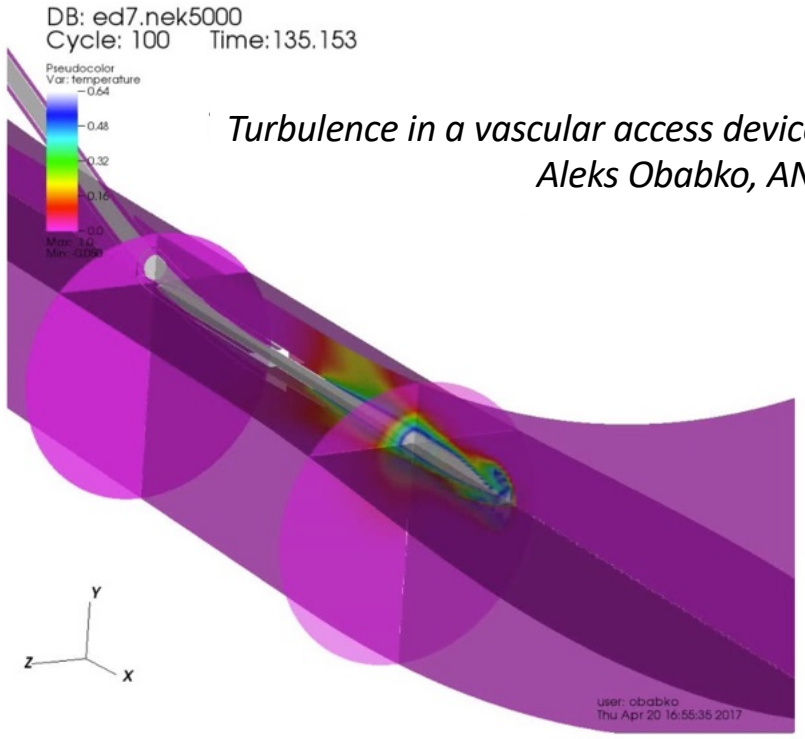
Exascale: A User's Perspective

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 the Center for Efficient Exascale Discretizations (CEED) under the DOE Exascale Computing Project.

Example: Compressed Turbulence (Nek5000-CPU)

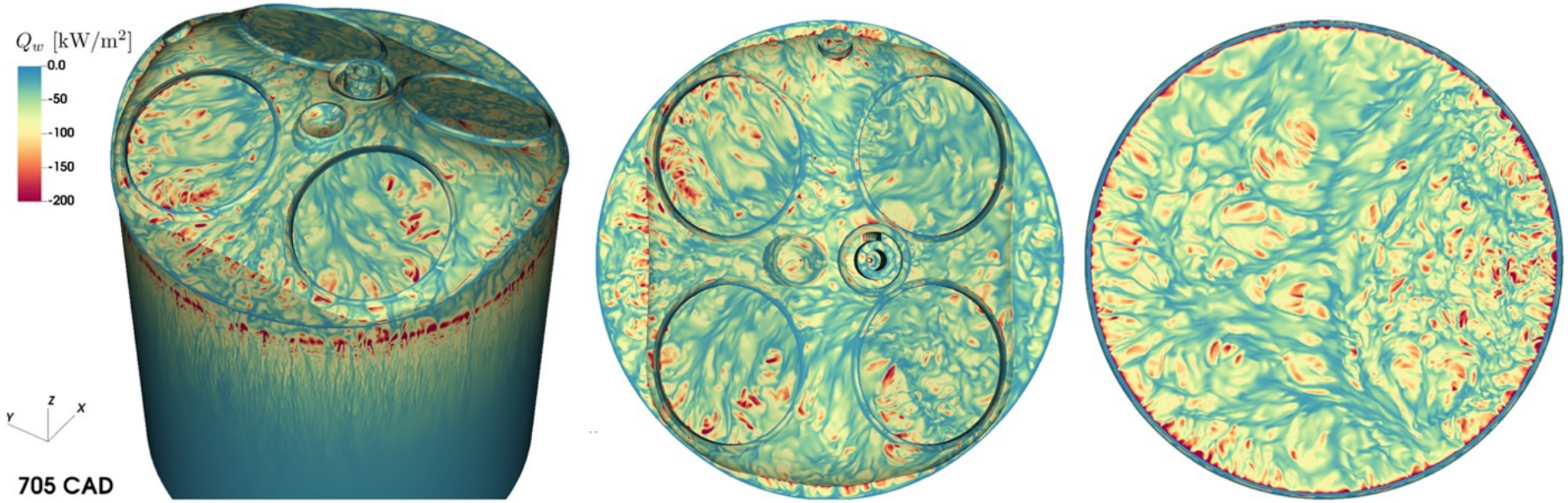
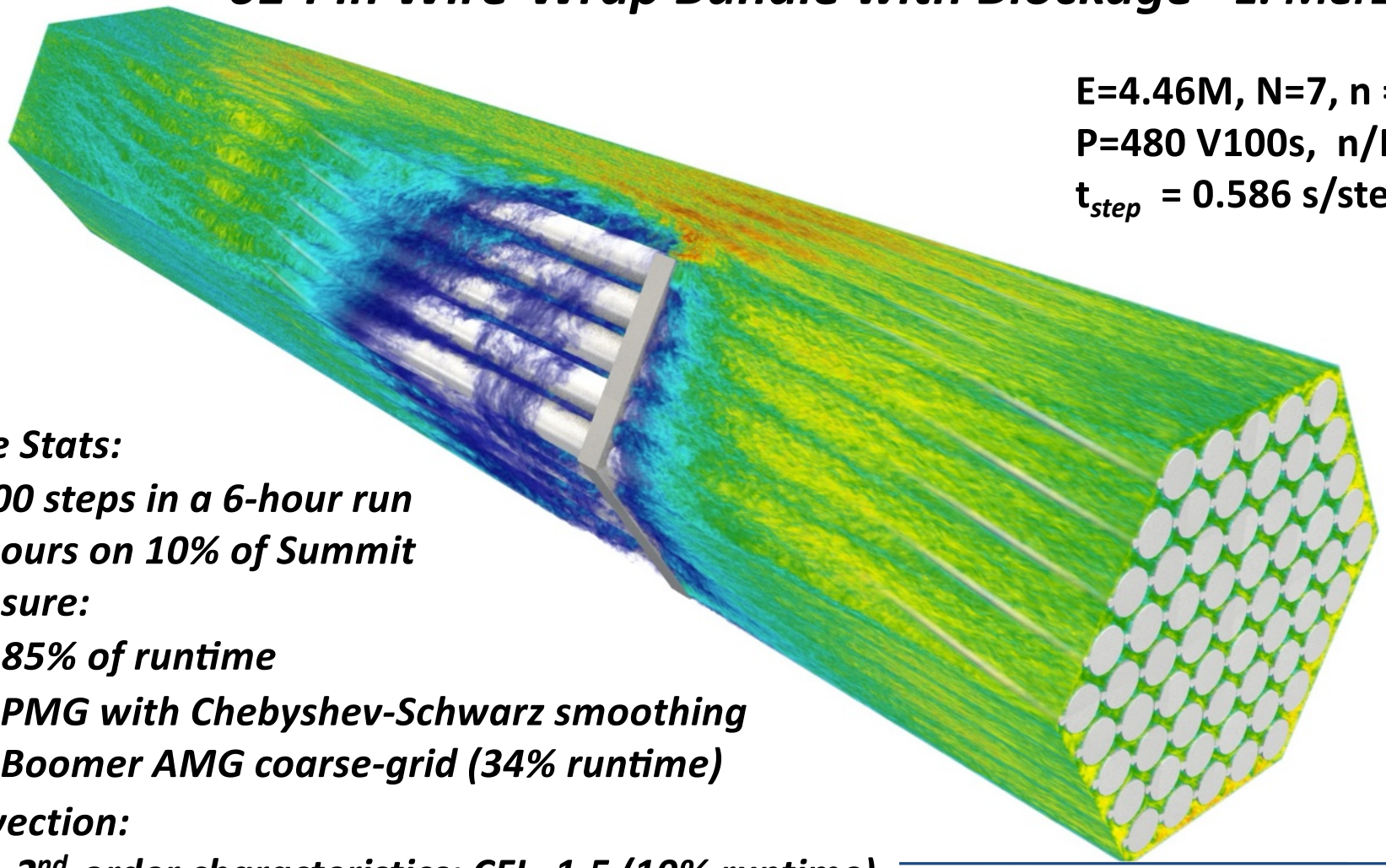


FIGURE 2.9 DNS of compression in an optical engine. Iso-contours of heat flux along the cylinder walls at 15^o bTDC, left-to-right: bird's eye view, cylinder head, piston.

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.*

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:***
  - ***2^{nd}-order characteristics: CFL=1.5 (10% runtime)***

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

Exascale Challenges - Scalability

- Key point:
  - Performance, $S_P = \eta P S_1$ ← *P-fold speed-up*
  - 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=1 Million. Why not?

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

SPECIAL ISSUE PAPER

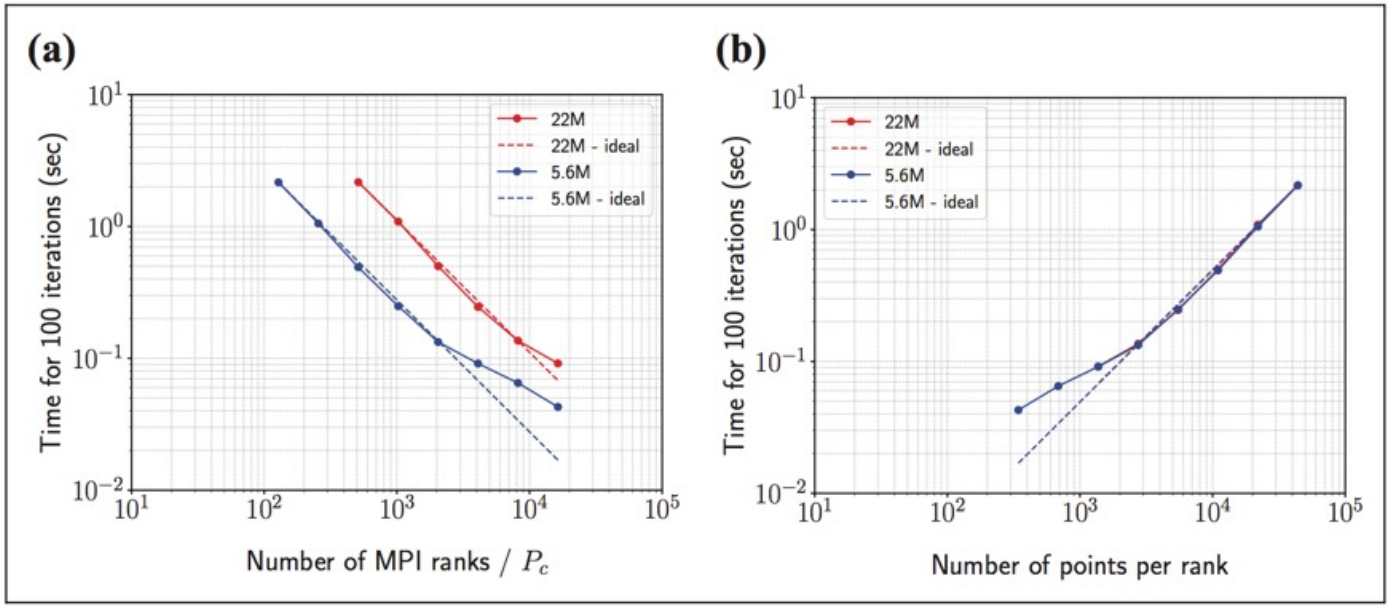
Toward message passing for a million processes: characterizing MPI on a massive scale blue gene/P

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

ECP-NASA meeting last year:

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

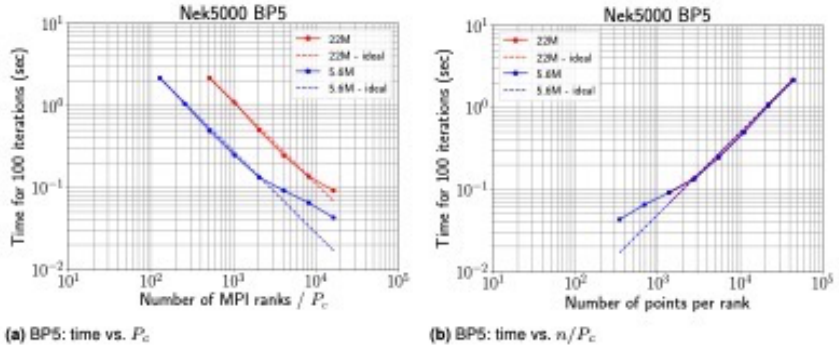
Parallelism: Stong-Scaling, Time to Solution, and Energy Consumption



Observations:

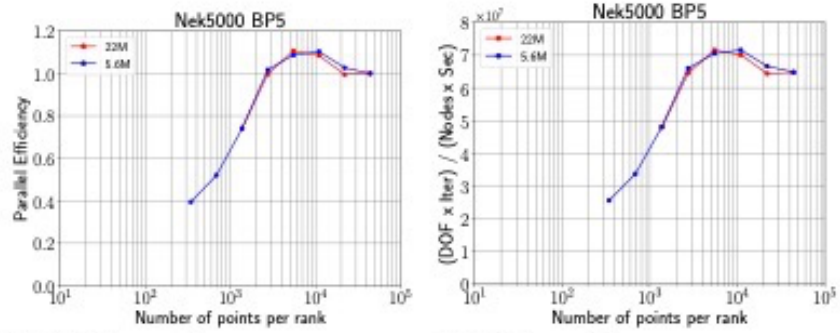
1. Time-to-solution goes down with increasing P, particularly for $\eta = 1$.
2. For $\eta = 1$, energy consumption ~ P x t_{sol} = *constant* — no penalty for increased P.
3. The red curve can use more processors than the blue. ***WHY?***
4. Why (for a problem of any size), do we find $\eta < 1$?
 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

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



(a) BP5: time vs. P_c (b) BP5: time vs. n/P_c

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



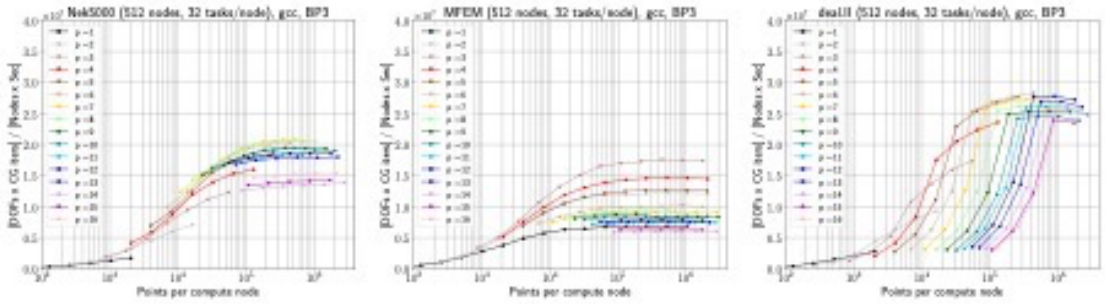
(a) BP5: efficiency vs. n/P_c (b) BP5: DOFs vs. n/P_c

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 $n/P_c \geq 2744$.

- ***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.***

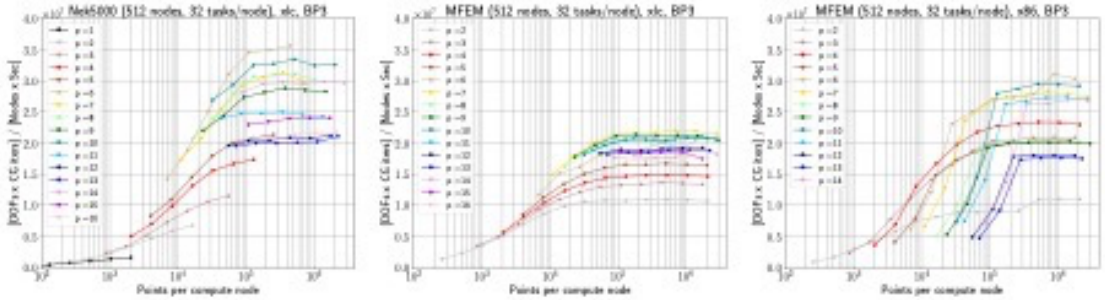
Fischer, Min, Rathnayake, Dutta, Kolev, Dobrev, Camier, Kronbichler, Warburton, Swirydowicz, and Brown. **Scalability of high-performance PDE solvers**. Int. J. of High Perf. Comp. Appl., 34(5):562–586, 2020.

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



(a) Nek5000 gcc (b) MFEM gcc (c) deal.II gcc

Figure 6. BP3 results with gcc compiler on 16,384 MPI ranks on 512 nodes of BG/Q; varying polynomial order ($p = 1, ..., 16$) and quadrature points ($q = p + 2$).



(a) Nek5000 xlc (b) MFEM xlc (c) MFEM xlc/x86

Figure 7. BP3 results with xlc compiler on 16,384 MPI ranks on 512 nodes of BG/Q; varying polynomial order ($p = 1, ..., 16$) and quadrature points ($q = p + 2$).

- *As part of CEED, we looked at six "bake-off" problems (BPs)*
- *Nek5000 / MFEM / deal.ii*
- *Up and to the left is better:*
  - *High throughput, low n/P*
- *Each code excelled on at least one BP*
- *These became standard figures of merit as new platforms / algorithms were introduced*

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:

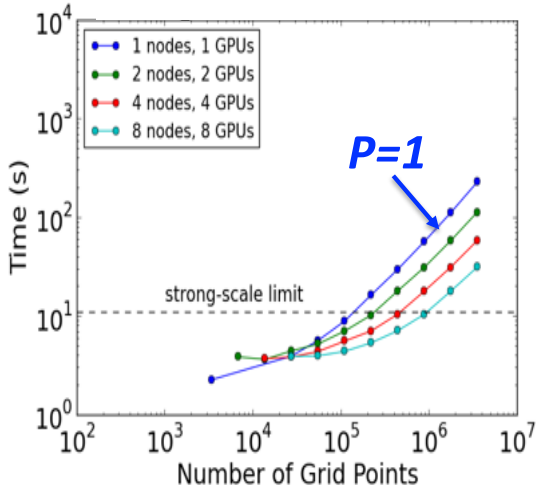
  At strong-scale limit (where users generally run),
  time-to-solution $\sim \frac{W}{0.8}\frac{n_{0.8}}{S_1}$

  W = number of flops per grid point
  $n_{0.8}$ = n/P, where $\eta \approx 0.8$
  S_1 = processing rate (GFLOPS) on a single rank
- To reduce time-to-solution, must not let the ratio $(n_{0.8}/S_1)$ 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*** $n_{0.8}$***?***

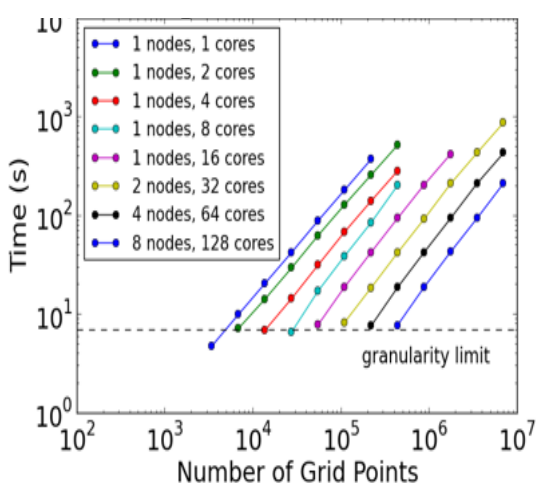
Influenced by OLCF Titan experience

Scalability: CPU vs GPU? ? How should we assess performance?

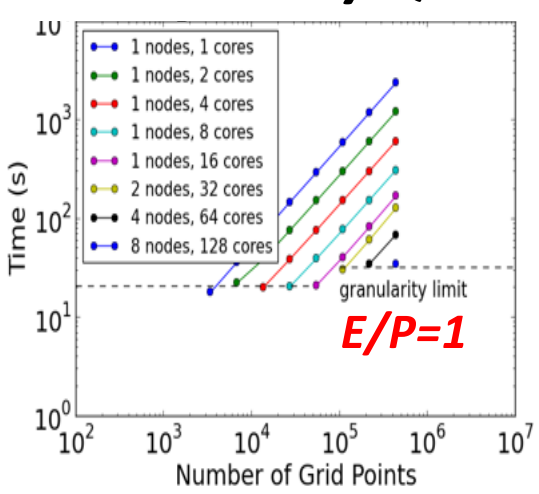
Titan GPU



Titan CPU



Vesta BG/Q

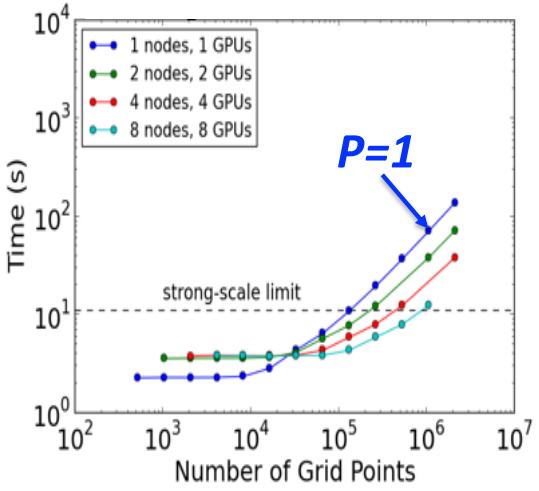


- ***GPU (K20) is faster than CPU, but performance falls off if GPU problem size is too small, even for P=1***
- ***Here, N=14th order elements - "coarse granularity"***

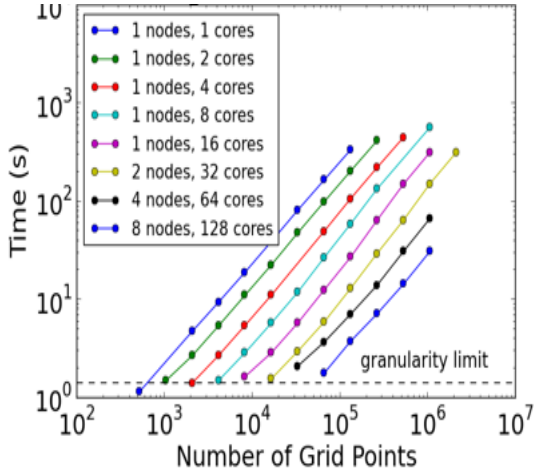
Otten, Gong, Mametjanov, Vose, Fischer, and Min, ***Hybrid MPI/OpenACC implementation for a high order electromagnetic solver on GPUDirect communication***, International Journal of High Performance Computing Applications, 30, No. 3, pp. 320–334, **2016**.

Scalability: CPU vs GPU? ? How should we assess performance?

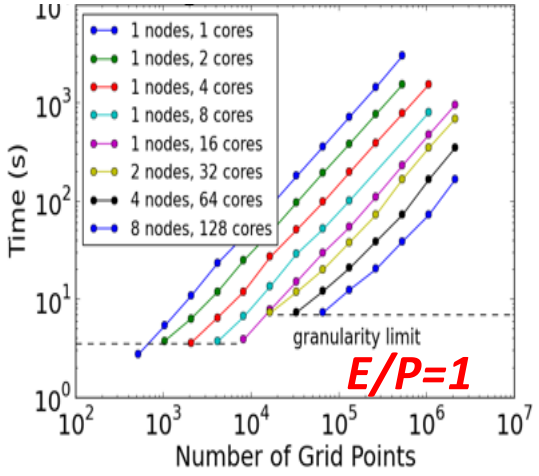
Titan GPU



Titan CPU

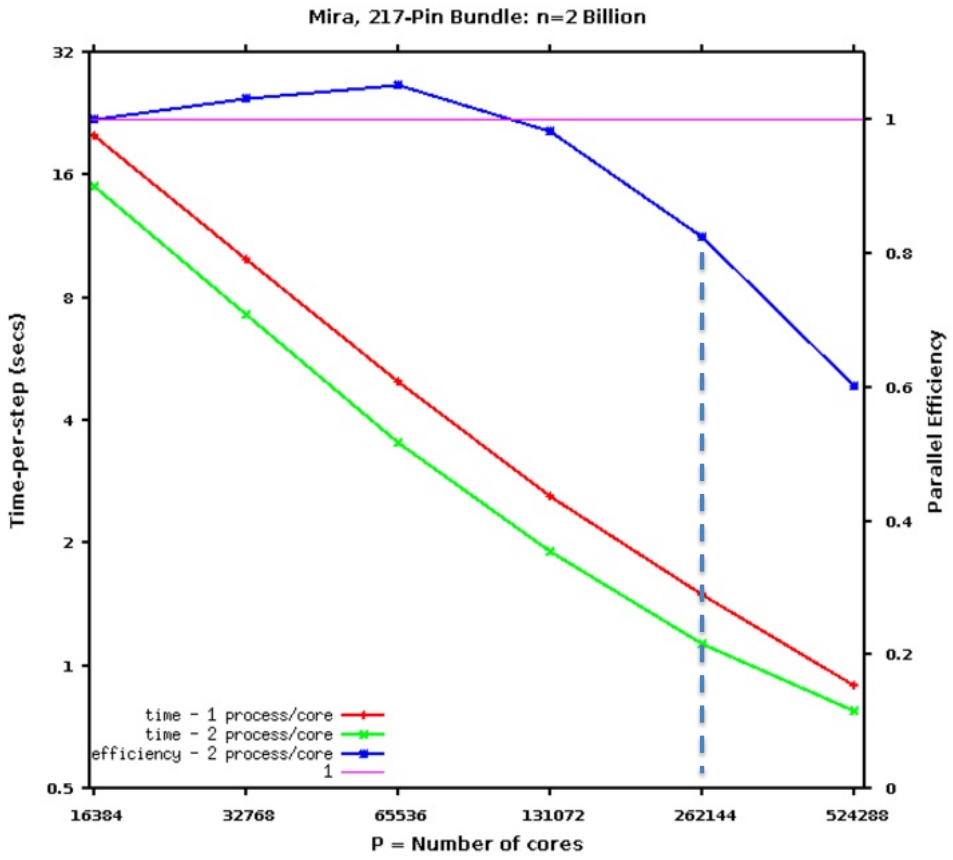


Vesta BG/Q



- ***GPU (K20) is faster than CPU, but performance falls off if GPU problem size is too small, even for P=1***
- ***Here, N=7th order elements: fine-grained, Titan CPU is faster***
- ***Perfect scaling → use CPUs. (WHY?)***

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

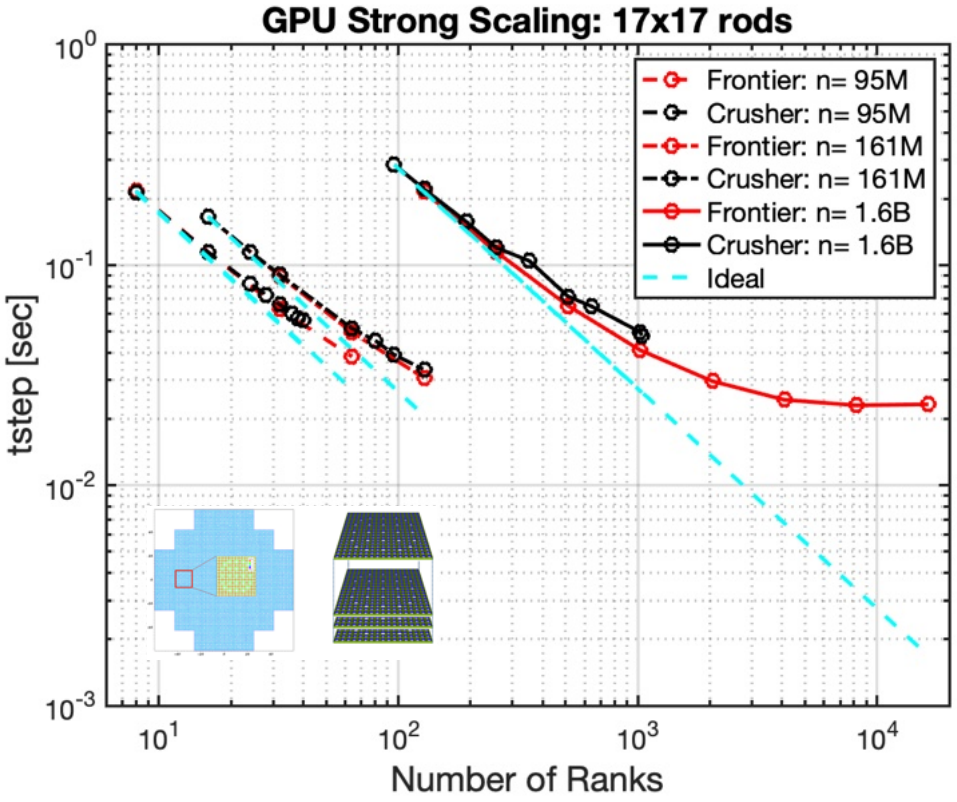


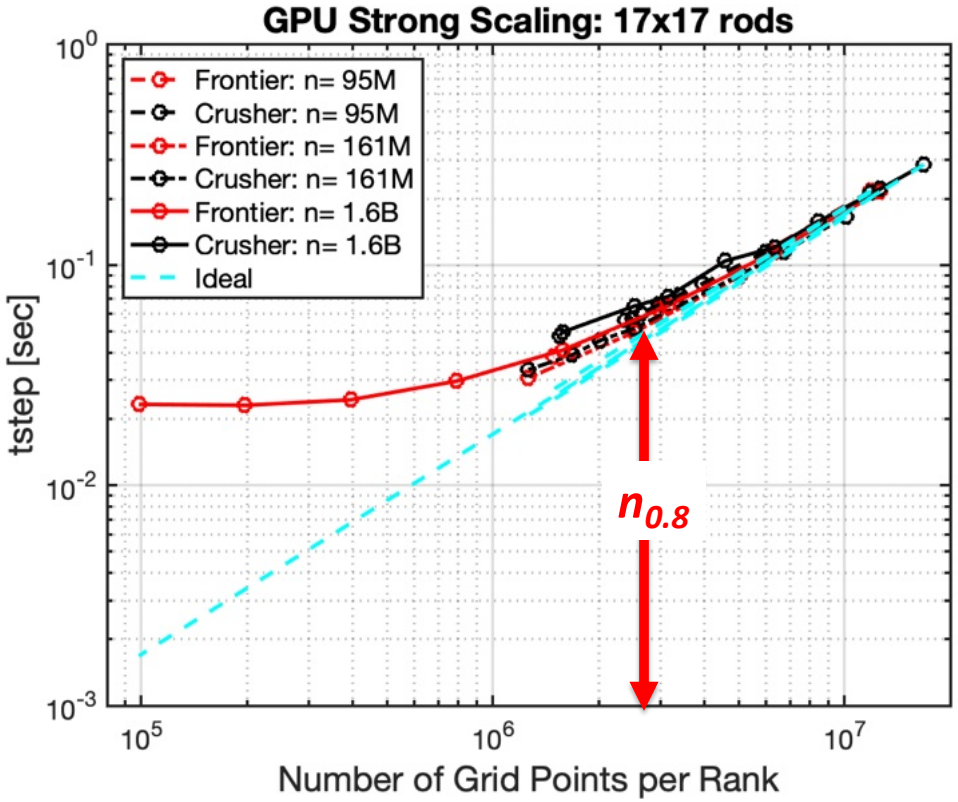
- Q: Do we use the 1-rank/core or 2-rank/core curve for strong-scale study?

  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





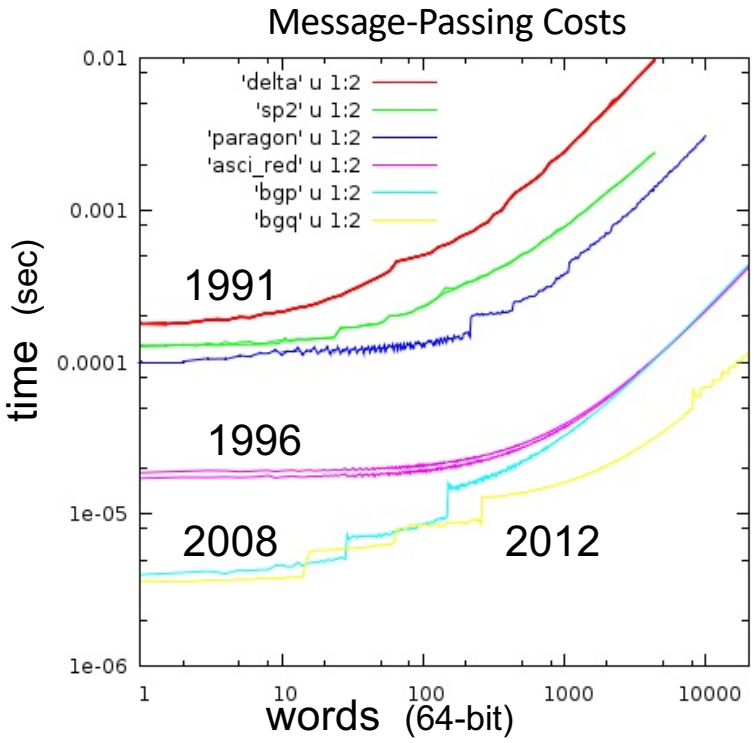
- ***Critical parameter: $n_{0.8}$ = number of points-per-rank to realize 80% efficiency.***
- ***This is where users will typically run and thus is the performance design point.***

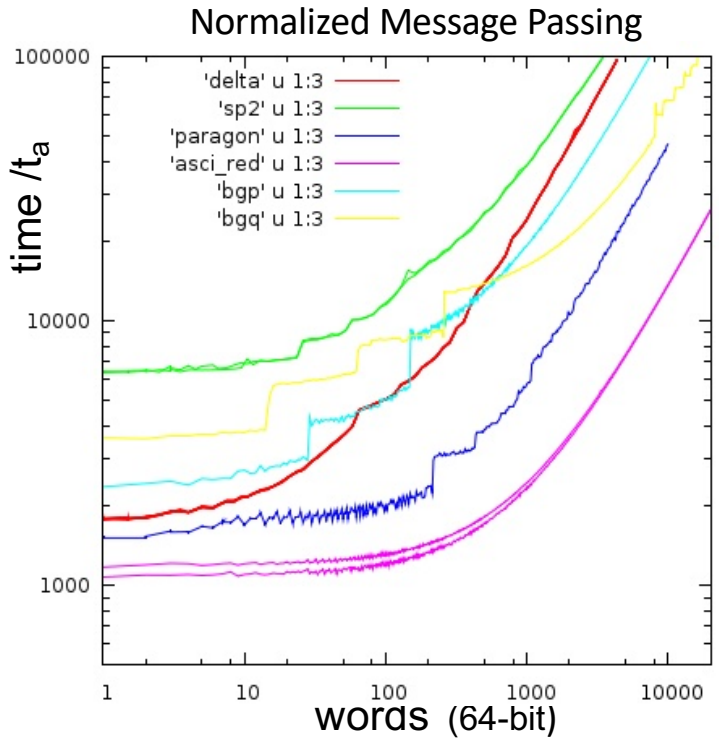
Addressing Efficiency Fall-Off

- 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

Eearly Ping-Pong Tests

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





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

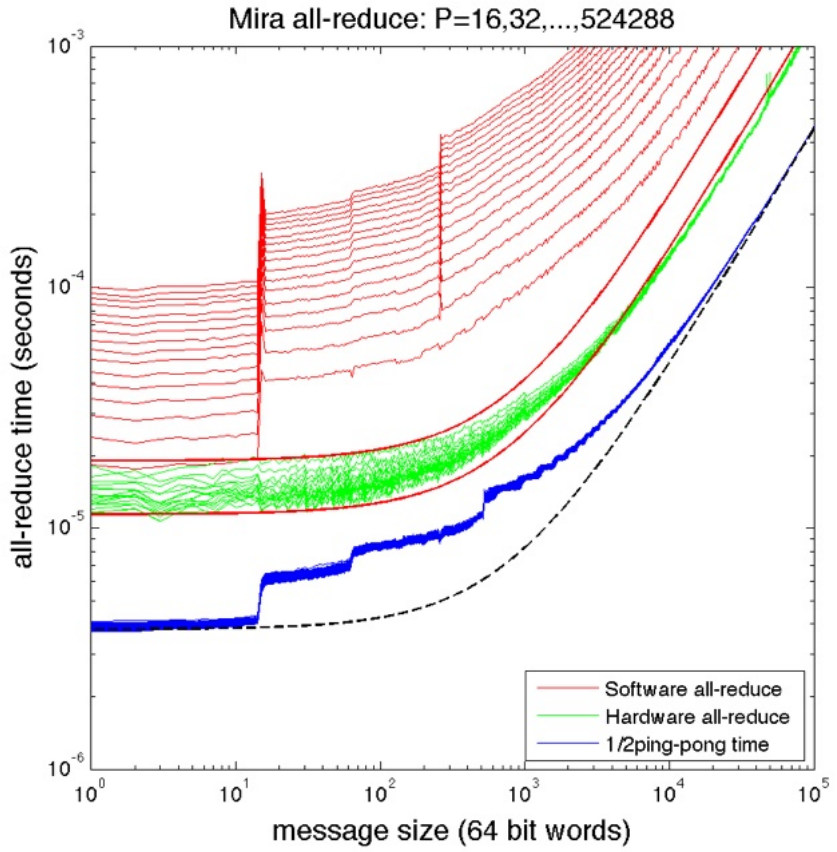
Thilina Rathnayake
University of Illinois
rbr2@illinois.edu

Matt Otten
Cornell University
mjo98@cornell.edu

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

Pavan Balaji
Argonne National Laboratory
balaji@anl.gov

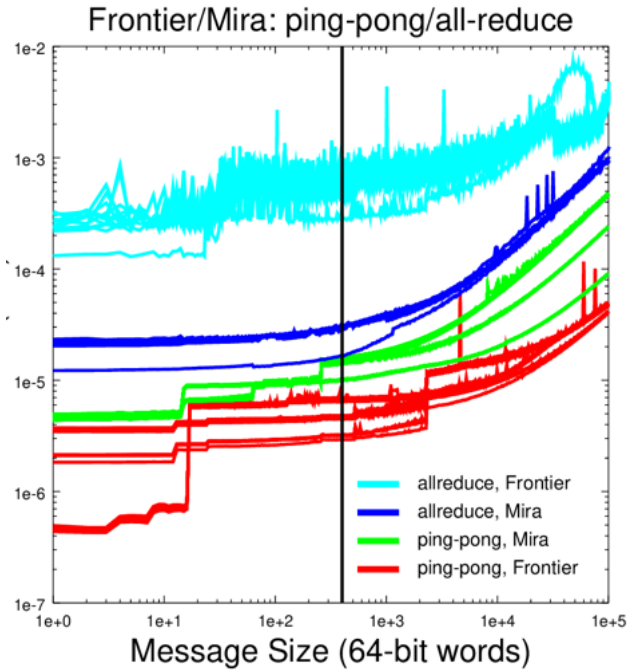
All-Reduce Cost Mitigation

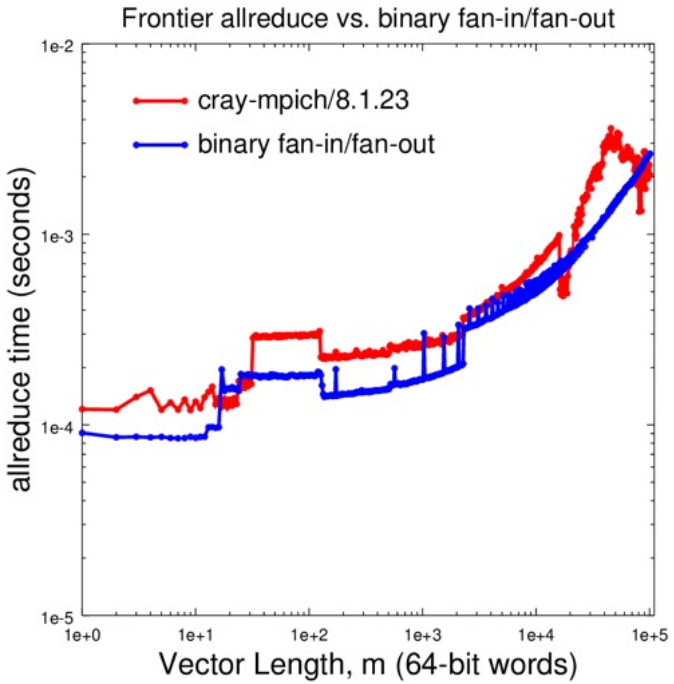


- *BG/Q (Mira, Sequoia, BG/P, BG/L)*
- *Isolated convex subnetworks - no traffic competing with User's resources*
- *18 cores per node - 16 compute, one for System, one for Yield*
- *All-reduce performed on NIC:*

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

Actually, there is a problem with Frontier 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…)

Exascale Challenges - Scalability & Portability

- ***We opted to use OCCA for portability:***
  - *Tim Warburton (V. Tech) a key team member*
  - *Long time developer of high performance kernels, esp. for high-order methods*
  - *Support for CUDA, HIP, OpenCL, DPC++*
  - *CS grad students are able to write backends*
- ***In an ideal world, we would have for accelerators what MPI did for the SPMD distributed-memory model - but not there yet.***

Highly-Tuned Kernels for Tensor Contractions, FP32 and FP64

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

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

From NekRS logfile, Perlmutter, SS10:

```
 Ax: N=7 FP64 GDOF/s=13.2 GB/s=1260 GFLOPS=2184 kv0
 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)
```

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

Advection Kernel (FP64)

New v16: use outer product for the advection operator to reduce SMEM access

- Performance improvement
  - A100: 1.35X, 4717 Gflop/s => 6375 Gflop/s
  - H100: 1.5X, 8312 Gflop/s =>12649 Gflop/s
  - Added chemistry field to the kernel

ellipticBlockPartialAxCoeffHex3D

- Added a v1 to reduce register pressure by utilizing multiple planes

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

- ***NekRS picks fastest kernel at setup***
- ***Never have performance regression***

Tuned Communication Options, FP32 and FP64

- Following the developments in Nek5000's gslib, there is an OCCA-based equivalent with several options for the gather-scatter communication.

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

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

- The test output also provides useful diagnostics.

From NekRS logfile, Perlmutter:

SS10:

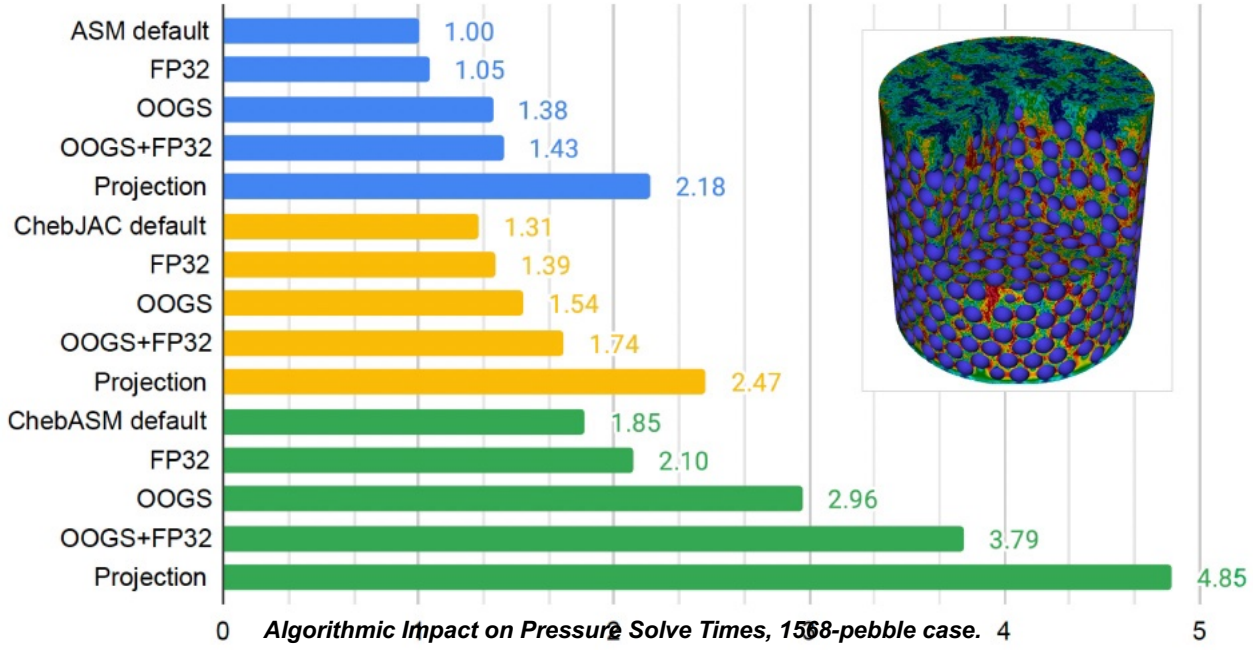
```
pw+device (MPI: 7.37e-05s / bi-bw:  54.5GB/s/rank)
pw+device (MPI: 1.75e-04s / bi-bw:  23.0GB/s/rank)
pw+device (MPI: 1.77e-04s / bi-bw:  22.7GB/s/rank)
pw+device (MPI: 1.76e-04s / bi-bw:  22.8GB/s/rank)
pw+device (MPI: 7.29e-05s / bi-bw:  55.2GB/s/rank)
pw+device (MPI: 7.29e-05s / bi-bw:  55.1GB/s/rank)
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

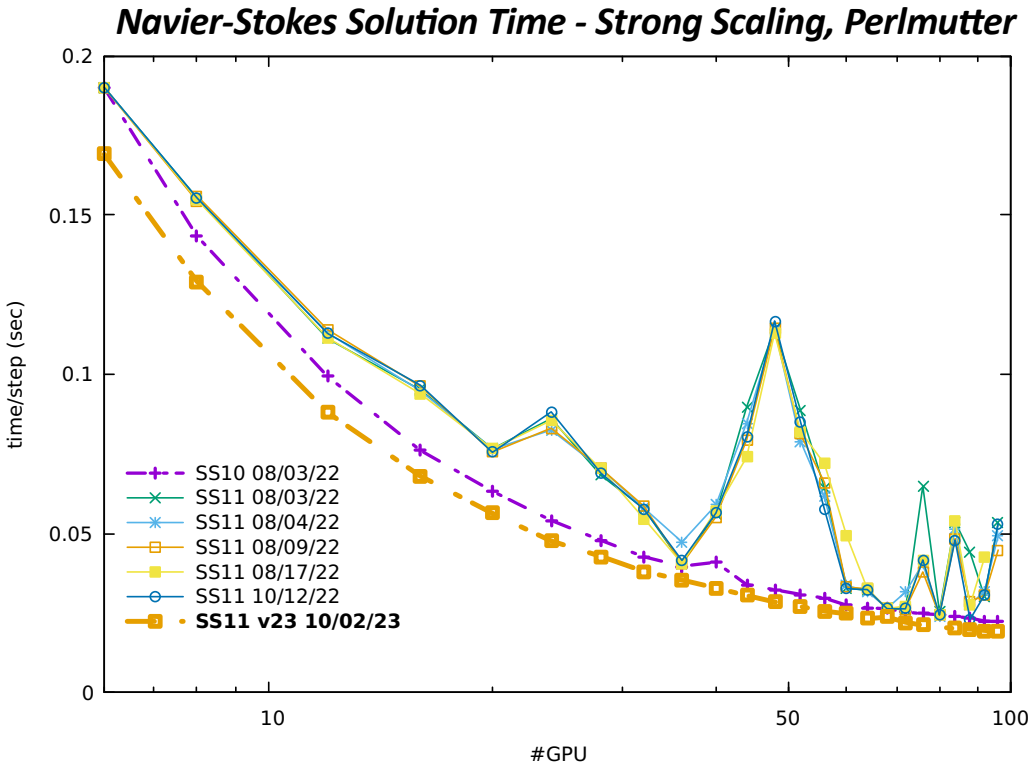
- Take-aways
  - overlapping communication/computation yields ~ 15% in pressure time
  - fp32 in preconditioner can yield 10-15%. Often, the fp32 advantage derives from reduced bandwidth demand on the network. *Q: Role of strong scaling?*



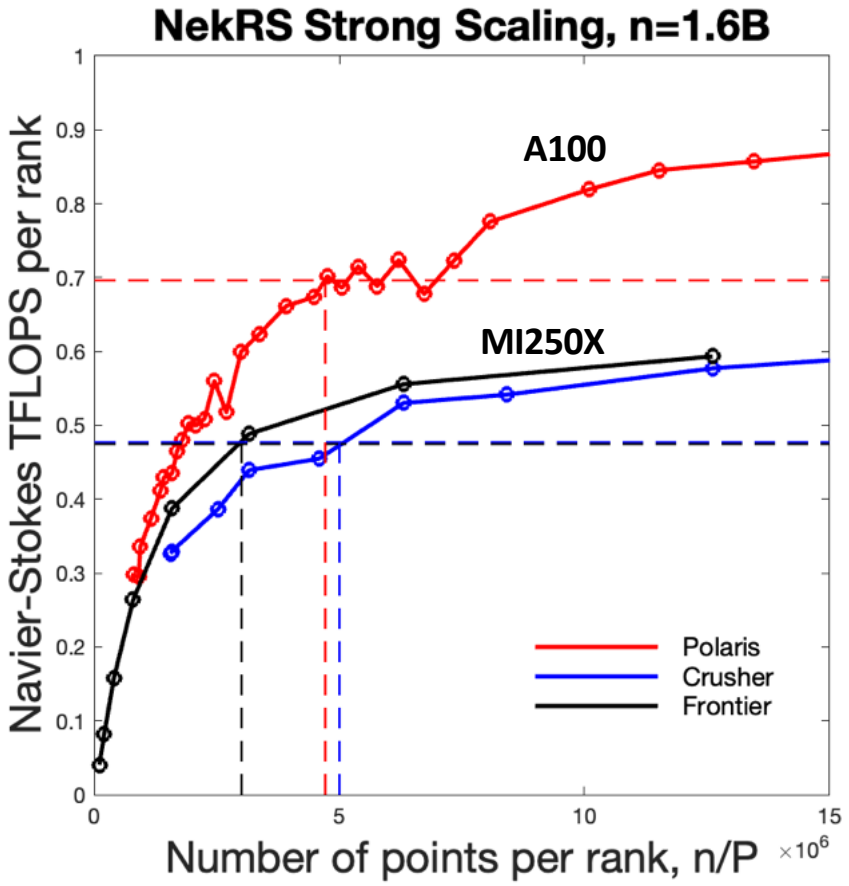
Algorithmic Impact on Pressure Solve Times, 1568-pebble case.

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



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

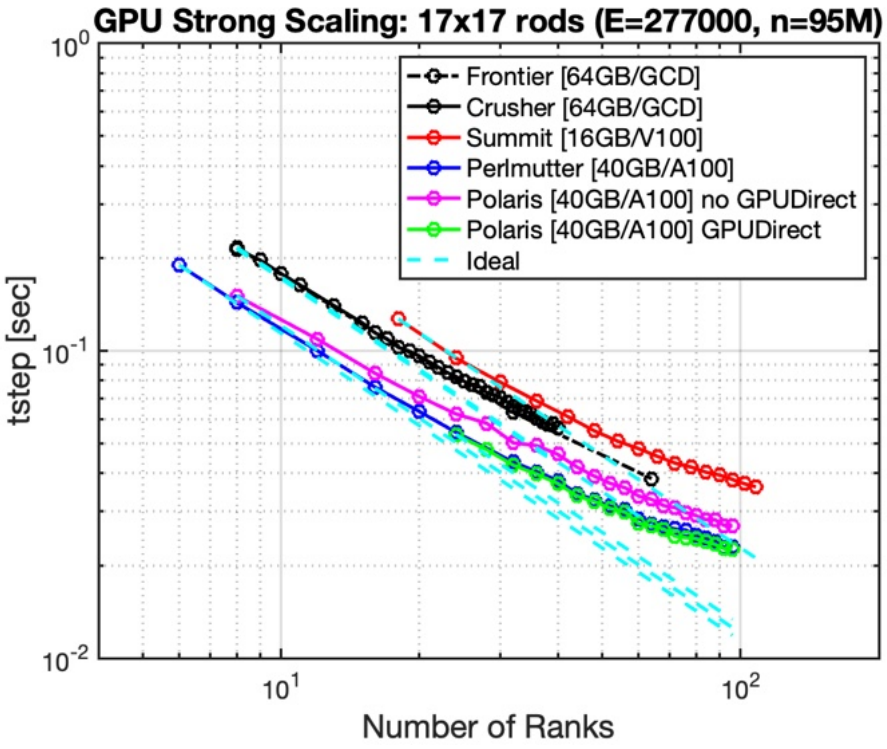


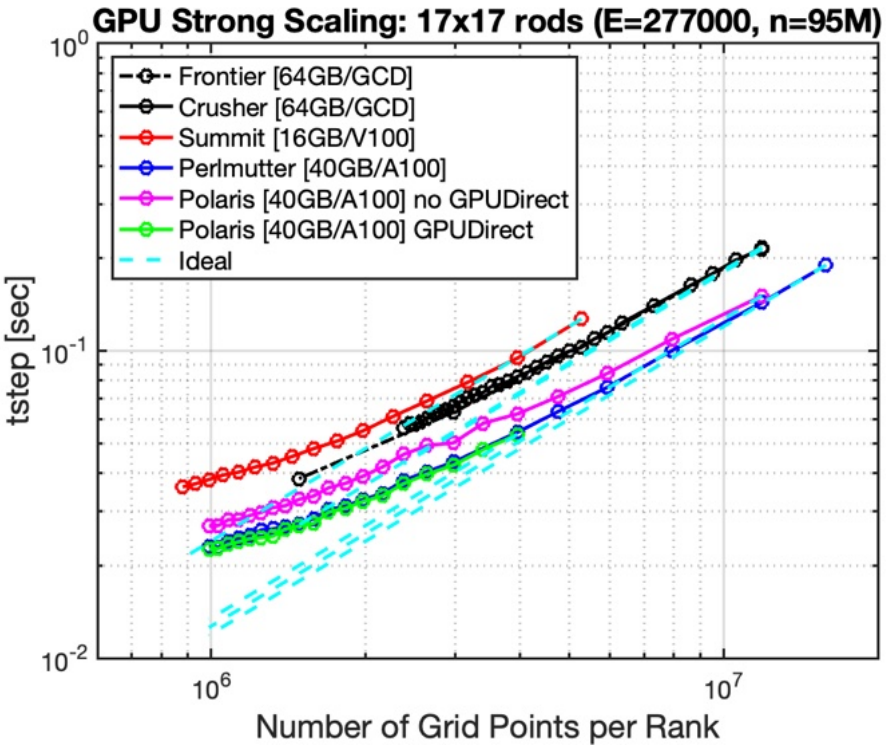
- *While the A100 has higher peak performance, its $n_{0.8}$ ~ 5M per GPU*
- *For Frontier (MI250X), $n_{0.8}$ ~ 3M per GCD*
- *At 80% efficiency, time to solution is actually lower (0.84) on Frontier than on Polaris because Frontier can use more ranks.*
- *Note that if we try to run on Polaris at ~ 100% efficiency the time to solution will be > 3 x 0.8 = 2.4x longer.*

  - 2.4 days, instead of 1 day.

Insight about Frontier Performance — *M. Min et al. 2022*

- ***Single GCD FLOP intensive kernels on par with A100***
- ***Communication-intensive phases are on par with Summit***

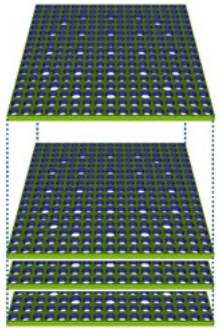




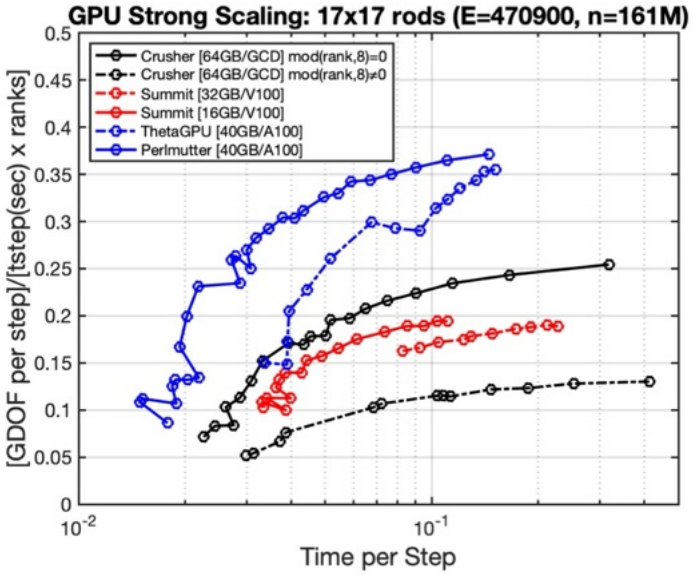
ExaSMR Timings

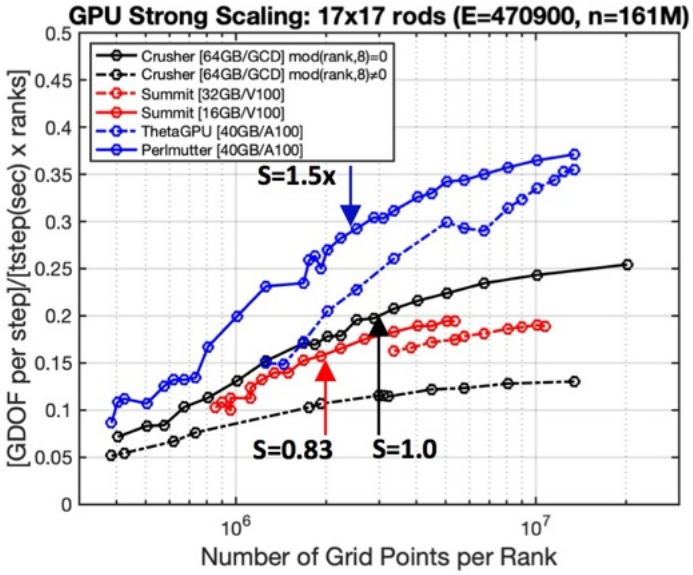
M.Min, Y.H. Lan, M. Phillips

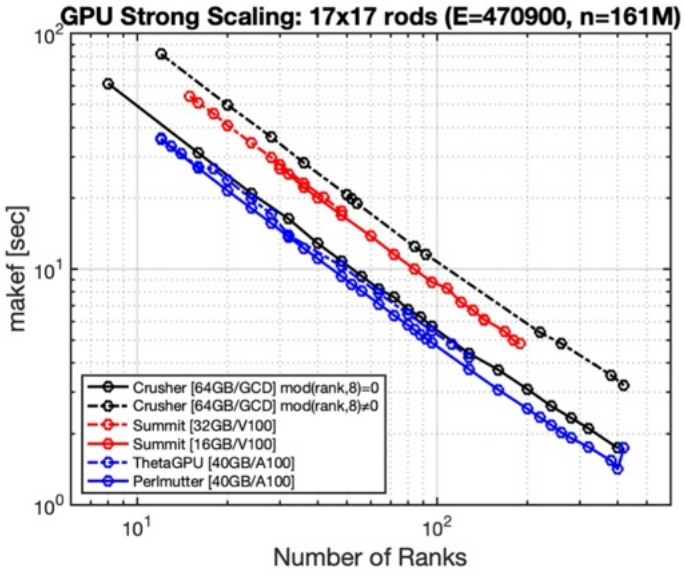
- ❑ Summit
- ❑ Crusher
- ❑ ThetaGPU
- ❑ Perlmutter

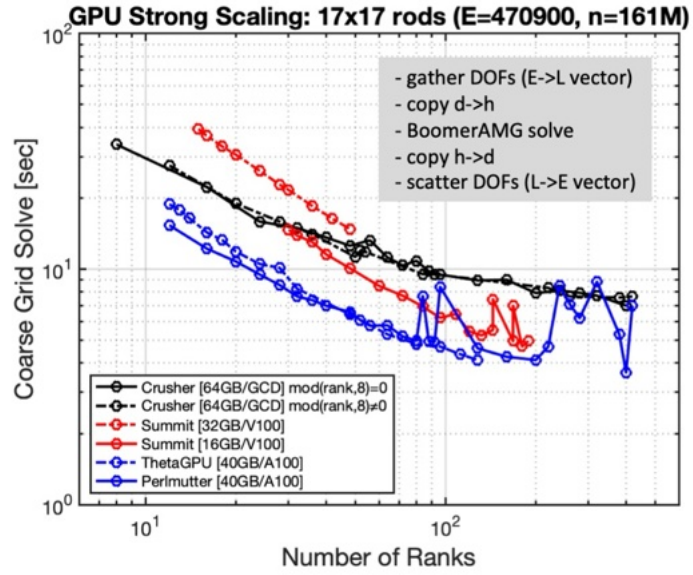


- 17x17 rod bundle
- Mesh:
  2D: E= 27700
  3D: E=27700 x 17 layers
- Resolution
  E=470,900, N=7
  n=161,518,700
- BDF3, CFL=0.67
- Re=5000

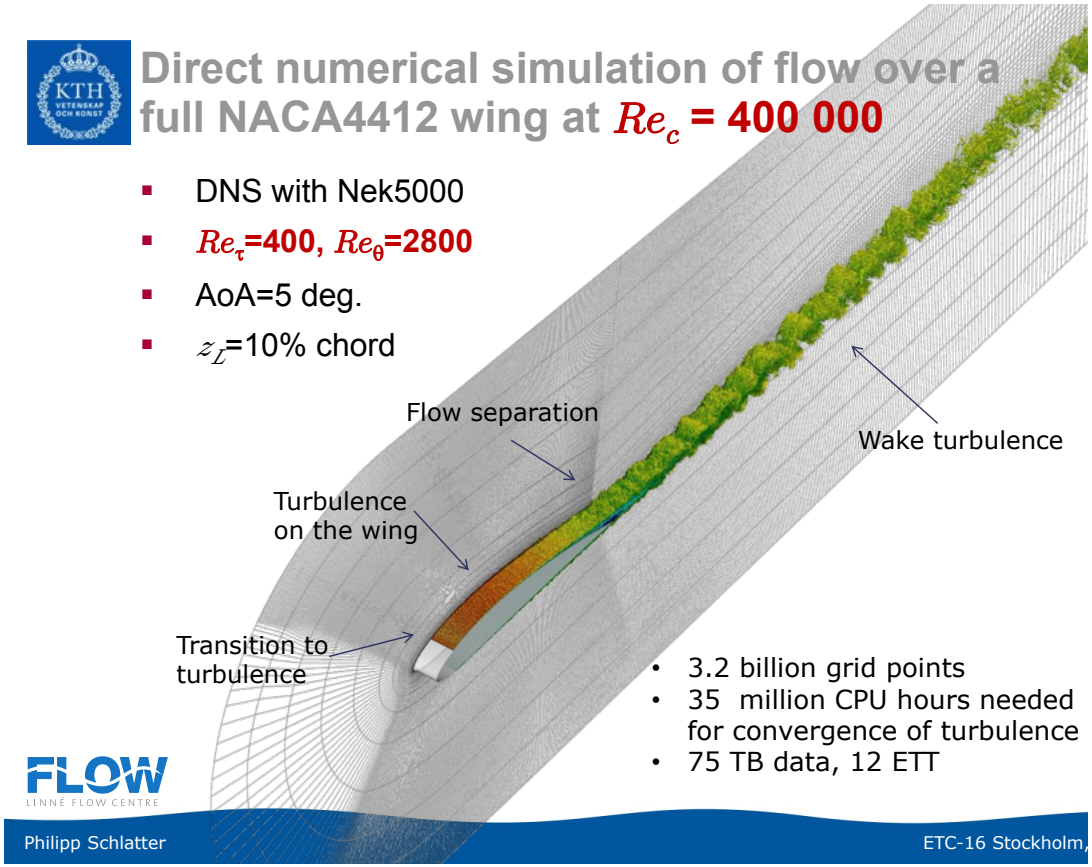








Answering a Common Question: How long will my job take?



- ❑ 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

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 ??***

  Inquiring users want to know!

Summit-Mira Comparison

Ramesh Balakrishnan ANL

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

Mira: *Nek5000*
P=524288 ranks (262144 core
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.)

Extreme Scalability: Full-Core Pebble Bed Simulations

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

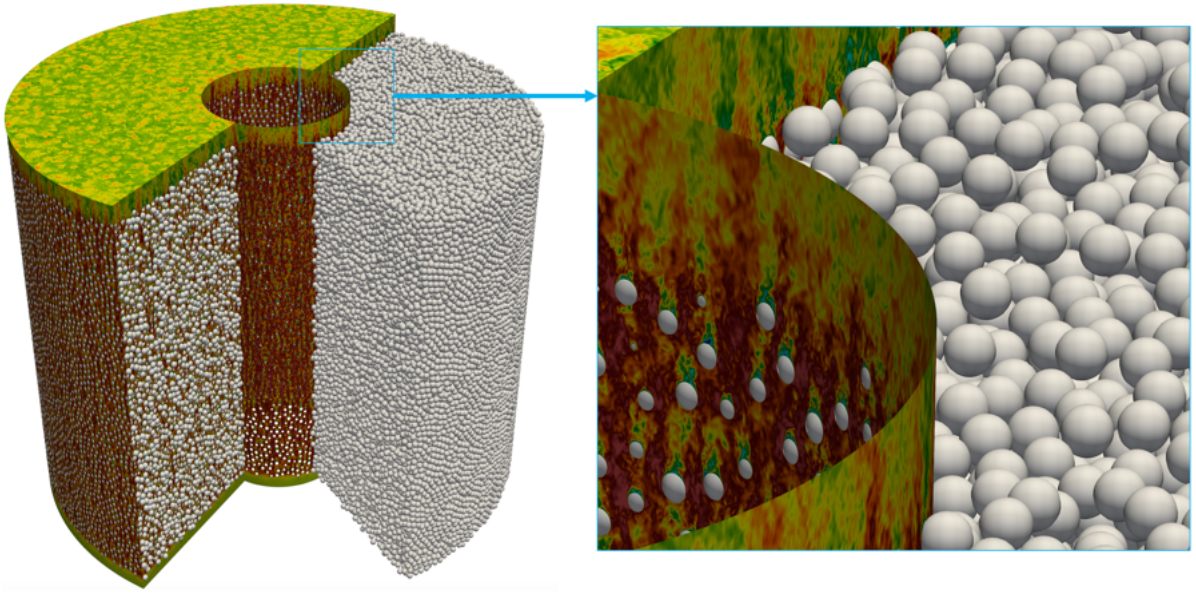


Figure 8: Turbulent flow in an annular packed bed with $\mathcal{N} = 352625$ spheres meshed with $E = 98,782,067$ spectral elements of order $N = 8$ ($n = 50$ billion gridpoints). This NekRS simulation requires 0.233 seconds per step using 27648 V100s on Summit. The average number of pressure iterations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

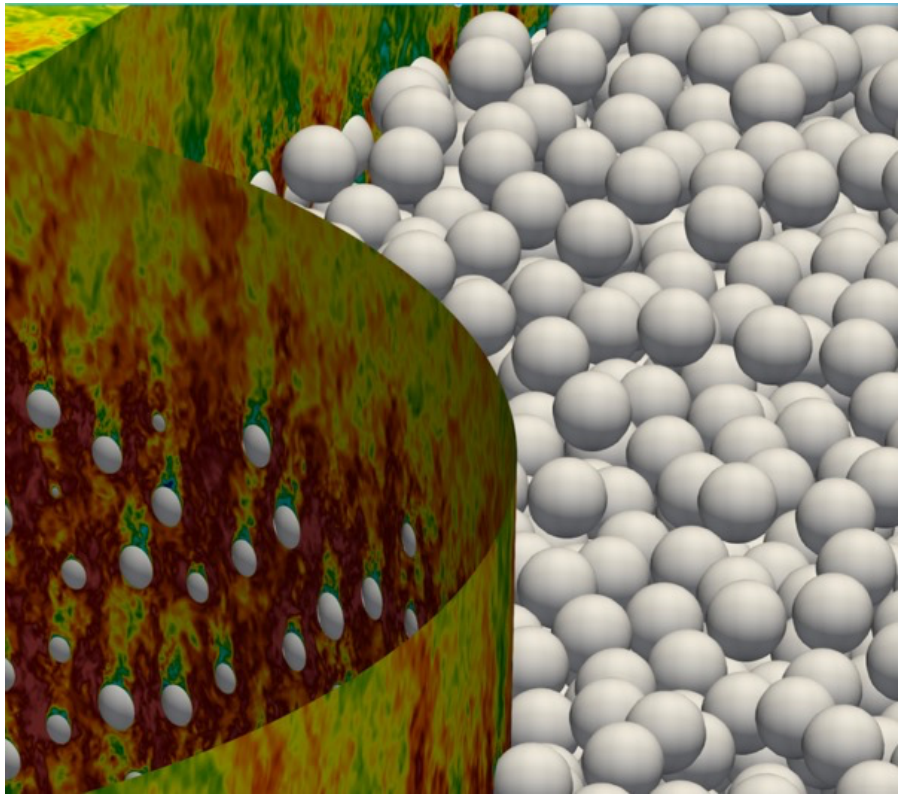
- *Turbulent flow in the interstitial region between the randomly-packed spheres.*

***Y. Lan, P. Fischer, E. Merzari, M. Min**: All hex meshing strategies for densely-packed spheres. Int. Meshing Roundtable, 2021.*

Net Improvements - Full Core Simulation

- Net reduction,

  t_{step} : 0.68 s → 0.24 s (effective 0.18 s)



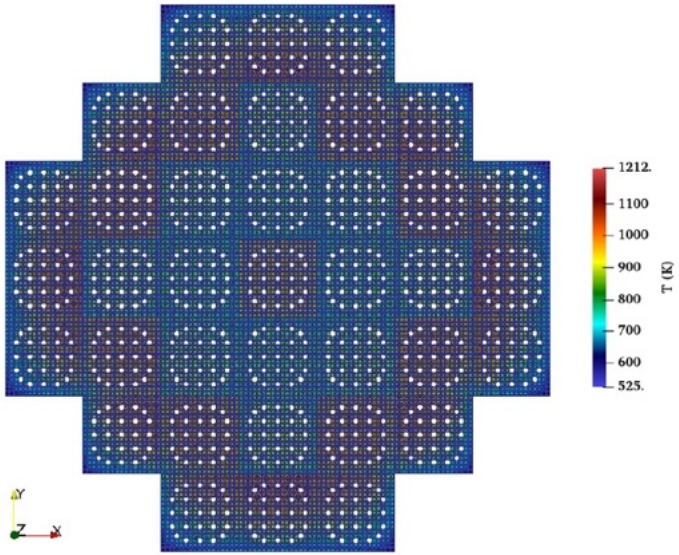
- With a 2X increase in step size (via characteristics), able to solve a ***full flow-through in just 6 hours on Summit***, which is a significant achievement compared to pre-ECP capabilities, both in size and speed.

- Record problem size on Mira was E=15M
- Here, E=98M on Summit and new runs on Frontier are at E=1.6B (N=7 or 9)

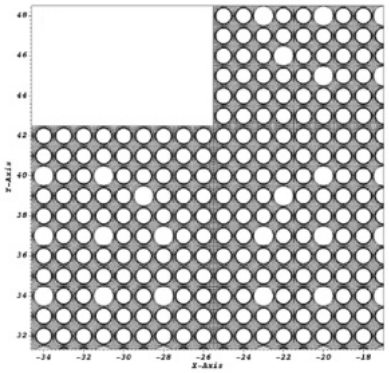
All of Frontier: SMR Full-Core Model

- The Shift model includes the division of each fuel pin into three radial rings as well as the modeling of gap and cladding regions.
  - The model includes both a top plug region and a bottom plug region a swell a a gas-filled plenum within each fuel pin.
  - An axial reflector of water 20 cm in height is present above and below the core.

- 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.)
  - Each assembly comprises E = 27,700 fluid elements per two dimensional layer and E = 31,680 solid elements per 2D layer. The full core mesh comprises 37 assemblies.
  - 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.

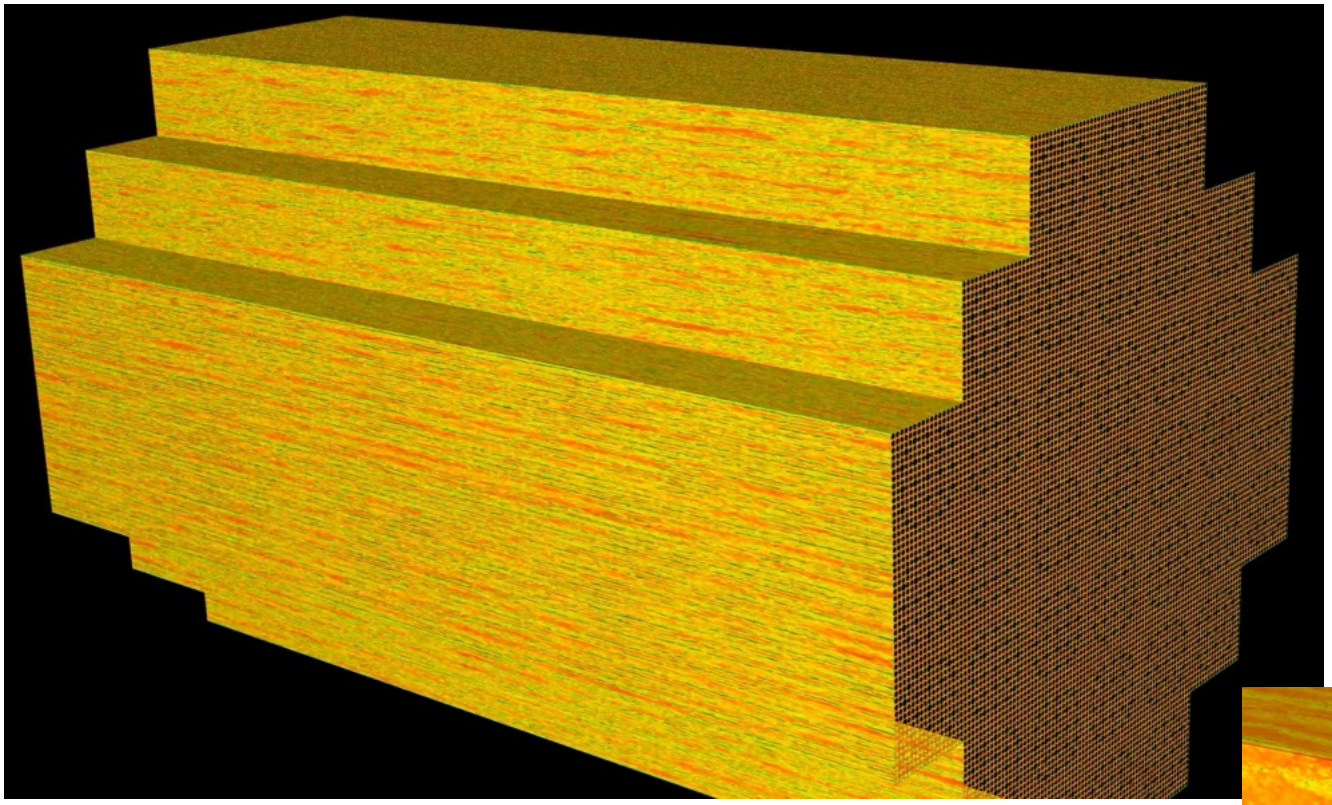


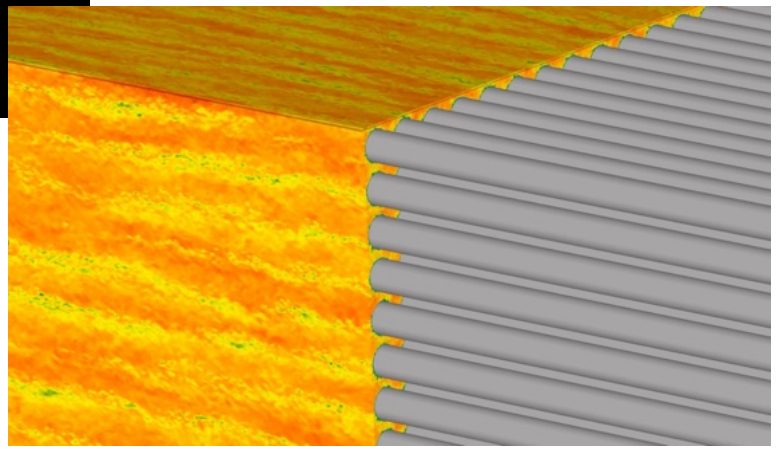
Temperature distribution in the core



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.





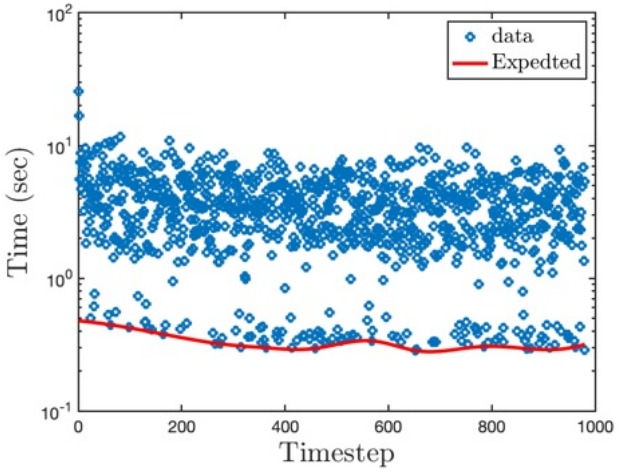
ExaSMR

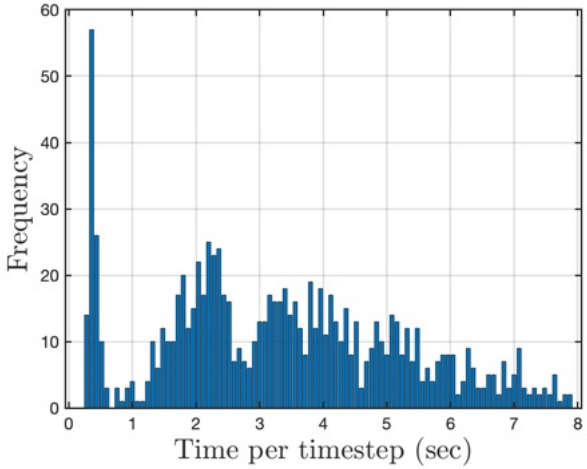
Collaboration with ExaSMR: 9000-Nodes Frontier Runs (72,000 GCDs)

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.

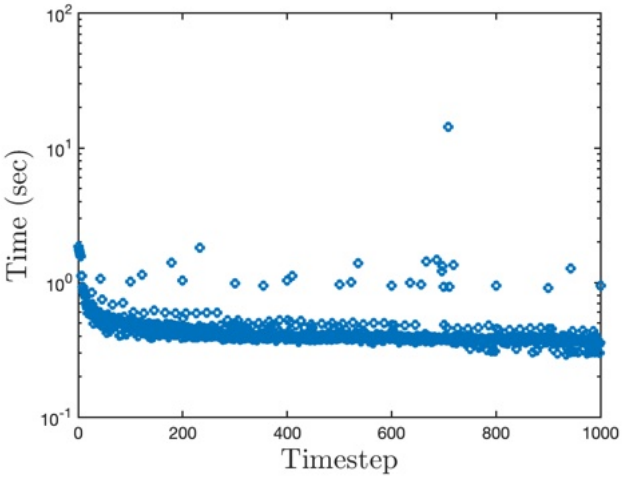
ExaSMR

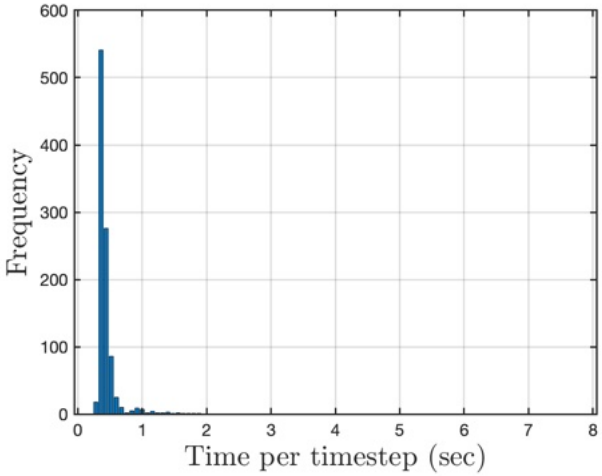
Collaboration with ExaSMR: 9000-Nodes Frontier Runs (72,000 GCDs)

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*





Progress Towards Exascale

- Users are observing ~3X increase at Strong-Scale limit (0.15 s/step vs 0.5 s/step)
  - Strong-scaling impacted by kernel launch overhead as well as MPI. (Some promise on both fronts)
- HUGE problems (billions of elements vs 10s of millions)
- Portable performance: OCCA
- ***Sustaining 930 GFLOPS per A100 on Polaris (counting fp32 as a 1/2 flop)***
- ***Quote from Elia Merzari :*** *"Once students switch to GPU variant, they never go back."*
- ***Bake-Offs*** have been a very good mechanism to increase productivity.

What Might We Do for the Future?

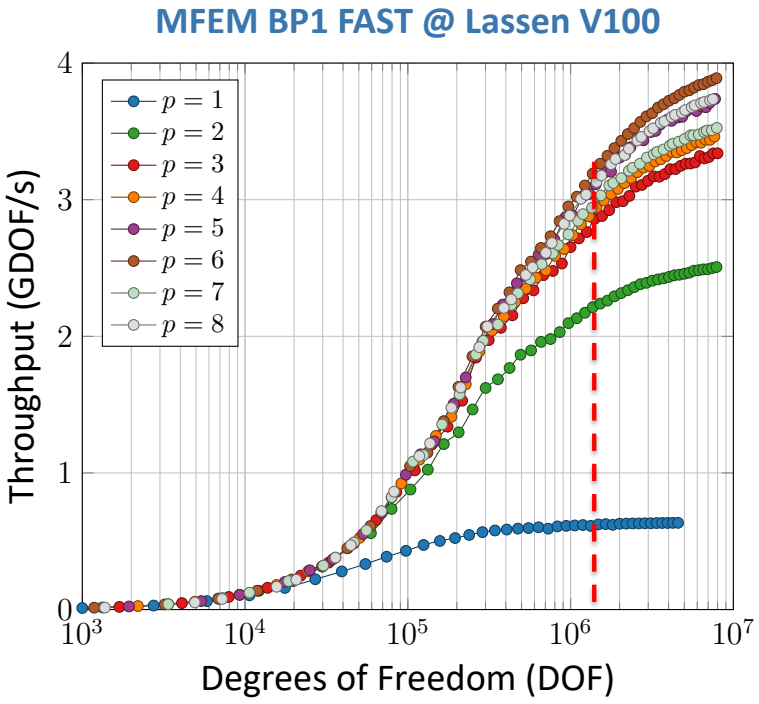
- Increase strong-scalability
- How?
- Two main issues with GPUs:
  - Reduce kernel launch overhead
  - Reduce message-passing latency
    - Convex subnetworks
    - Hardware collectives
    - One-sided message exchanges (Thomas Gillis)

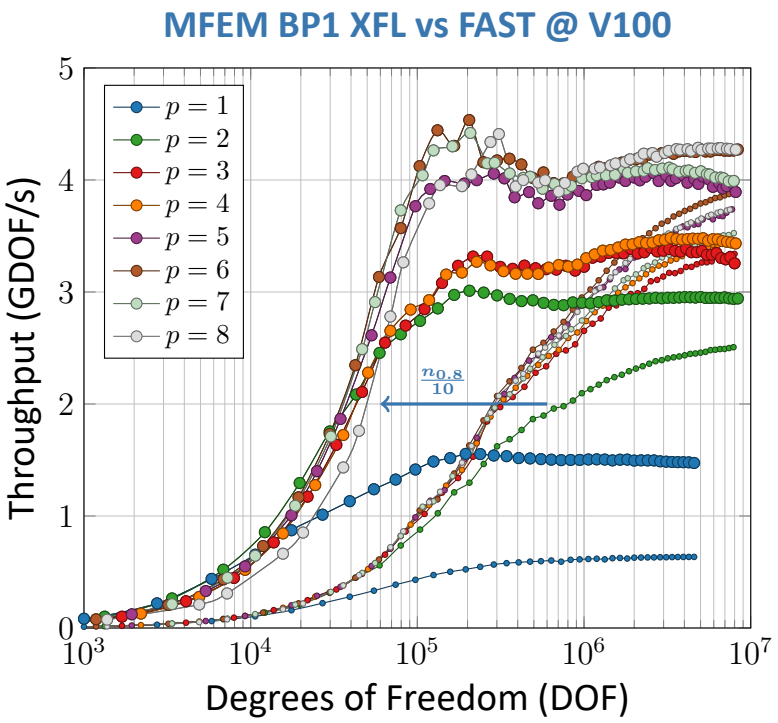
$n_{0.8}$ on a Single GPU (V100)

John Camier, LLNL (CEED MS37, 2022)

CEED Bake-Off BP1:

Throughput vs. Local Problem Size *(Up and to the left is good.)*





What Might We Do for the Future?

- Increase strong-scalability
- How?
- Two main issues with GPUs:
  - Reduce kernel launch overhead
  - Reduce message-passing latency
    - Convex subnetworks
    - Hardware collectives
    - One-sided message exchanges (Thomas Gillis)

Thank You for Your Attention!