.*
6	*4754*	*459490.*
7	*927*	*25760.*
8	*138*	*506.*
9	*18*	*20.*

gs() times – P=131K

- ***Red – pairwise***, ***green – cr()***, ***blue – all_reduce***
- Horizontal axis – number of nontrivial (shared) columns in matrix
- cr() and all_reduce > *5-10 X* faster in many cases



Coarse-Grid Solve times on Mira



Controlling Coarse-Grid Solve Costs – 352K pebble case

- Reduce the number of times we visit the bottom of the V-cycle, i.e., the number of iterations
  - Projection onto previous timesteps
  - GMRES instead of Flexible CG
  - More smoothing on the fine levels
- *Note that, based on previous work, we know that we are at or a bit below the strong-scale limit: 50.5B points / 27648 = 1.83 M points per GPU*

NekRS Strong Scale: Rod-Bundle, 200 Steps						
Node	GPU	E	n	n/P	$t_{\text{step}}[s]$	Eff
1810	10860	175M	60B	5.5M	1.85e-01	100
2536	15216	175M	60B	3.9M	1.51e-01	87
3620	21720	175M	60B	2.7M	1.12e-01	82
4180	25080	175M	60B	2.4M	1.12e-01	71
4608	27648	175M	60B	2.1M	1.03e-01	70

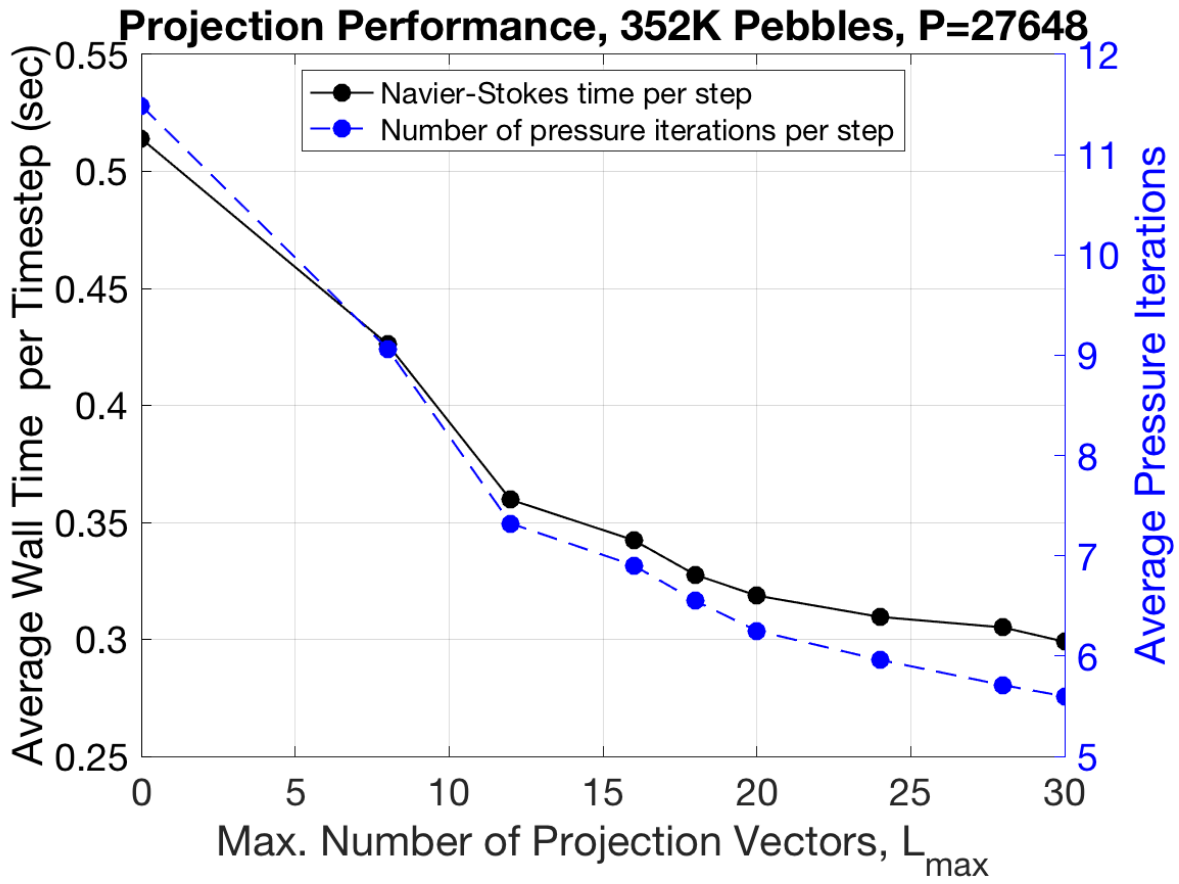
NekRS, a GPU-Accelerated Spectral Element Navier-Stokes Solver Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, Tim Warburton

Controlling Coarse-Grid Solve Costs – 352K pebble case

Major Algorithmic Variations, 352K Pebbles, P=27648								
Case	Solver	Smoother	L	N_q	Δt	v_i	p_i	t_{step}
(a)	FlexCG	1-Cheb-ASM:851	8	13	4e-4	3.6	22.8	.68
(b)	FlexCG	2-Cheb-Jac:851	8	13	4e-4	3.6	17.5	.557
(c)	"	2-Cheb-ASM:851	8	13	4e-4	3.6	12.8	.468
(d.8)	"	2-Cheb-ASM:8641	8	13	4e-4	3.6	9.1	.426
(d.L)	"	2-Cheb-ASM:8641	0–30	13	4e-4	3.6	5.6	.299
(e)	GMRES	"	30	13	4e-4	3.5	4.6	.240
(f)	"	"	30	11	8e-4	5.7	7.2	.376
(g)	"	"	30	11	8e-4	5.7	7.2	.361 (no I/O)

Controlling Coarse-Grid Solve Costs – 352K pebble case

NekRS Strong Scale: 352K pebbles, E=98M, n=50B						
$N=8$, $N_q=13$, Δt = **4.e-4**, $L=8$, **1-Cheb-Jac:851**						
Node	GPU	n/P	v_i	p_i	t_{step}	Eff
1536	9216	5.4M	3.6	17.3	.97	1.00
2304	13824	3.6M	3.6	18.0	.84	76.9
3072	18432	2.7M	3.6	16.6	.75	64.6
3840	23040	2.1M	3.6	19.6	.67	57.9
4608	27648	1.8M	3.6	17.5	.55	58.7
$N=8$, $N_q=13$, Δt = **4.e-4**, $L=8$, **1-Cheb-ASM:851**						
Node	GPU	n/P	v_i	p_i	t_{step}	Eff
1536	9216	5.4M	3.6	11.6	.81	100
2304	13824	3.6M	3.6	12.3	.65	83.0
3072	18432	2.7M	3.6	12.3	.71	57.0
3840	23040	2.1M	3.6	13.5	.54	60.0
4608	27648	1.8M	3.6	12.8	.46	58.6
$N=8$, $N_q=11$, Δt = **8.e-4**, $L=30$, **2-Cheb-ASM:8641**						
Node	GPU	n/P	v_i	p_i	t_{step}	Eff
1536	9216	5.4M	-	-	-	-
2304	13824	3.6M	5.7	7.2	.55	100
3072	18432	2.7M	5.7	7.2	.56	73.6
3840	23040	2.1M	5.7	7.2	.39	84.6
4608	27648	1.8M	5.7	7.2	.36	76.3



Controlling Coarse-Grid Solve Costs – 352K pebble case

- Summary:
  - Projection / GMRES quite important
  - More smoothing is better at scale, to take pressure off coarse grid solve
  - I/O rates: output snapshots (32bit) ~ 4TB , 40 seconds per output (100X step time).
- Bottom line:
  - Total flow-through time for full core is only 6 hours of wall-clock time on Summit!

Highly-Tuned Solver Kernels

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

FDM FP32 Kernel Performance (GFLOPS)				
p	A100 pre-tune	MI250X pre-tune	A100 post-tune	MI250X 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

From NekRS logfile:

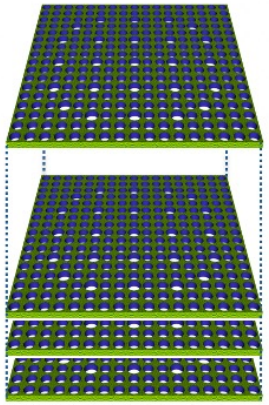
```
 Ax: N=7 FP64  GB/s=  997.573  GFLOPS= 1730.2  kernel=4
 Ax: N=7 FP64  GB/s=  997.311  GFLOPS= 1729.7  kernel=4
 Ax: N=7 FP32  GB/s= 1010.990  GFLOPS= 3506.9  kernel=2

 Ax: N=3 FP64  GB/s=  691.006  GFLOPS=  680.2  kernel=0
 Ax: N=3 FP32  GB/s=  696.434  GFLOPS= 1371.1  kernel=1

fdm: N=9 FP32  GB/s=  601.657  GFLOPS= 5515.2  kernel=3
fdm: N=5 FP32  GB/s=  670.688  GFLOPS= 3497.2  kernel=2
```

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

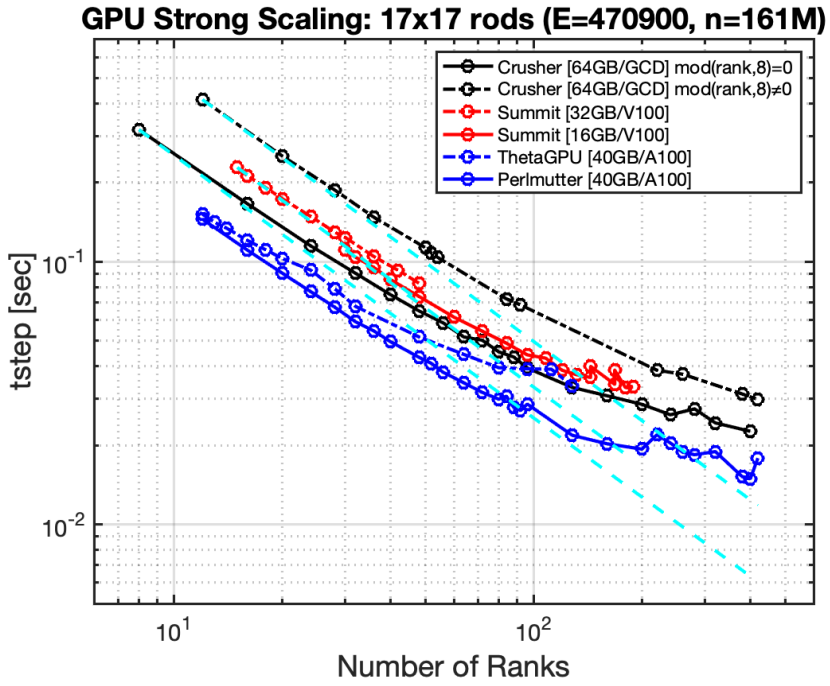
ExaSMR Timings

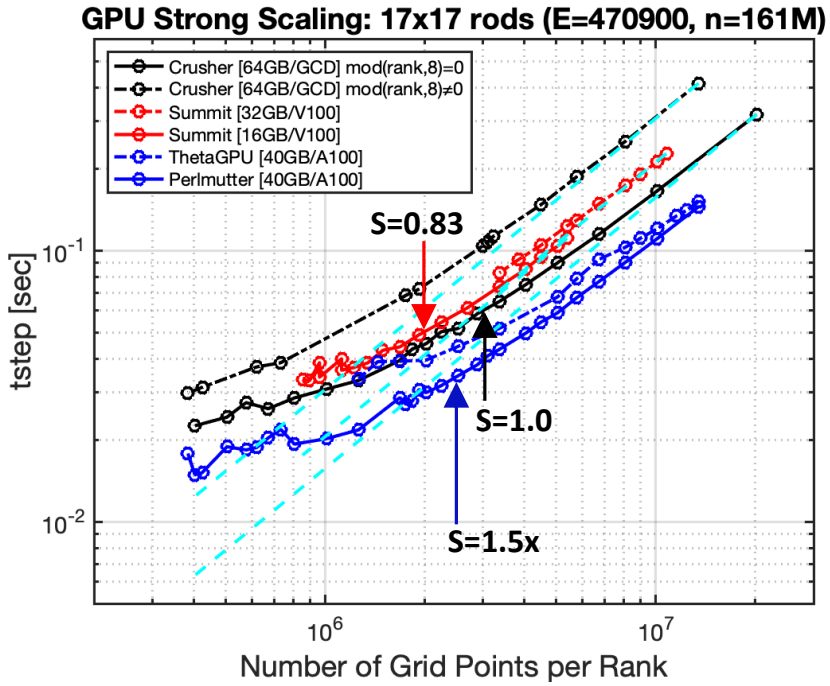
M.Min, Y.H. Lan, M. Phillips

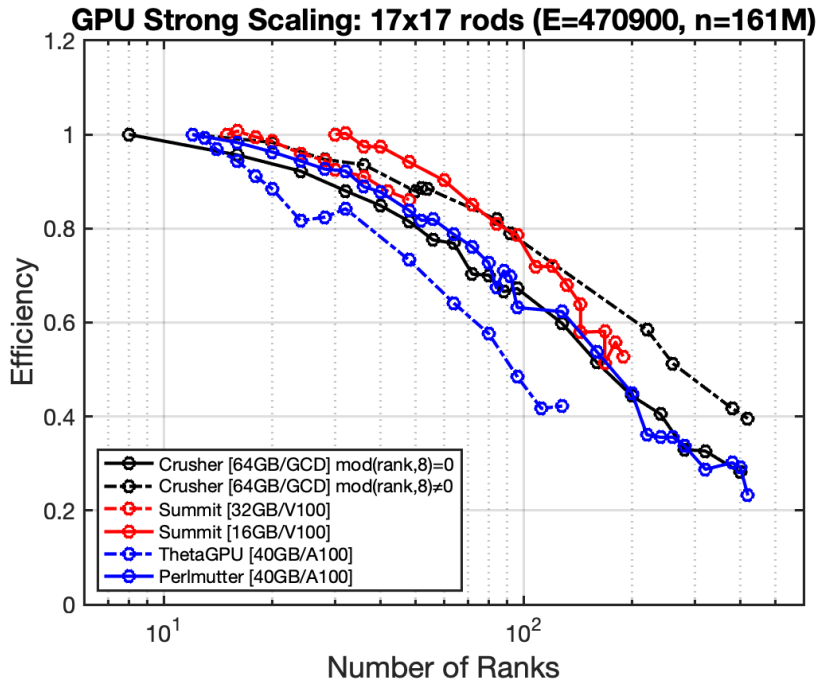
- ❑ Summit
- ❑ Crusher
- ❑ ThetaGPU
- ❑ Perlmutter

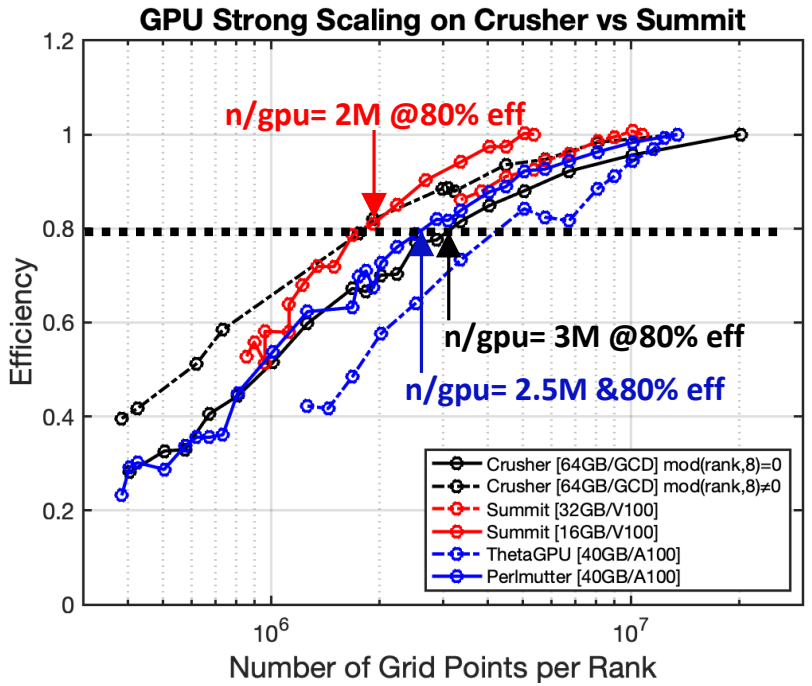


- 17x17 rod bundle
- Mesh:
  2D: E= 27700
  3D: E=27700 x 17 layers
- Resolution
  E=470,900, N=7
  n=161,518,700
- BDF3, CFL=0.67
- Re=5000





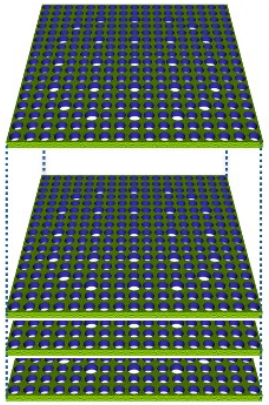




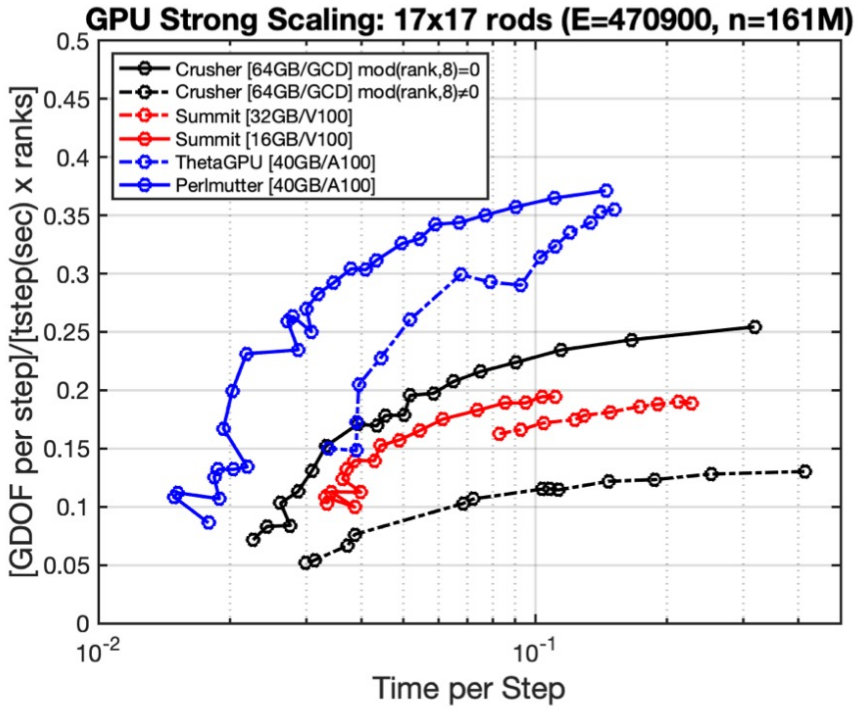
ExaSMR Timings

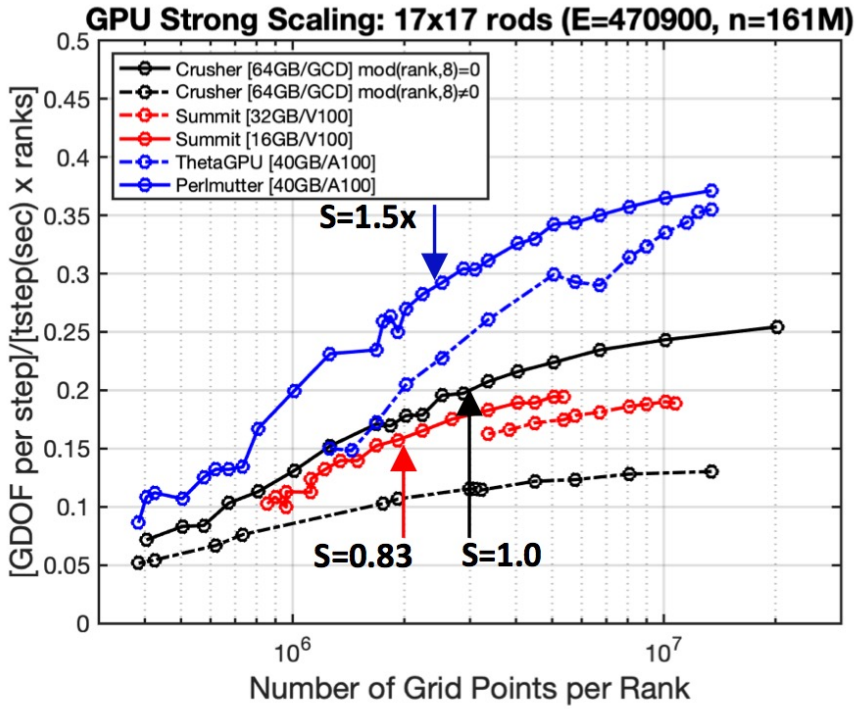
M.Min, Y.H. Lan, M. Phillips

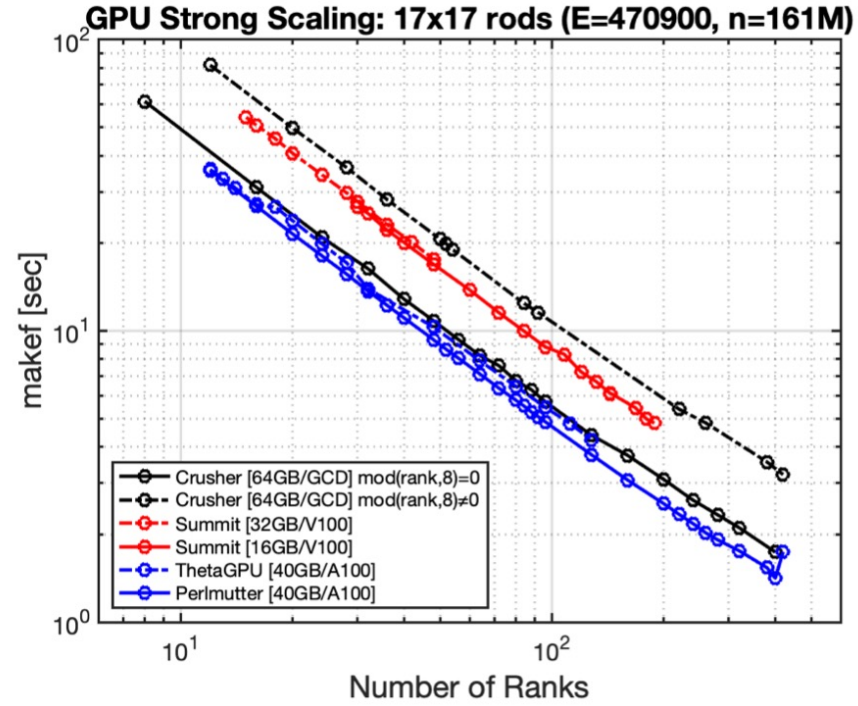
- ❑ Summit
- ❑ Crusher
- ❑ ThetaGPU
- ❑ Perlmutter

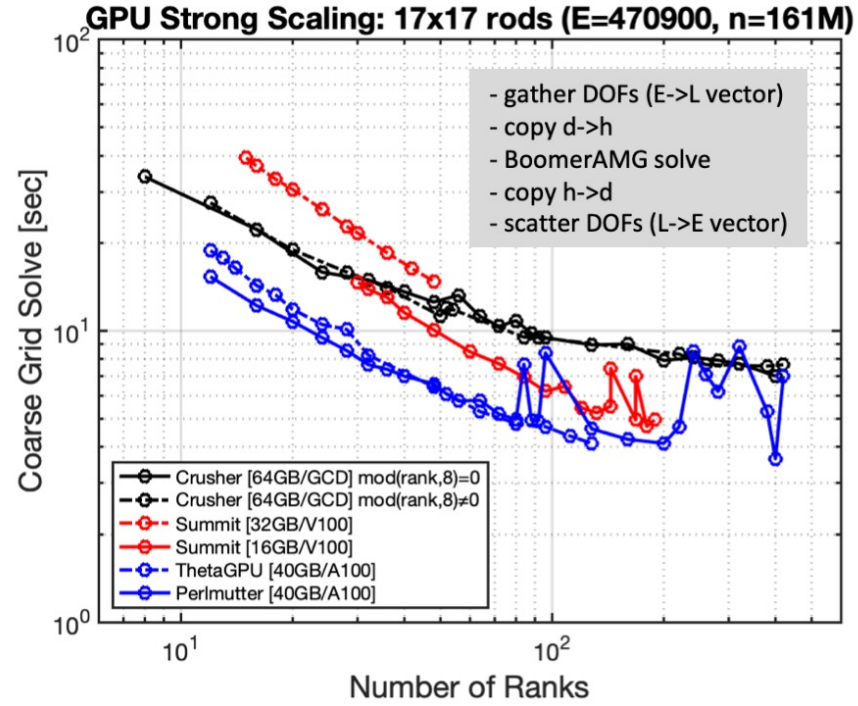


- 17x17 rod bundle
- Mesh:
  2D: E= 27700
  3D: E=27700 x 17 layers
- Resolution
  E=470,900, N=7
  n=161,518,700
- BDF3, CFL=0.67
- Re=5000









ExaSMR: *Performance on Crusher (vs. A100, V100)*

M. Min, Y. Lan ANL

Crusher, $n_{0.8}$ = 4 M

crusher: mod(rank,8)=0

Node	Ranks	n/rank	tstep	eff
1.00e+00	8.00e+00	2.01e+07	3.17e-01	1.00e+00
2.00e+00	1.60e+01	1.00e+07	1.66e-01	9.56e-01
3.00e+00	2.40e+01	6.72e+06	1.14e-01	9.22e-01
4.00e+00	3.20e+01	5.04e+06	9.02e-02	8.79e-01
5.00e+00	4.00e+01	4.03e+06	7.48e-02	8.48e-01
6.00e+00	4.80e+01	3.36e+06	6.49e-02	8.15e-01
7.00e+00	5.60e+01	2.88e+06	5.84e-02	7.75e-01
8.00e+00	6.40e+01	2.52e+06	5.16e-02	7.68e-01
9.00e+00	7.20e+01	2.24e+06	5.01e-02	7.02e-01
1.00e+01	8.00e+01	2.01e+06	4.53e-02	7.00e-01
1.10e+01	8.80e+01	1.83e+06	4.33e-02	6.65e-01
1.20e+01	9.60e+01	1.68e+06	3.93e-02	6.72e-01
1.60e+01	1.28e+02	1.26e+06	3.31e-02	5.97e-01
2.00e+01	1.60e+02	1.00e+06	3.08e-02	5.14e-01
2.50e+01	2.00e+02	8.07e+05	2.86e-02	4.43e-01
3.00e+01	2.40e+02	6.72e+05	2.61e-02	4.05e-01
3.50e+01	2.80e+02	5.76e+05	2.75e-02	3.29e-01
4.00e+01	3.20e+02	5.04e+05	2.43e-02	3.26e-01
5.00e+01	4.00e+02	4.03e+05	2.26e-02	2.80e-01

crusher: mod(rank,8)\=0

1.50e+00	1.20e+01	1.34e+07	4.13e-01	1.00e+00
2.50e+00	2.00e+01	8.07e+06	2.52e-01	9.83e-01
3.50e+00	2.80e+01	5.76e+06	1.87e-01	9.46e-01
4.50e+00	3.60e+01	4.48e+06	1.47e-01	9.35e-01
6.25e+00	5.00e+01	3.23e+06	1.12e-01	8.79e-01
6.50e+00	5.20e+01	3.10e+06	1.07e-01	8.86e-01
6.75e+00	5.40e+01	2.99e+06	1.03e-01	8.85e-01
1.05e+01	8.40e+01	1.92e+06	7.21e-02	8.19e-01
1.15e+01	9.20e+01	1.75e+06	6.83e-02	7.89e-01
2.75e+01	2.20e+02	7.34e+05	3.85e-02	5.84e-01
3.25e+01	2.60e+02	6.21e+05	3.72e-02	5.12e-01
4.75e+01	3.80e+02	4.25e+05	3.13e-02	4.16e-01
5.25e+01	4.20e+02	3.84e+05	2.98e-02	3.95e-01

Perlmutter, $n_{0.8}$ = 2.9 M

perlmutter a100:

Node	Ranks	n/rank	tstep	eff
1.50e+00	1.20e+01	1.34e+07	1.45e-01	1.00e+00
2.00e+00	1.60e+01	1.00e+07	1.10e-01	9.82e-01
2.50e+00	2.00e+01	8.07e+06	9.03e-02	9.62e-01
3.00e+00	2.40e+01	6.72e+06	7.68e-02	9.43e-01
3.50e+00	2.80e+01	5.76e+06	6.70e-02	9.26e-01
4.00e+00	3.20e+01	5.04e+06	5.89e-02	9.22e-01
4.50e+00	3.60e+01	4.48e+06	5.43e-02	8.88e-01
5.00e+00	4.00e+01	4.03e+06	4.95e-02	8.77e-01
6.00e+00	4.80e+01	3.36e+06	4.32e-02	8.38e-01
6.50e+00	5.20e+01	3.10e+06	4.09e-02	8.17e-01
7.00e+00	5.60e+01	2.88e+06	3.79e-02	8.19e-01
8.00e+00	6.40e+01	2.52e+06	3.45e-02	7.87e-01
9.00e+00	7.20e+01	2.24e+06	3.17e-02	7.60e-01
1.00e+01	8.00e+01	2.01e+06	2.99e-02	7.26e-01
1.05e+01	8.40e+01	1.92e+06	3.07e-02	6.73e-01
1.10e+01	8.80e+01	1.83e+06	2.78e-02	7.09e-01
1.15e+01	9.20e+01	1.75e+06	2.71e-02	6.97e-01
1.20e+01	9.60e+01	1.68e+06	2.86e-02	6.31e-01
1.60e+01	1.28e+02	1.26e+06	2.18e-02	6.23e-01
2.00e+01	1.60e+02	1.00e+06	2.02e-02	5.37e-01
2.50e+01	2.00e+02	8.07e+05	1.93e-02	4.50e-01
2.75e+01	2.20e+02	7.34e+05	2.19e-02	3.61e-01
3.00e+01	2.40e+02	6.72e+05	2.03e-02	3.55e-01
3.25e+01	2.60e+02	6.21e+05	1.88e-02	3.56e-01
3.50e+01	2.80e+02	5.76e+05	1.84e-02	3.37e-01
4.00e+01	3.20e+02	5.04e+05	1.89e-02	2.87e-01
4.75e+01	3.80e+02	4.25e+05	1.51e-02	3.01e-01
5.00e+01	4.00e+02	4.03e+05	1.49e-02	2.91e-01
5.25e+01	4.20e+02	3.84e+05	1.78e-02	2.31e-01

ThetaGPU, $n_{0.8}$ = 6.7 M

thetagpu a100:

Node	Ranks	n/rank	tstep	eff
1.50e+00	1.20e+01	1.34e+07	1.51e-01	1.00e+00
1.62e+00	1.30e+01	1.24e+07	1.40e-01	9.92e-01
1.75e+00	1.40e+01	1.15e+07	1.34e-01	9.68e-01
2.00e+00	1.60e+01	1.00e+07	1.20e-01	9.44e-01
2.25e+00	1.80e+01	8.97e+06	1.10e-01	9.10e-01
2.50e+00	2.00e+01	8.07e+06	1.02e-01	8.84e-01
3.00e+00	2.40e+01	6.72e+06	9.27e-02	8.16e-01
3.50e+00	2.80e+01	5.76e+06	7.88e-02	8.24e-01
4.00e+00	3.20e+01	5.04e+06	6.74e-02	8.41e-01
6.00e+00	4.80e+01	3.36e+06	5.16e-02	7.33e-01
8.00e+00	6.40e+01	2.52e+06	4.43e-02	6.41e-01
1.00e+01	8.00e+01	2.01e+06	3.94e-02	5.76e-01
1.20e+01	9.60e+01	1.68e+06	3.90e-02	4.84e-01
1.40e+01	1.12e+02	1.44e+06	3.89e-02	4.16e-01
1.60e+01	1.28e+02	1.26e+06	3.36e-02	4.22e-01

Summit, $n_{0.8}$ = 1.92 M

summit v100 [16GB nodes]:

Node	Ranks	n/rank	tstep	eff
5.00e+00	3.00e+01	5.38e+06	1.10e-01	1.00e+00
5.33e+00	3.20e+01	5.04e+06	1.03e-01	1.00e+00
6.00e+00	3.60e+01	4.48e+06	9.49e-02	9.74e-01
6.66e+00	4.00e+01	4.03e+06	8.54e-02	9.74e-01
8.00e+00	4.80e+01	3.36e+06	7.36e-02	9.41e-01
1.00e+01	6.00e+01	2.69e+06	6.14e-02	9.02e-01
1.20e+01	7.20e+01	2.24e+06	5.43e-02	8.51e-01
1.40e+01	8.40e+01	1.92e+06	4.89e-02	8.10e-01
1.60e+01	9.60e+01	1.68e+06	4.41e-02	7.85e-01
1.80e+01	1.08e+02	1.49e+06	4.28e-02	7.18e-01
2.00e+01	1.20e+02	1.34e+06	3.85e-02	7.19e-01
2.20e+01	1.32e+02	1.22e+06	3.70e-02	6.80e-01
2.40e+01	1.44e+02	1.12e+06	3.62e-02	6.38e-01
2.80e+01	1.68e+02	9.61e+05	3.41e-02	5.81e-01
3.00e+01	1.80e+02	8.97e+05	3.31e-02	5.58e-01
3.15e+01	1.89e+02	8.54e+05	3.33e-02	5.27e-01

summit v100 [32GB nodes]:

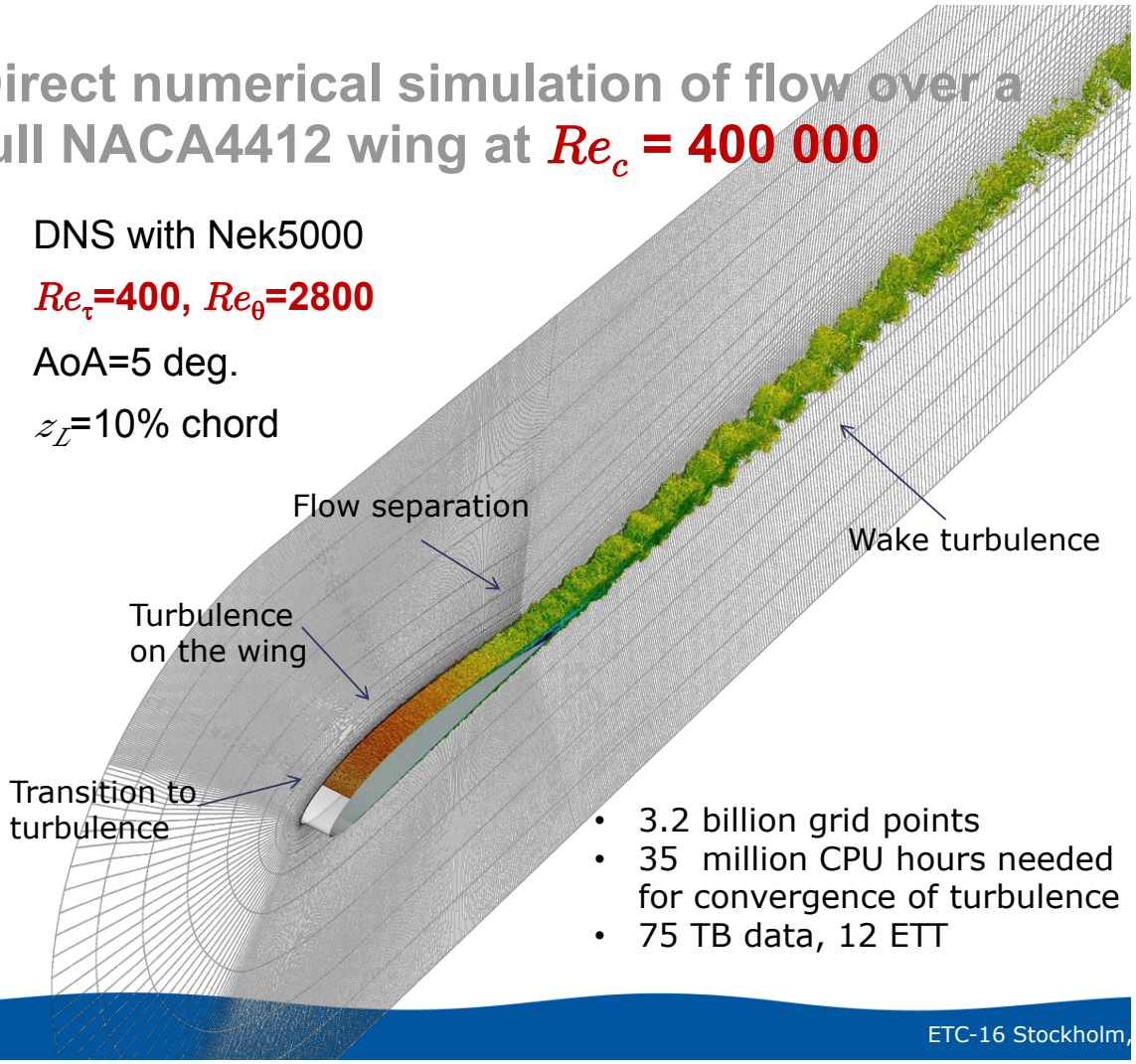
2.50e+00	1.50e+01	1.07e+07	2.28e-01	1.00e+00
2.66e+00	1.60e+01	1.00e+07	2.12e-01	1.00e+00
3.00e+00	1.80e+01	8.97e+06	1.91e-01	9.94e-01
3.33e+00	2.00e+01	8.07e+06	1.73e-01	9.85e-01
4.00e+00	2.40e+01	6.72e+06	1.48e-01	9.59e-01
4.66e+00	2.80e+01	5.76e+06	1.29e-01	9.44e-01
5.00e+00	3.00e+01	5.38e+06	1.23e-01	9.25e-01
6.00e+00	3.60e+01	4.48e+06	1.04e-01	9.10e-01
7.00e+00	4.20e+01	3.84e+06	9.26e-02	8.79e-01
8.00e+00	4.80e+01	3.36e+06	8.27e-02	8.61e-01

ThetaGPU: 0.91 TFLOPs/GPU, aggregate
Crusher: 0.64 TFLOPs
Summit: 0.48 TFLOPs

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

FLOW
LINNÉ FLOW CENTRE

- ❑ Consider this hero calculation from a few years ago.
- ❑ How many A100s?
- ❑ How many A100 hours?
- ❑ How many node hours?
- ❑ 1000 A100s
  - ❑ Each ~300X a CPU
  - ❑ 110K GPU hours
  - ❑ 110 wall clock hours

Conclusions

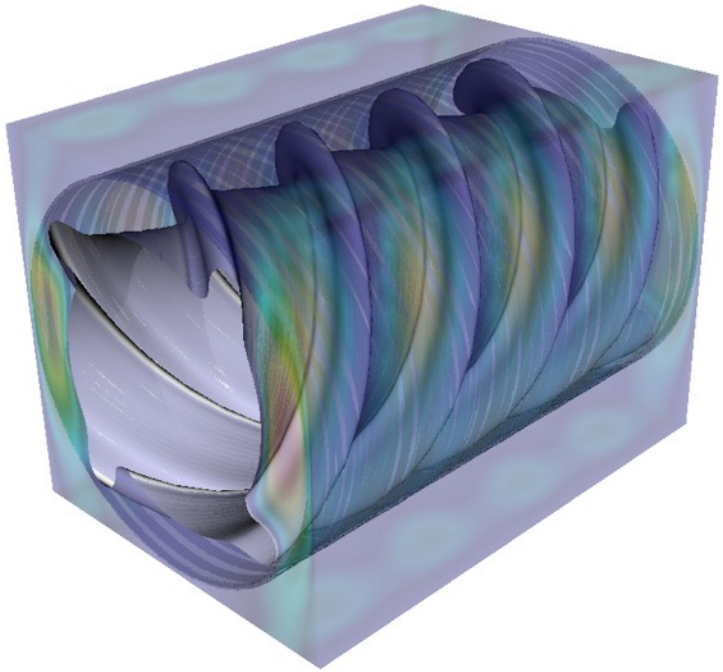
- Overview of scaling issues for Navier-Stokes
  - Users are most likely to operate around 80% efficiency (give or take…)
- Several preconditioners
- Importance of the coarse-grid solve (e.g., modify solution approach)
- The new machines are in fact delivering about 3X performance at the strong-scale limit:
  - New preconditioners (Cheby-RAS/ASM)
  - Highly-tuned OCCA kernels
  - Overlapped communication/computation
  - Mixed-precision preconditioners
  - On-the-fly tuning, everywhere.

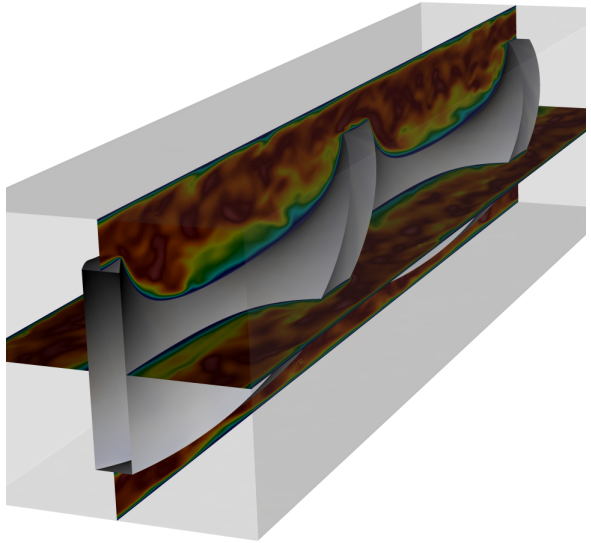
Thank you for your attention!

Schwarz-Based Nonconforming Methods

Y. Peet '12, '16, K. Mittal'19, Lindquist & Min '21

- Independent NS solves on distinct MPI communicators
  - prototype GPU port by Neil Lindquist & Misun Min
    - production version in NekRS V22.1
- Requires scalable general interpolation, ***findpts()*** [Lottes 2010]
- Extremely useful for
  - domains with relative twist
  - domains where two or more complex mesh topologies meet
  - domains with rapid variation in resolution requirements
  - rotating machinery
  - *etc.*





Interesting Chebyshev Smoother

James Lottes (Google)

James Lottes, Optimal Polynomial Smoothers for Multigrid V-cycles, ArXiv, 2022.

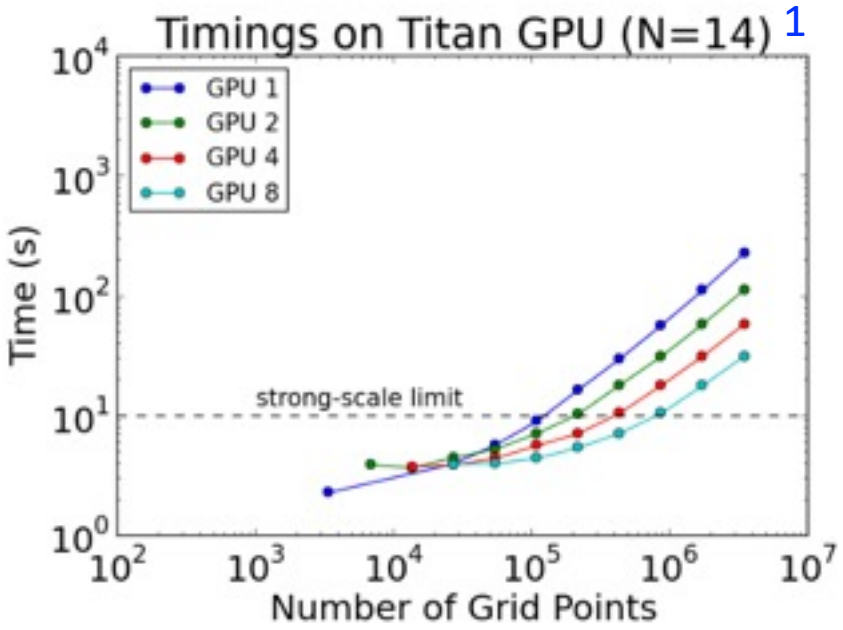
Outline:

0. *Introduction (preceding slides)*

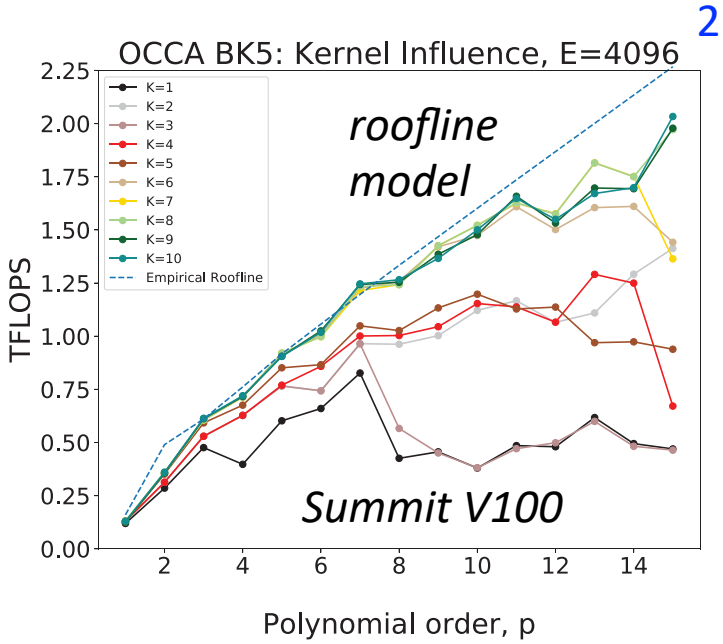
1. *Mathematical Background*
   - ❑ *Discretization - SEM*
   - ❑ *Navier-Stokes timestepping*
   - ❑ *Pressure Poisson Problem & Preconditioning*

2. *Parallel Computing Concerns*
   - ❑ *Scalability ⇐⇒ Speed*

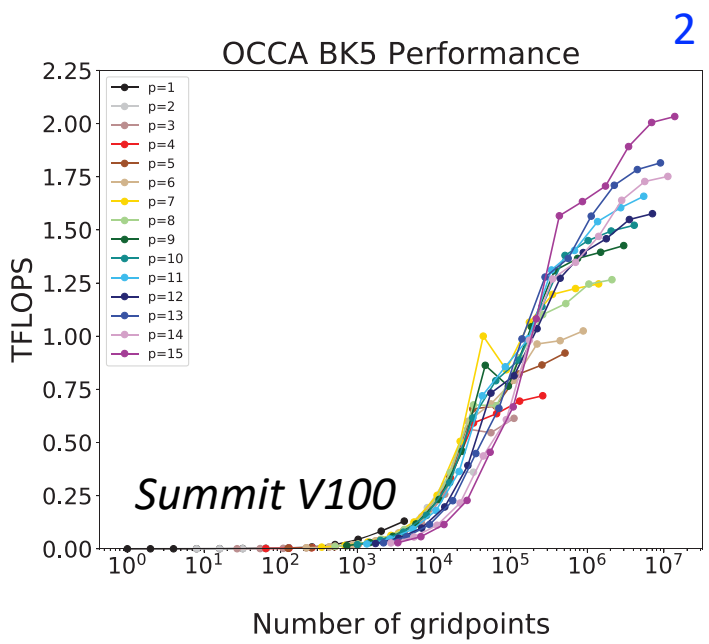
For HPC (exascale) systems, both MDOFS and $n_{0.8}$ are important



1



2



2

- On GPUs, can obviously realize high MDOFS (with significant effort)
- *Low $n_{0.8}$ is more challenging — even on one GPU!*
- In NekRS (and CEED in general), seek ways to reduce ($n_{0.8}$ / MDOFS) algorithmically and through vendor interactions (ECP co-design).

1 M. Otten, J. Gong, A. Mametjanov, A. Vose, J. Levesque, P. Fischer, and M. Min. An MPI/OpenACC implementation of a high order electromagnetics solver with GPUDirect communication. *Int. J. High Perf. Comput. Appl.*, 2016.

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

GPU Developments (NekRS)

- ❑ Highly-tuned kernels* sustaining 1-2 TFLOPS on V100 (w/o communication)
- ❑ Overlapped computation/communication
- ❑ Auto-tuned communication (similar to Nek5000/CEM *gslib*)
- ❑ 32-bit preconditioners to reduce communication overhead
- ❑ Extensive suite of multilevel preconditioners for pressure Poisson problem
- ❑ Extensive (and growing) support for multi-physics problems
- ❑ Scalable parallel I/O
- ❑ Ported to multiple GPUs (V100, A100, MI100, …)

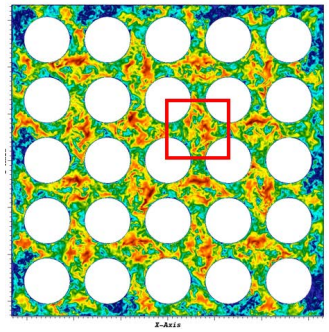
Initial OCCA kernel development from libparanumal library out of Tim Warburton's group at V.Tech.

GFLOPS

E=4096	V100	A100
AxHelm-BK5		
9	1406	2207
7	1280	2132
5	876	1287
AxHelm-BK5 FP32		
9	2584	3936
7	2350	3978
5	1440	2556
advSub		
9	2701	3147
7	2800	3800
5	2702	3158
FDM FP32		
9	3954	5092
7	2210	2870
5	1847	2475

Spacer Grid Performance: n=1.1 B

Misun Min, ANL, Elia Merzari, PSU



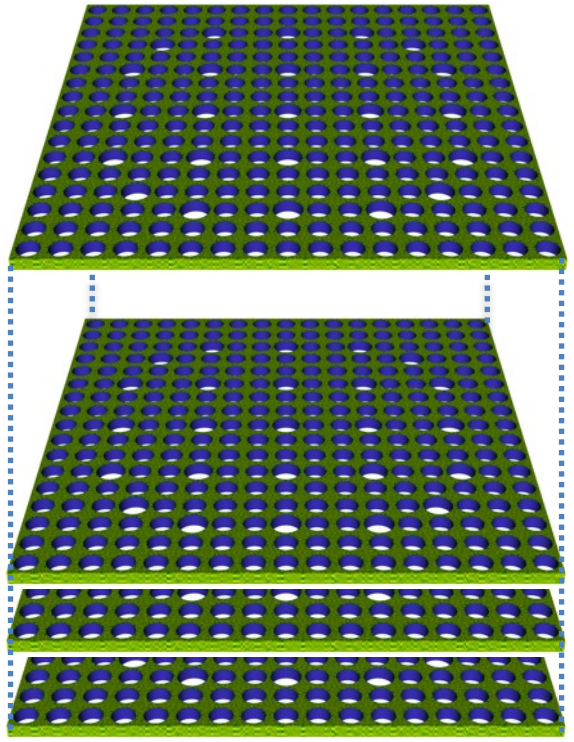
Spacer-Grid DNS Performance on Mira vs. Summit, $E = 3235953$, $N = 7$, $n = 1.10B$

System	Code	Device	Node	Rank	R/N	E/Rank	n/Rank	$t_{step}(s)$	Eff
Mira	Nek5000	CPU	8192	262144	32	12	4116	6.90e-01	100
Summit	Nek5000	CPU	38	1596	42	2027	695446	1.68e+01	100
			76	3192	42	1013	347723	0.50e+01	106
			152	6384	42	506	173861	0.23e+01	114
			304	12768	42	253	86930	0.11e+01	120
Summit	NekRS	GPU	38	228	6	14193	4.8M	2.75e-01	100
			60	360	6	8988	3.0M	1.92e-01	90
			76	456	6	7096	2.4M	1.64e-01	84
			98	588	6	5503	1.8M	1.39e-01	77

- ***Code is running fastest at strong-scale limit (large processor count, parallel efficiency~0.8)***
- ***For Navier-Stokes, Summit is faster at the strong-scale limit than Mira (surprisingly).***

ExaSMR: 17x17 Rod-Bundle Performance

Misun Min, Yu-Hsiang Lan ANL



Performance on Full Summit

Strong- and Weak-Scaling on ORNL Summit

NekRS Strong Scale: Rod-Bundle, 200 Steps						
Node	GPU	E	n	n/P	$t_{\text{step}}[s]$	Eff
1810	10860	175M	60B	5.5M	1.85e-01	100
2536	15216	175M	60B	3.9M	1.51e-01	87
3620	21720	175M	60B	2.7M	1.12e-01	82
4180	25080	175M	60B	2.4M	1.12e-01	71
4608	27648	175M	60B	2.1M	1.03e-01	70

NekRS Weak Scale: Rod-Bundle, 200 Steps						
Node	GPU	E	n	n/P	$t_{\text{step}}[s]$	Eff
87	522	3M	1.1B	2.1M	8.57e-02	100
320	1920	12M	4.1B	2.1M	8.67e-02	99
800	4800	30M	10B	2.1M	9.11e-02	94
1600	9600	60M	20B	2.1M	9.33e-02	92
3200	19200	121M	41B	2.1M	9.71e-02	88
4608	27648	175M	60B	2.1M	1.03e-01	83

NekRS on Summit: 352,000-Pebble Bed – 27648 V100s

Y.Lan, M.Min, E. Merzari

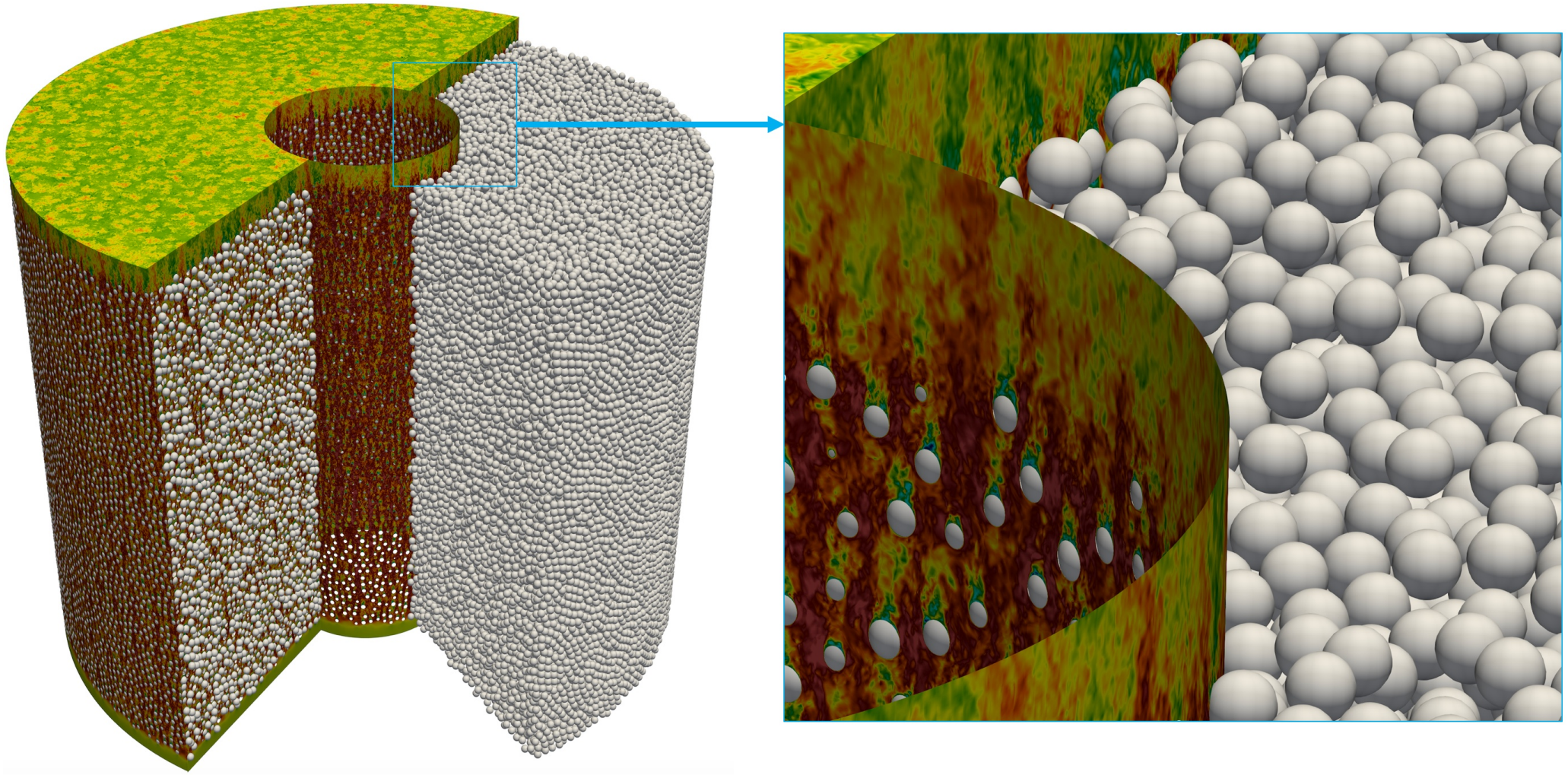


Figure 8: Turbulent flow in an annular packed bed with $\mathcal{N} = 352625$ spheres meshed with $E = 98,782,067$ spectral elements of order $N = 8$ ($n = 50$ billion gridpoints). This NekRS simulation requires 0.233 seconds per step using 27648 V100s on Summit. The average number of pressure iterations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

NekRS Per-Step Timing Breakdown: n=51B				
	pre-tuning		post-tuning	
Operation	time (s)	%	time (s)	%
computation	5.95-01	100	2.74-01	100
advection	2.91-02	5	2.25-02	8
viscous update	2.69-02	5	2.99-02	11
pressure solve	5.40-01	90	2.19-01	80
precond.	4.65-01	78	18.4-01	67
coarse grid	2.70-01	45	3.02-02	11
projection	3.39-03	1	6.05-03	2
dotp	2.46-02	4	9.60-03	4

Initial Formulation:

- 45% of the time in the coarse-grid solve
- Alleviate coarse-grid pressure by improving fine-level smoother, increasing dimension of pressure-projection space, adding GMRES, etc.

Result:

- 9X reduction in coarse-grid solve overhead
- 2.2X reduction in time per step
- ***Only 6 hours of run-time required for full core simulation.***

Major Algorithmic Variations, 352K Pebbles, P=27648								
Case	Solver	Smoother	L	N_q	Δt	v_i	p_i	t_{step}
(a)	FlexCG	1-Cheb-ASM:851	8	13	4e-4	3.6	22.8	.68
(b)	FlexCG	2-Cheb-Jac:851	8	13	4e-4	3.6	17.5	.557
(c)	”	2-Cheb-ASM:851	8	13	4e-4	3.6	12.8	.468
(d.8)	”	2-Cheb-ASM:8641	8	13	4e-4	3.6	9.1	.426
(d.L)	”	2-Cheb-ASM:8641	0–30	13	4e-4	3.6	5.6	.299
(e)	GMRES	”	30	13	4e-4	3.5	4.6	.240
(f)	”	”	30	11	8e-4	5.7	7.2	.376
(g)	”	”	30	11	8e-4	5.7	7.2	.361 (no I/O)

Summary of Current HPC Landscape - Scientific Computing Perspective

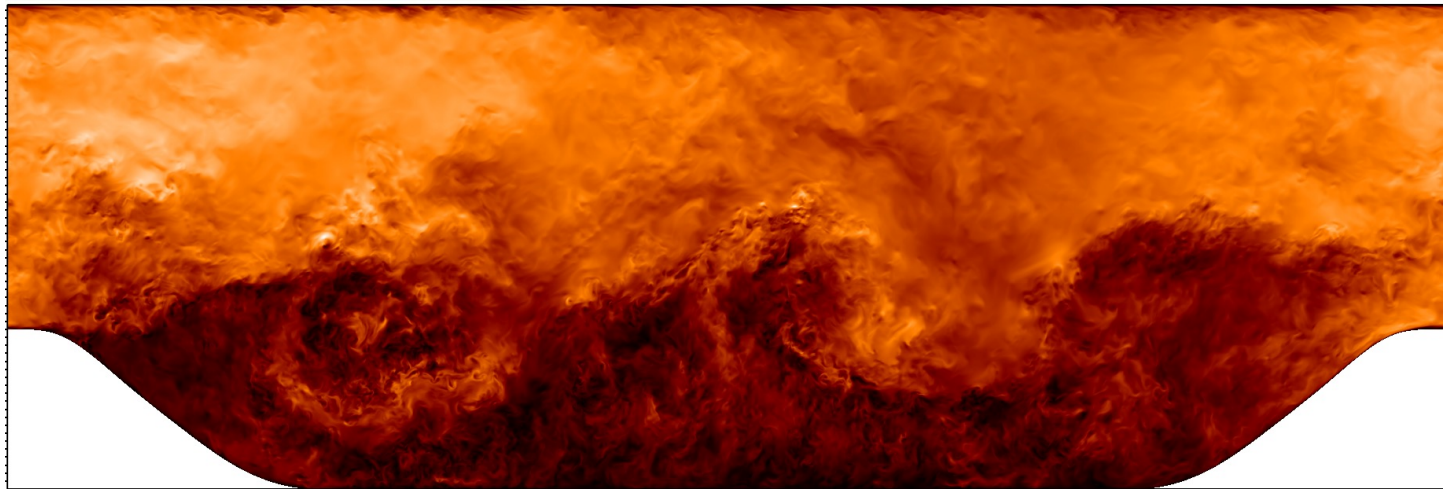
- Pre-exascale systems such as Summit are realizing 3X performance gains over strong-scale limit on Mira (i.e., 3X reduction in time-per-step).

- Summit enables much larger problems – 175M elements vs. 15M elements (factor of 11 in problem size)

- Summit is a 200 TFLOPS platform; 5X smaller than exascale.

- Exascale will not reduce time to solution unless your job is too large for Summit, but it will make large Summit runs look small, which is good for exploring parameter spaces.

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



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

Summit:
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*

(This result not a foregone conclusion…2020 BP Paper.)

Local Matrix-Free Stiffness Matrix in 3D

- For a deformed spectral element, Ω^e, never form local stiffness matrix.

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

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

- Through use of chain rule + GLL quadrature:
  - *Matrix-free operator evaluation.*
  - *Operation count is only $O(N^4)$ not $O(N^6)$ [Orszag '80]*
  - *Memory access is 7n (G_{rr}, G_{rs}, etc., are diagonal)*
  - *Work is dominated by matrix-matrix products involving D_r, D_s, etc.*

B – *diagonal mass matrix*

J – *Jacobian on GLL nodes*