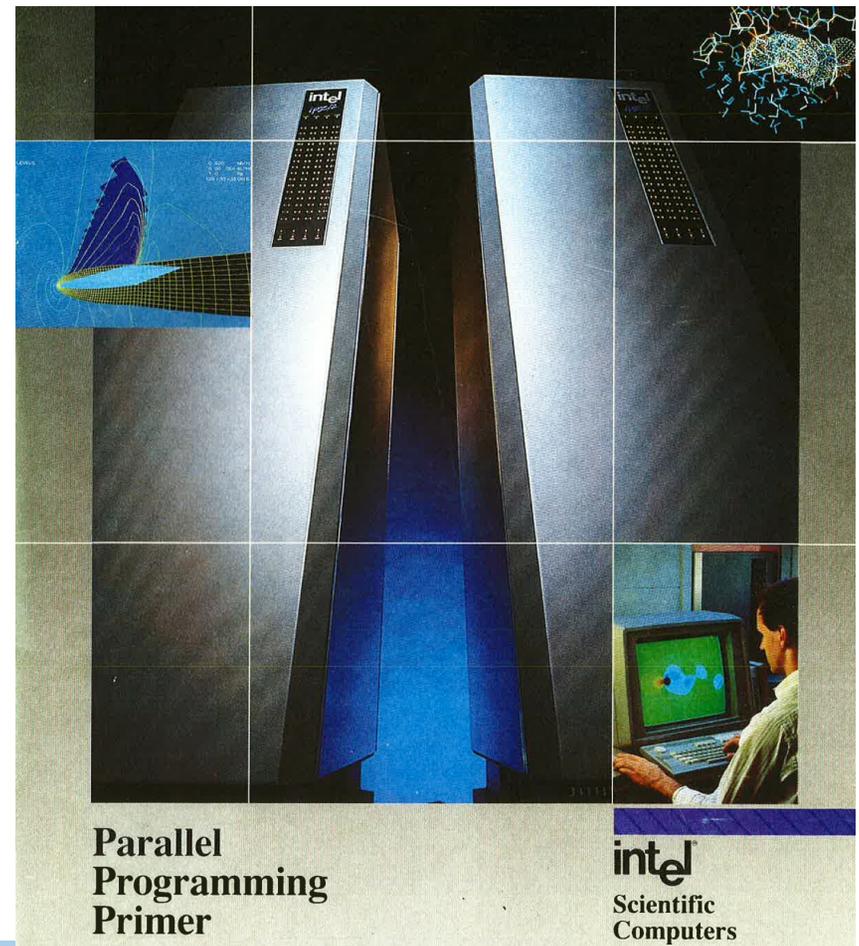


Parallel Computing and Fluid Mechanics: A Quest for Scales

One Perspective...

*Paul Fischer
Mathematics & Computer Science
Argonne National Laboratory*



Fluid Dynamics and Computing: *Scale Complexity*

- Fluid dynamics governs a broad range of physical phenomena governing our daily lives: vascular flow, transportation, energy production and consumption, weather (atmosphere and ocean), astrophysics, ...
- The majority of fluid flow is *turbulent*
 - A broad range of scales of motion with nonlinear interaction.
 - Sensitive to initial conditions (and other forcing) – Lorenz '63
 - *Nonrepeatable* ☹️
- However, in the mean, many flows are repeatable and predictable.
 - Reynolds-Averaged Navier-Stokes (RANS) equations are excellent models for computing the predictable cases – 10^5 x cheaper than LES
- The trick is to know which cases are repeatable and which are not.
 - Unfortunately, there is no theory to say which cases are amenable to RANS approaches, and which are not.



Example of Sensitivity: ANL MAX Experimental Validation Study

- Argonne has constructed a highly instrumented experiment (MAX) to provide detailed velocity and temperature data for code validation.
 - 1 x 1 x 1.68 m³ mock-up of mixing in outlet plenum (SFR or VHTR)
 - PIV for velocity measurements
 - Fast thermal video imaging for temperature measurements

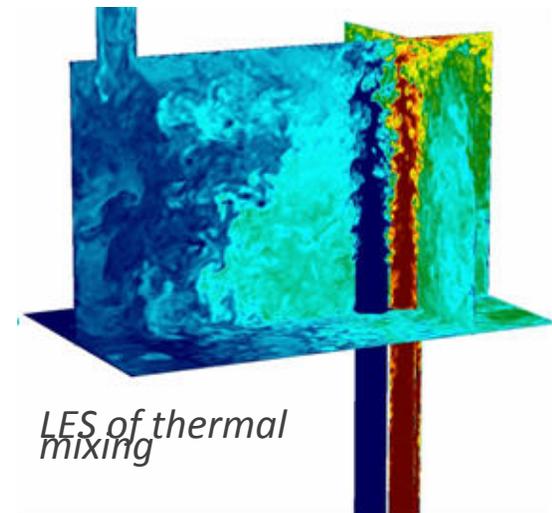
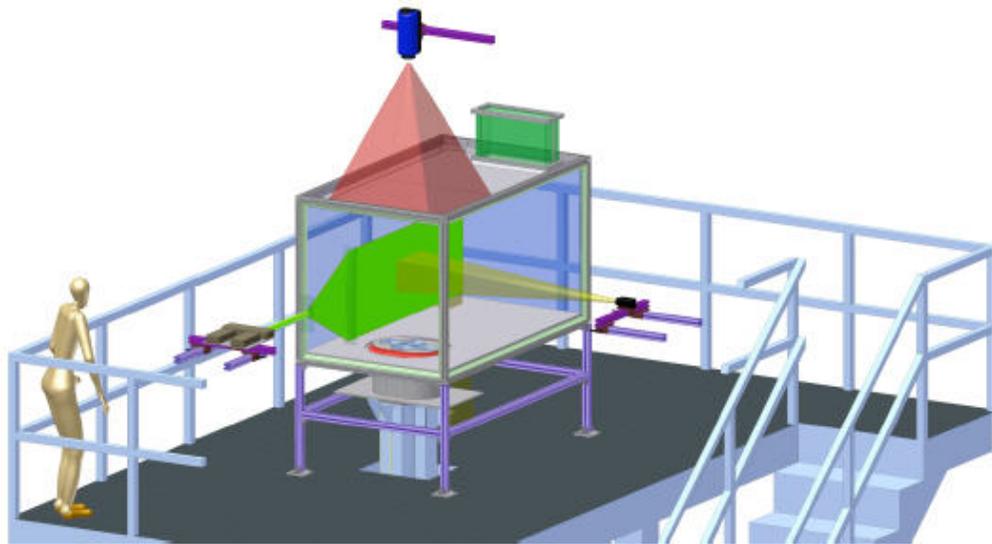
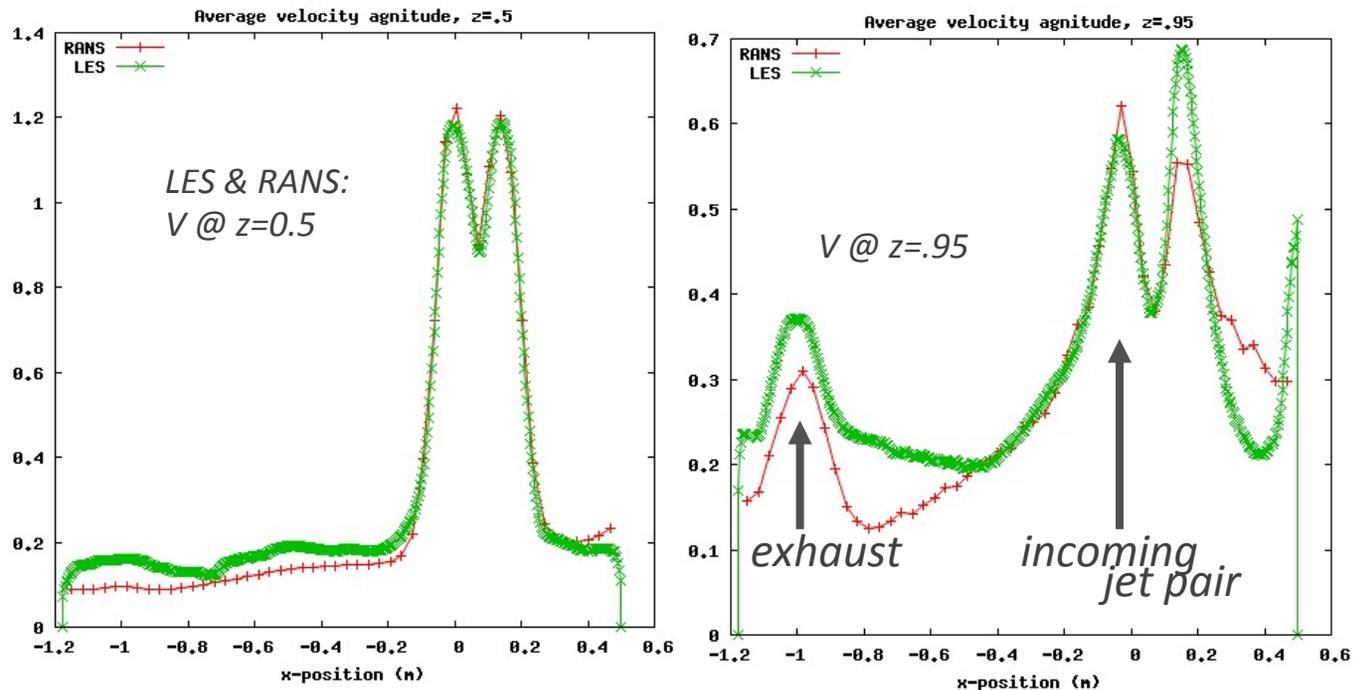


Figure 1. Apparatus for gas mixing experiments: Nd:YLF laser (left), infrared camera (top), PIV camera (right), and hexagonal flow channels (below).

ANL MAX Experiment: LES / RANS Comparisons¹

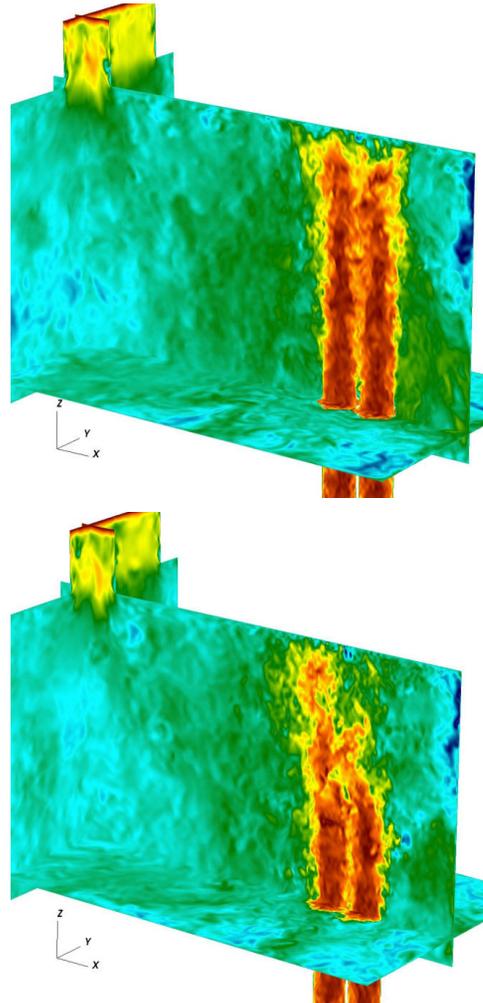
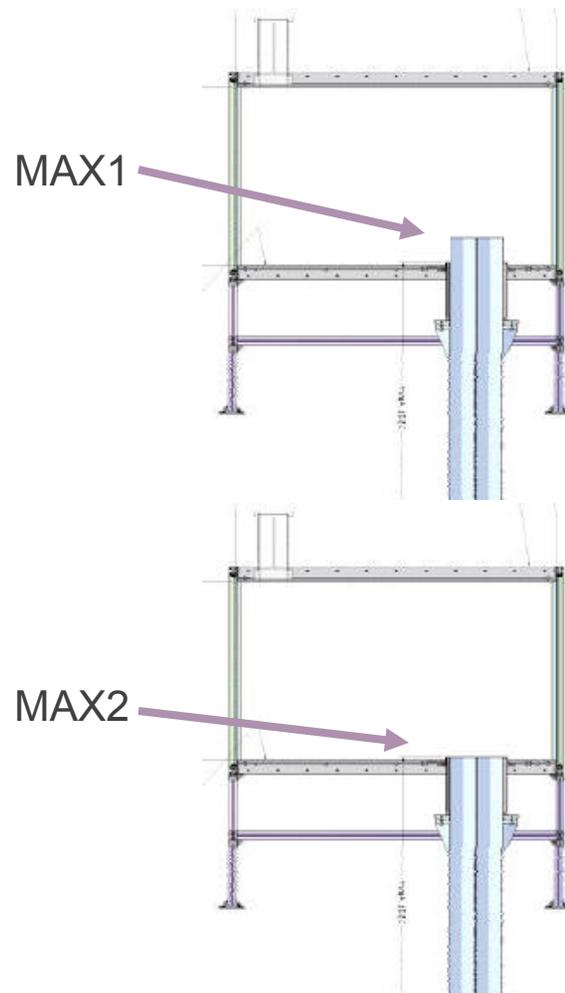
- Time-averaged Nek5000 LES vs. Star-CD RANS velocity profiles at $(x,y,z) = (x,0,z)$, with $z=0.5$ m and $z=0.95$ m
- RANS about 10^5 x cheaper.***



¹ Merzari et al., *On the numerical simulation of thermal striping in the upper plenum of a fast reactor*, ICAPP (2010)



Major Difference in Behavior for Minor Design Change



Simulation Results:

- *Small perturbation yields $O(1)$ change in jet behavior*
- *Unstable jet, with low-frequency (20 – 30 s) oscillations*
- *Visualization shows change due to jet / cross-flow interaction*
- ***MAX2 results NOT predicted by RANS***



Fluid Dynamics and Computing: Scale Complexity (2)

- Fortunately, Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) rely on very little modeling (none, in the case of DNS) and are therefore able to capture many features of turbulent flow.
- These approaches require simulation of a ***broad range of scales*** and their success is largely tied to that of parallel computing.
- Prior to the 80s, the majority of fluid flow simulations were 2D
 - *Definitely not turbulence!*
- A brief taxonomy gives some insight to the fluid dynamics computational landscape

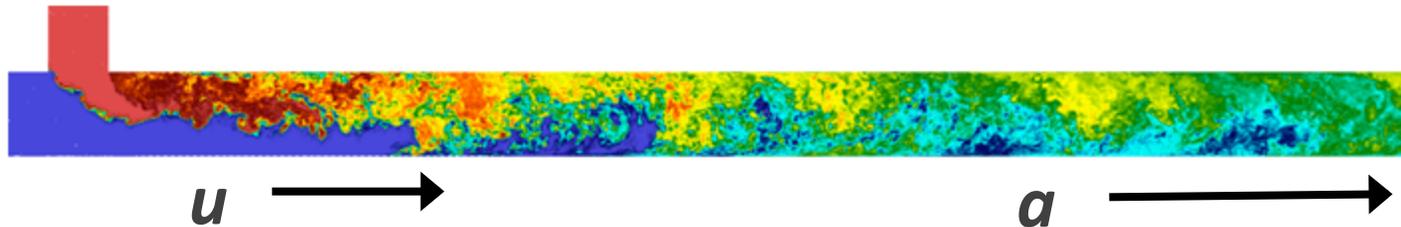


Incompressible Navier-Stokes Equations

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

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

- *Physics: Low Mach-number flow:*



- Interesting speed $u \ll$ sound speed, a (1000x)

- *Multiscale Math:*

- *Replace fast time-scale with an infinitely fast one and use optimal solvers to solve resulting elliptic problem: $\nabla^2 p = f$*

- *Architectural Influence: global coupling*

- *Highly nonlinear & singularly perturbed – a huge range of scales*

Fluid Simulation Taxonomy

- **Explicit** (Jacobi-like) – compressible, high Mach-number: $\|u\| \sim a$
- **Implicit or semi-implicit** low-Mach and incompressible: $\|u\| \ll a$
 - **Direct solvers:**
 - Structured: tensor-product domains / FFT
 - Unstructured: e.g., nested dissection orderings – OK for 2D
 - **Iterative solvers:** unstructured 3D–
 - *Very active research topic from 1980 on...*
 - Projection / acceleration methods: CG, GMRES, etc.
 - Preconditioning: spectra of $A\underline{z} = \lambda M\underline{z}$
 - Polynomial preconditioning, e.g., Chebyshev iteration to reduce communication, etc.



Fluid Simulation Taxonomy

- **Explicit** (Jacobi-like) – compressible, high Mach-number: $\|u\| \sim a$
- **Implicit or semi-implicit** low-Mach and incompressible: $\|u\| \ll a$
 - **Direct solvers:**
 - Structured: tensor-product domains / FFT
 - Unstructured: e.g., nested dissection orderings – OK for 2D
 - **Iterative solvers:** unstructured 3D–
 - *Very active research topic from 1980 on...*
 - Projection / acceleration methods: CG, GMRES, etc.
 - Preconditioning: spectra of $A\underline{z} = \lambda M\underline{z}$
 - Polynomial preconditioning, e.g., Chebyshev iteration to reduce communication, etc.

Spectral FFT-based methods about 10x faster than unstructured counterparts (lower memory, fewer variables to track, etc.)



Scaling Beyond Petaflops - An Applications-Development Perspective

- Three issues —
 - Parallelism — scaling beyond $P = 10^6$
 - Algorithmic scaling —
 - Linear system solves
 - Discretizations
 - Single node performance
- Two model discretizations —
 - Global spectral methods
 - Nearest neighbor methods
(e.g., finite difference/element/volume/spectral element methods)



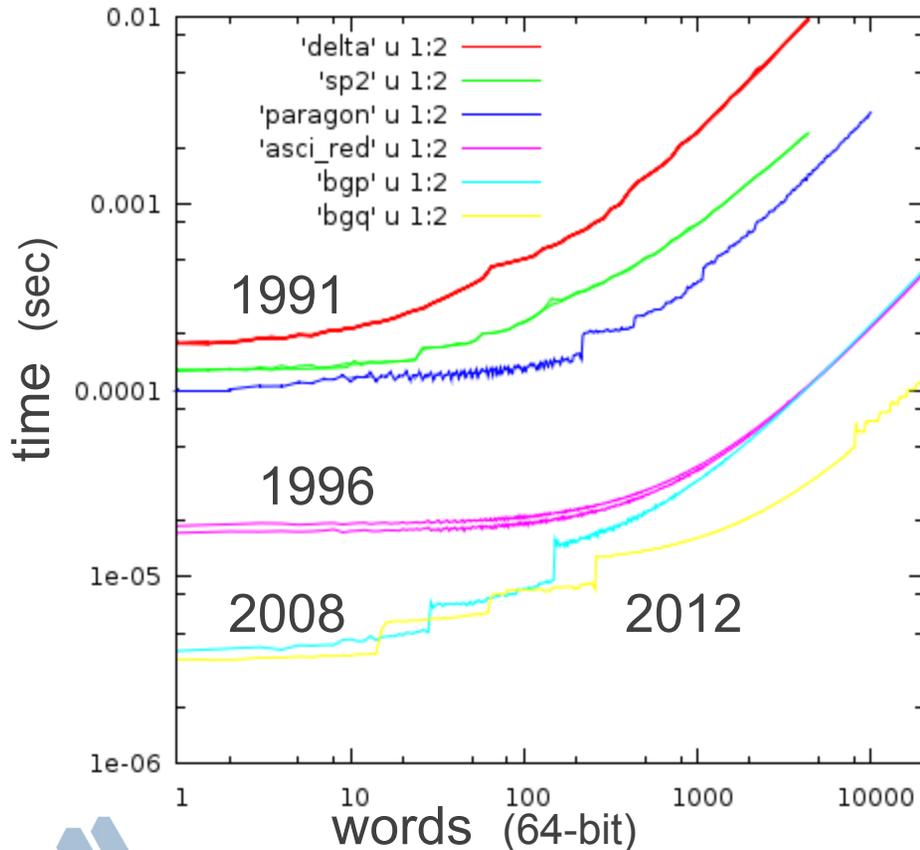
Metric for Scalability

- P-processor solution time for n points:
 - $T(P,n) = T_A(P,n) + T_C(P,n)$, **or** *nonoverlapping comm.*
 - $T(P,n) = \max (T_A(P,n) , T_C(P,n))$ *overlapping comm.*
- As a metric for *scalable*, seek conditions where $T_A > T_C$ *i.e., communication is subdominant*, with
 - $T_A(P,n) = T(1,n) / P =$ parallel work
 - $T_C(P,n) =$ total communication cost



Communication Performance

- Interprocessor communication performance has improved by orders of magnitude over the past decades...



Linear communication model :

$$t_c(m) = \alpha^* + \beta^* m, \quad m: 64\text{-bit words}$$

Nondimensionalize by t_a [$c = a*b$] :

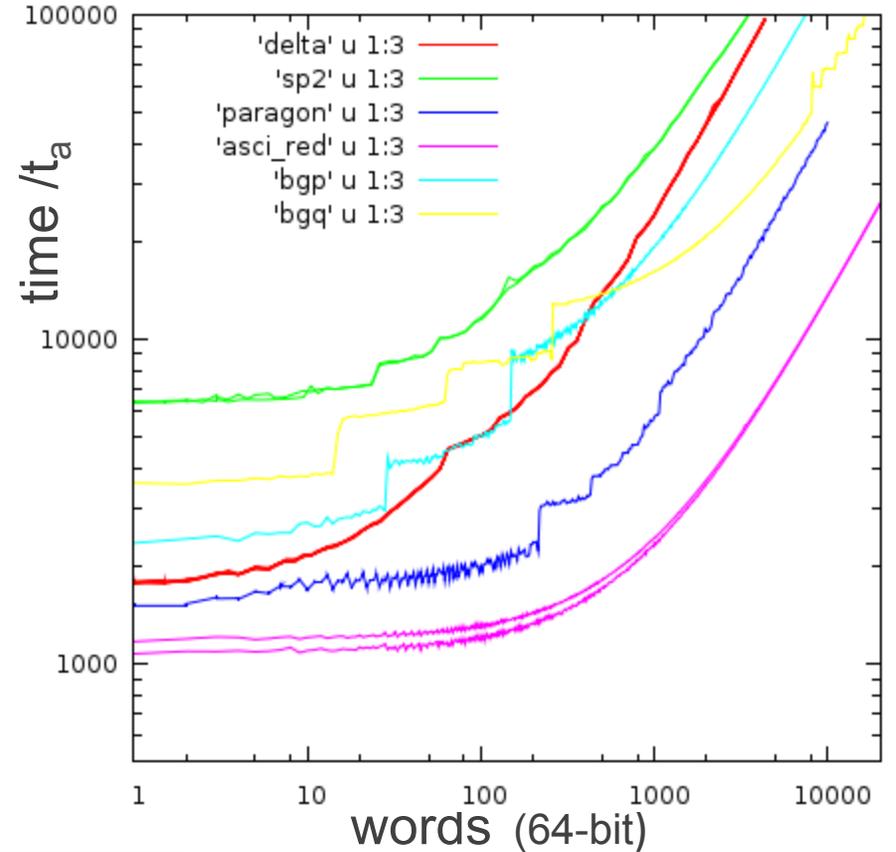
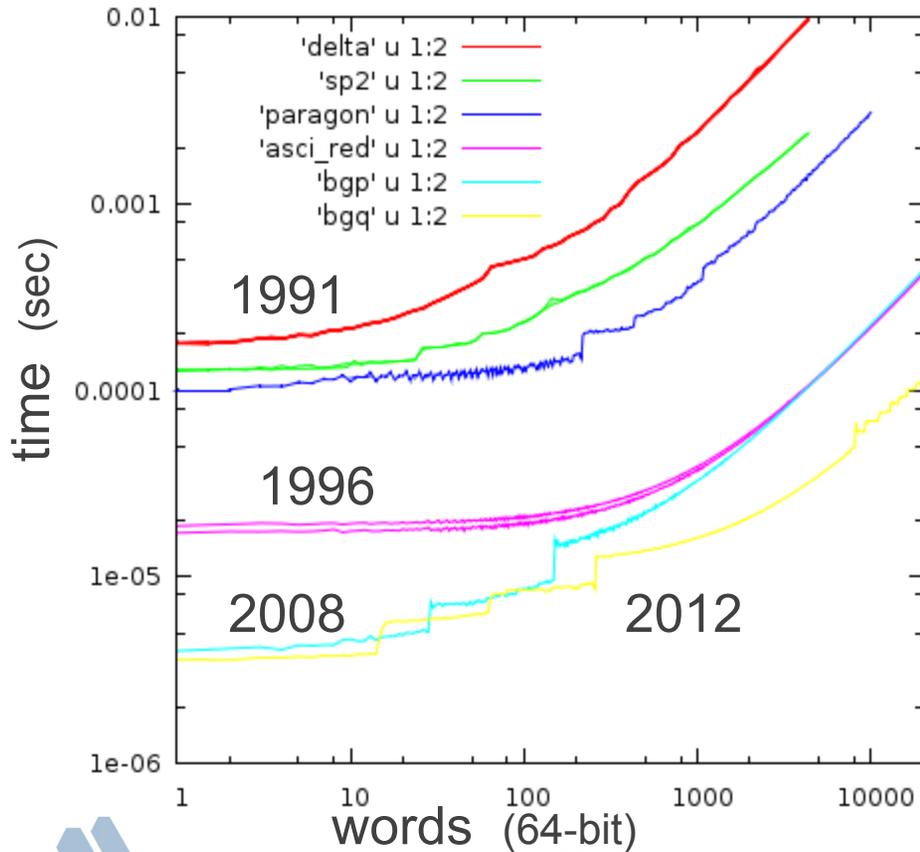
$$t_c(m) = (\alpha + \beta m) t_a$$

$$\alpha = \alpha^* / t_a, \quad \beta = \beta^* / t_a$$



Communication Performance

- Computational rates have also improved... (*reduced t_a*)



26 Years of Nondimensional Machine Parameters

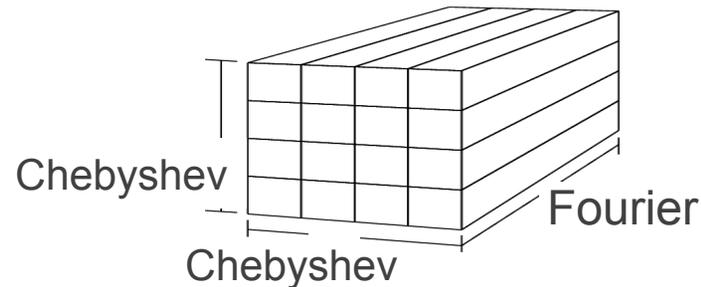
YEAR	t_a (us)	α^*	β^*	α	β	m_2	MACHINE
1986	50.00	5960.	64	119.2	1.3	93	Intel iPSC-1 (286)
1987	.333	5960.	64	18060	192	93	Intel iPSC-1/VX
1988	10.00	938.	2.8	93.8	.28	335	Intel iPSC-2 (386)
1988	.250	938.	2.8	3752	11	335	Intel iPSC-2/VX
1990	.100	80.	2.8	800	28	29	Intel iPSC-i860
1991	.100	60.	.80	600	8	75	Intel Delta
1992	.066	50.	.15	758	2.3	330	Intel Paragon
1995	.020	60.	.27	3000	15	200	IBM SP2 (BU96)
1996	.016	30.	.02	1800	1.25	1500	ASCI Red 333
1998	.006	14.	.06	2300	10	230	SGI Origin 2000
1999	.005	20.	.04	4000	8	375	Cray T3E/450
2005	.002	4.	.026	2000	13	154	BGL/ANL
2008	.0017	4.	.021	2353	12.6	185	BGP/ANL
2011	.0007	2.5	.002	3570	3	1190	Cray Xe6 (KTH) [m2=24]
2012	.0010	4.	.005	5000	5.0	1000	BGQ/ANL

- $m_2 := \alpha / \beta \sim$ message size \rightarrow twice cost of single-word message
- t_a based on matrix-matrix products of order 10—13

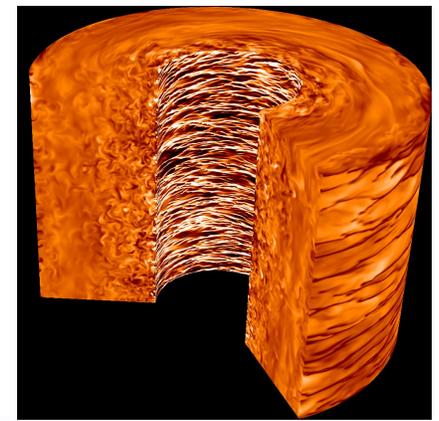
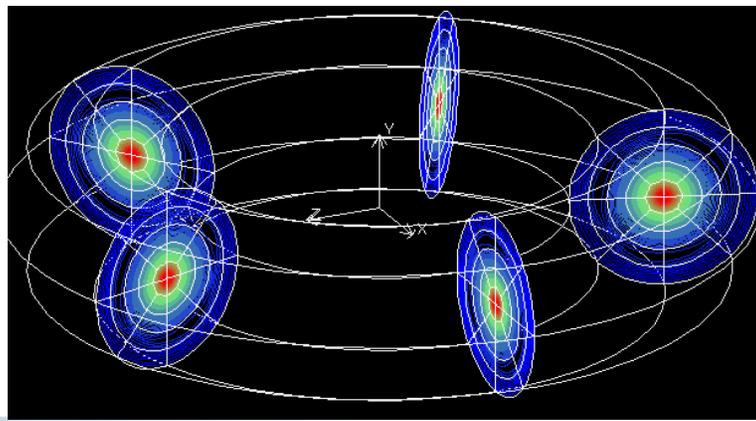
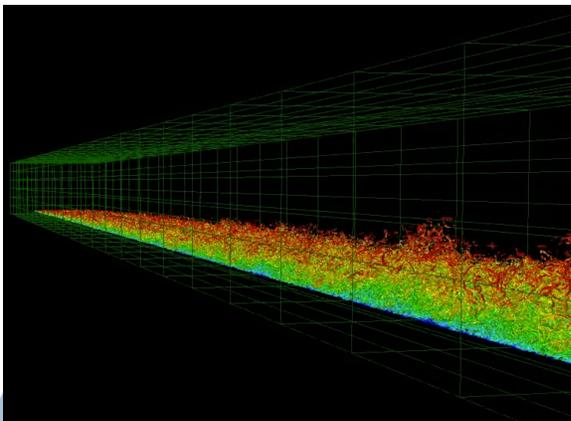


Example 1: Global Spectral Methods

- Global Fourier, Chebyshev, or Legendre expansions in each direction
 - very fast (typically 10x over iterative methods)
 - very accurate (Fourier saves 1.5 in each direction over spectral elements)

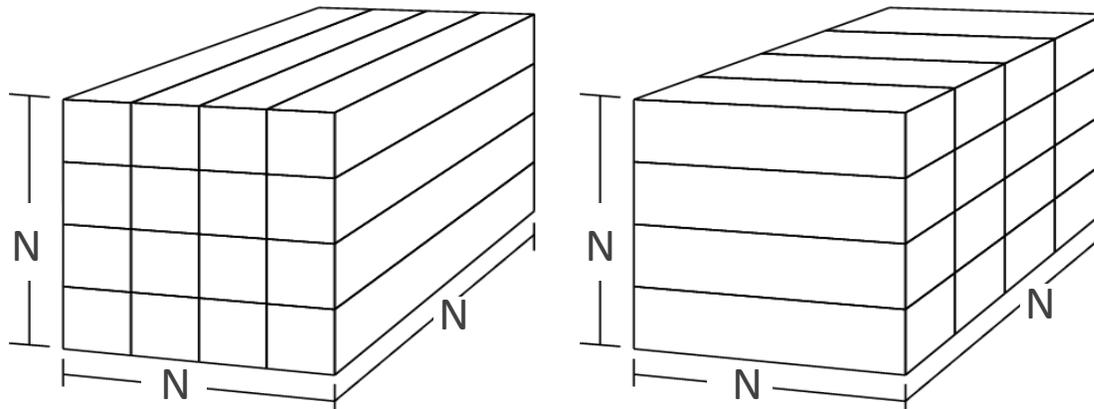


- Relevant to leading-edge *science* applications:
 - turbulence theory, geophysics, astrophysics, fusion (tokamaks, stellarators)



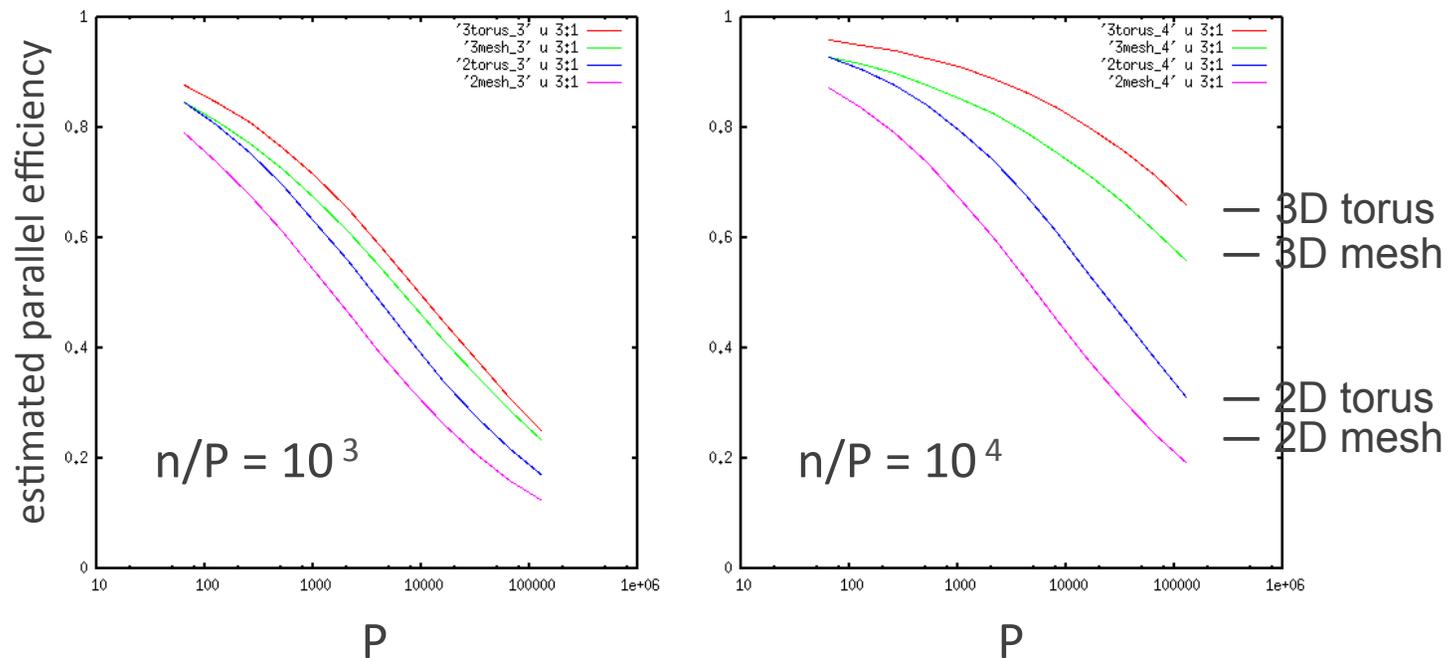
Global Spectral Methods - Computational Complexity

- $n = N^3$ points; n/P = number of points per processor
- Work: $\nabla^2 u = f$: 6 FFTs: $T_a \sim t_a 30 N^3 \log_2 N / P = 10 (n/P) \log_2 n t_a$
- Communication: 4 complete exchanges (all-to-all)
 - $4(P^{1/2} - 1)$ messages of length $N/P^{3/2}$:
 - without contention: $T_c \sim [4 P^{1/2} + 4 (n/P) / m_2] \alpha t_a$
 - with contention on 3D torus: $T_c \sim [4 P^{1/2} + (n/P)/(2m_2) P^{1/3}] \alpha t_a$
 - with contention on 2D mesh: $T_c \sim [4 P^{1/2} + (n/P) / m_2 P^{1/2}] \alpha t_a$
- Define $\eta = T_a / (T_a + T_c)$



Global Spectral Methods - Scalability

- For small n/P , latency dominates – performance is independent of topology
- For $n/P \rightarrow 10^4$, a 3D topology offers significant benefit
- With a 2D topology, an inordinate number of points are required for $\eta = 1/2$
 - However, $\eta = 0.1$ can be realized for relatively small problem sizes



Spectral Examples:

- **First parallel CFD simulation:**
 - Moin & Kim (Stanford/Ames):
Illiac IV (1979-82)
- Turbulent Channel Flow
 - 64 processors; 200 Mflops
 - 22 s/step for 63x64x128

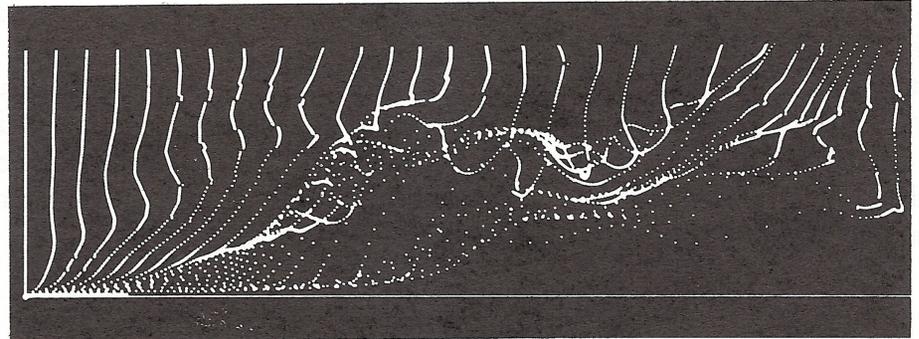
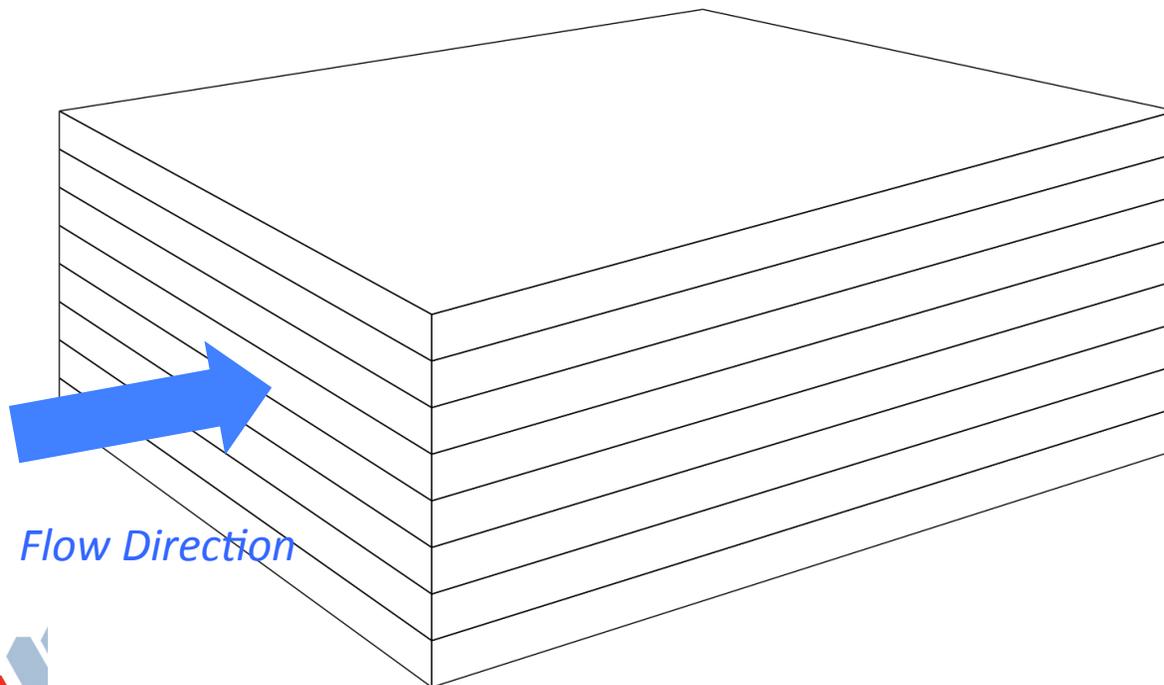


Figure 9 Wall-normal hydrogen-bubble flow visualization from ILLIAC IV large-eddy turbulent channel-flow simulation (from Moin & Kim 1982, with permission); inflectional instantaneous velocity profiles and strong shear layers corroborate experimental observations (Kim et al 1971).



Out-of-core solver.

**Two Active Data Planes
for x-z FFTs;**

**Then, two active y-z
planes loaded for block
tridiagonal solves.**



Spectral Methods on the World's Fastest Computer, May, 1991

- Wray & Rogallo (NASA Ames) – 256^3 Fourier spectral method
 - 2/3 of the time spent in data transpose
 - Hardest part was coding the i860, not parallel communication

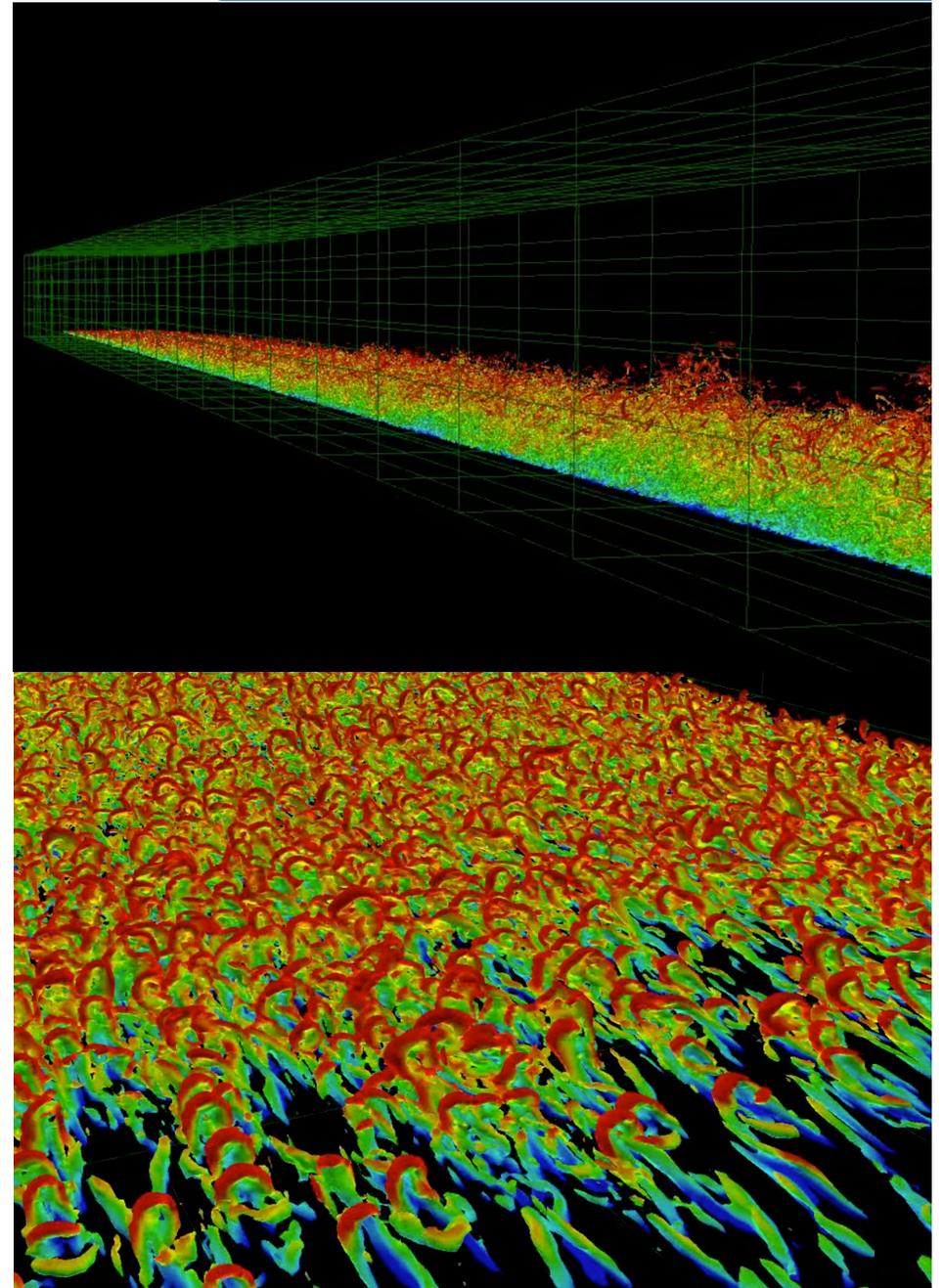
Q: Which of these was the World's fastest computer in May, 1991?



More Recent Spectral Results

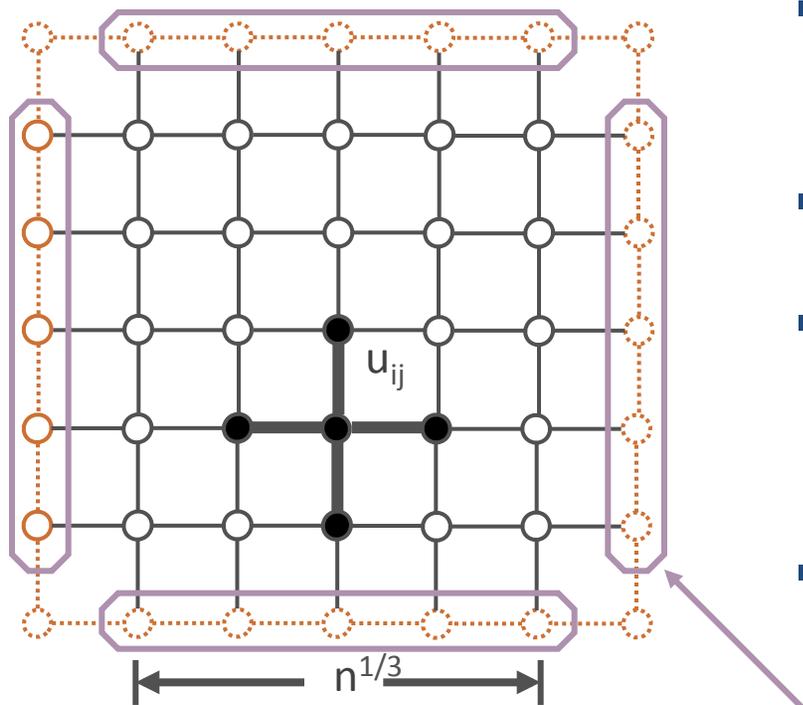
- Schlatter, Chevalier, Ilak & Henningson (2010):
 - Turbulent boundary layer
 - 8192 x 513 x 768

- Current problem sizes in this community are up to $\sim 10,000^3$
 - Getting reasonable parallel efficiency because of sufficient bisection bandwidth



Example 2: Nearest Neighbor Algorithms

- *How far can we scale these algorithms?*
- *What limits scalability?*



processor p

data from neighbor
allows stencil update

- Consider complexity estimates for several iterative solvers.
- n/P points on each processor
- Captures essential complexities of modern iterative solvers for general unstructured discretizations.
- Provides baseline to interpret other physics (e.g., vector fields, combustion)

Complexity Models for Iterative Solvers

– Point Jacobi iteration (7-point stencil):

— Work: $T_{aJ} \sim 14 n/P t_a$

— Communication: $T_{cJ} \sim (6 + (n/P)^{2/3} (1/m_2)) \alpha t_a$

– Conjugate gradient iteration (7-point stencil): (alt: *Chebyshev iteration*)

— Work: $T_{aCG} \sim 27 n/P t_a$

— Communication: $T_{cCG} \sim T_{cJ} + 4 \log_2 P \alpha t_a$

– Geometric Multigrid:

— Work: $T_{aMG} \sim 50 n/P t_a$

— Communication: $T_{cMG} \sim (8 \log_2 n/P + 30/m_2 (n/P)^{2/3} + 8 \log_2 P) \alpha t_a$



Scaling Estimates: Jacobi

- Q: How large must granularity (n/P) be for $T_a \sim T_c$?

$$\frac{T_c}{T_a} = \frac{6 \left(1 + \frac{1}{m_2} (n/P)^{2/3}\right) \alpha}{14 n/P} \leq 1$$

$$\left. \begin{array}{l} \alpha = 2300 \\ \beta = 12.6 \\ m_2 = 185 \end{array} \right\} \text{BG/P parameters}$$

$$(n/P) \approx 2000$$

- ❑ Jacobi scaling is independent of P .
 - ❑ Of course, need occasional all_reduce to check convergence...
 - ❑ Also, not a scalable algorithm.



Scaling Estimates: Conjugate Gradients

$$\frac{T_c}{T_a} = \frac{6 \left(1 + \frac{1}{m_2} (n/P)^{2/3} + 4 \log_2 P \right) \alpha}{27 n/P} \leq 1$$

$$P = 10^6, \quad \log_2 P = 20,$$

$$(n/P) \approx 8500$$

$$P = 10^9, \quad \log_2 P = 30,$$

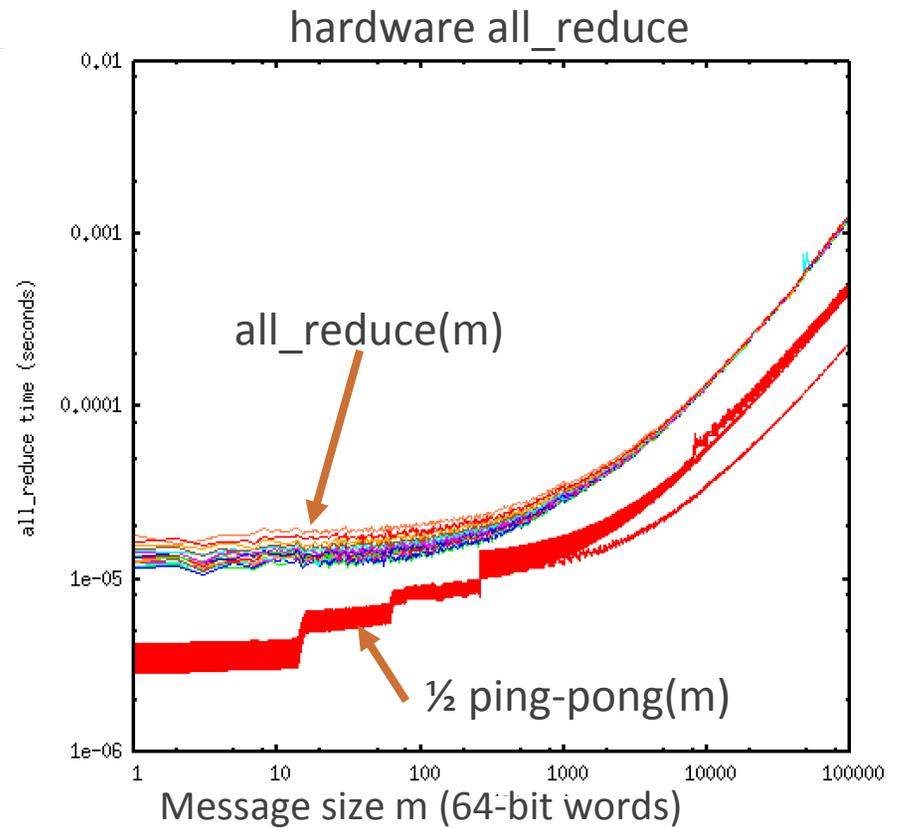
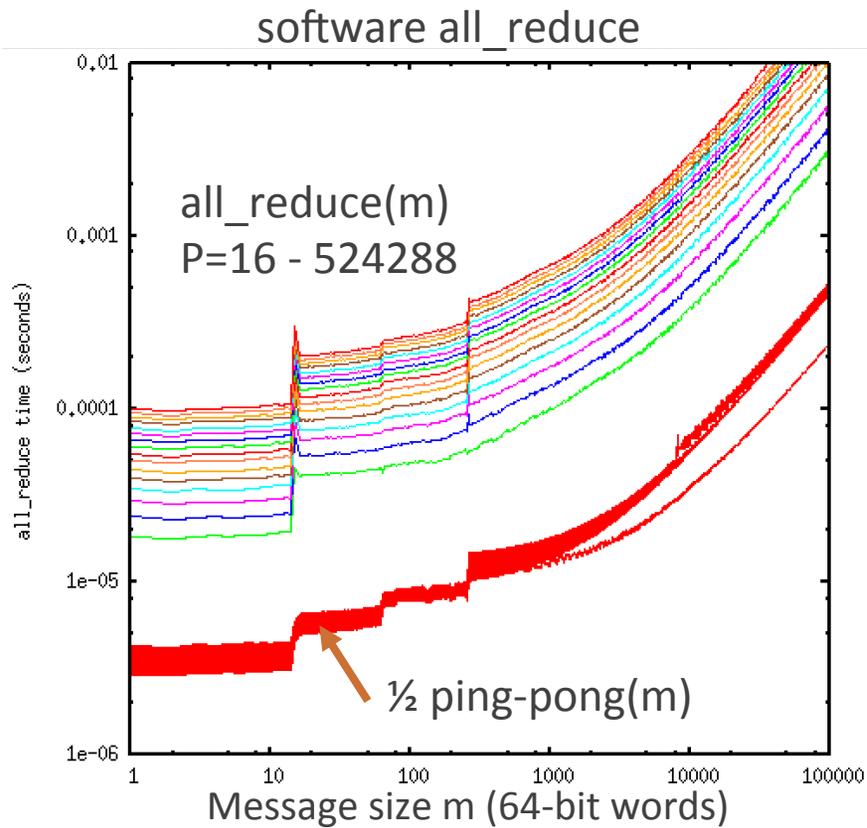
$$(n/P) \approx 12000$$

- ❑ The inner-products in CG, which give it its optimality, drive up the minimal effective granularity because of the $\log P$ scaling of `all_reduce`.
- ❑ On BG/L, /P, /Q, however, `all_reduce` is effectively P -independent.



Eliminating log P term in CG

- On BG/L, /P, /Q, all_reduce is nearly *P-independent*.
- For $P=524288$, all_reduce(1) is only 4α !



Eliminating log P term in CG

$$\frac{T_c}{T_a} = \frac{6 \left(1 + \frac{1}{m_2} (n/P)^{2/3} + 4 \log_2 P \right) \alpha}{27 n/P} \leq 1$$

2 · 4

$$n/P \approx 1200$$

- ❑ On BG/L, /P, /Q, CG is effectively P-independent because of hardware supported all_reduce.
- ❑ In this (admittedly simple) exascale model, net result is a 10x improvement in granularity (n/P=1200 vs. 12,000).



Scaling Estimates: Geometric Multigrid

$$\frac{T_c}{T_a} = \frac{\left(8 \log_2 n/P + \frac{30}{m_2} (n/P)^{2/3} + 8 \log_2 P\right) \alpha}{50 n/P} \leq 1$$

$$n/P (P = 10^3) \approx 13,000$$

$$n/P (P = 10^6) \approx 17,000$$

$$n/P (P = 10^9) \approx 22,000$$

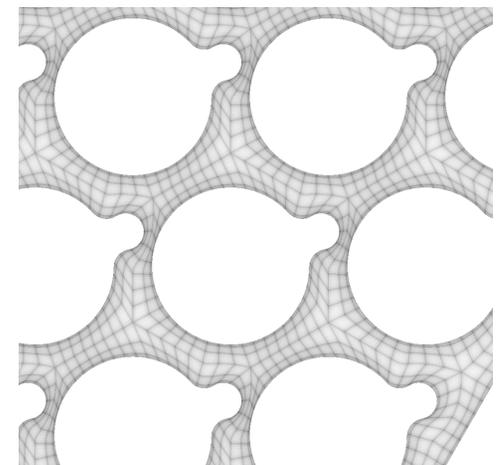
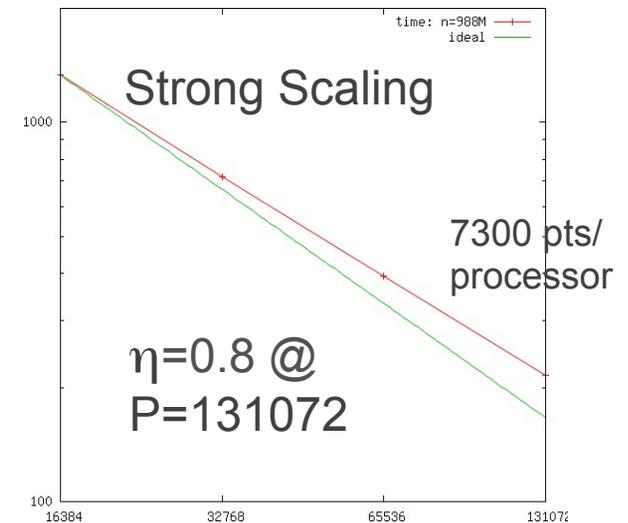
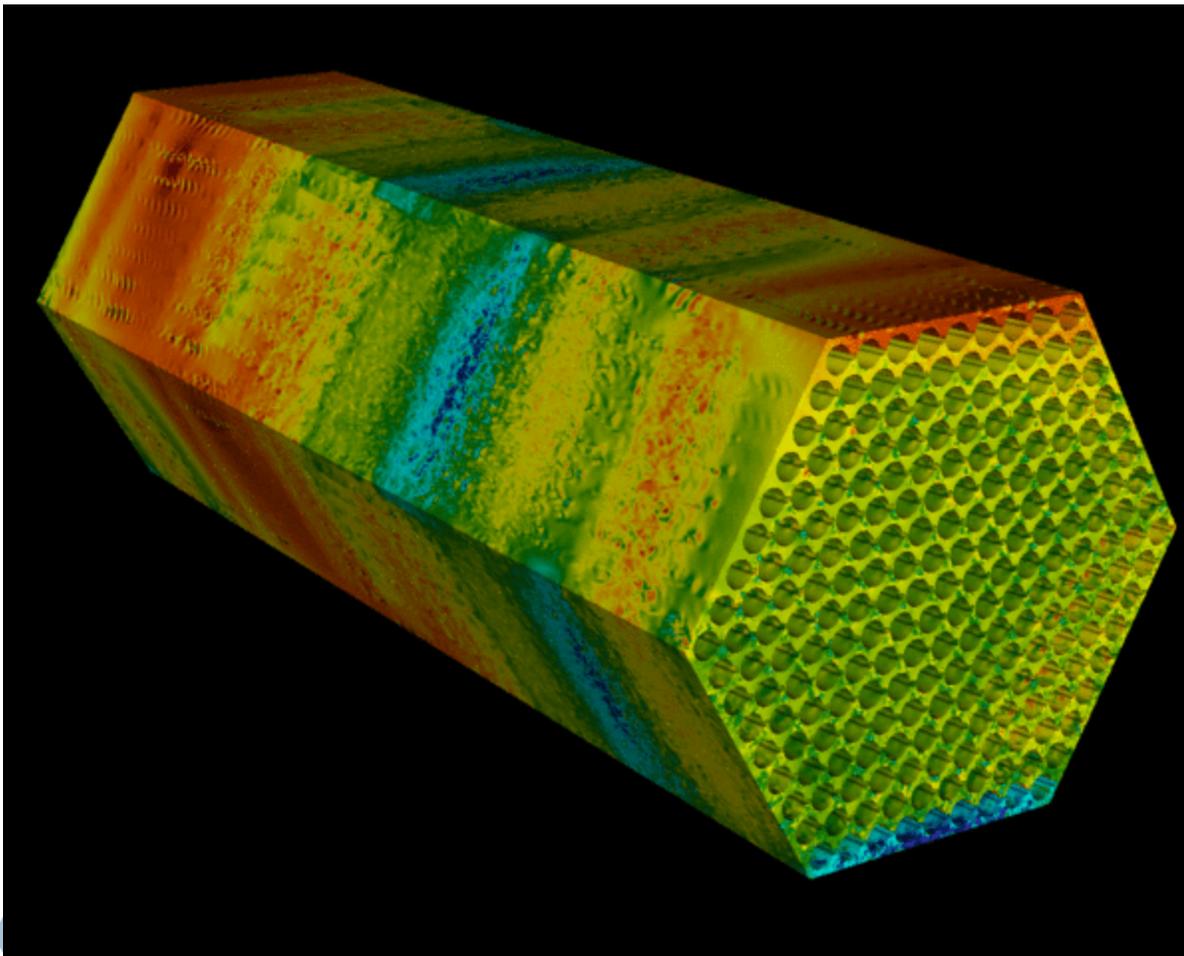
- ❑ In this case, granularity is relatively high because of the $8 \log_2 P$ term, which is associated with the coarse solve in MG.
- ❑ Replacing $8 \alpha \log_2 P$ with 16α yields $n / P \sim 9000$, $> 2x$ gain in scalability.
 - Further savings possible if first term is also reduced.
 - Gains could be realized through hardware support for *parallel prefix operations*
 - Scan / Reduce primitives were central to CM5 programming model are useful for a host of application problems:
TSP/AMG/banded solves/ etc.



Nearest Neighbor Scaling Examples

Subassembly with 217 Wire-Wrapped Pins

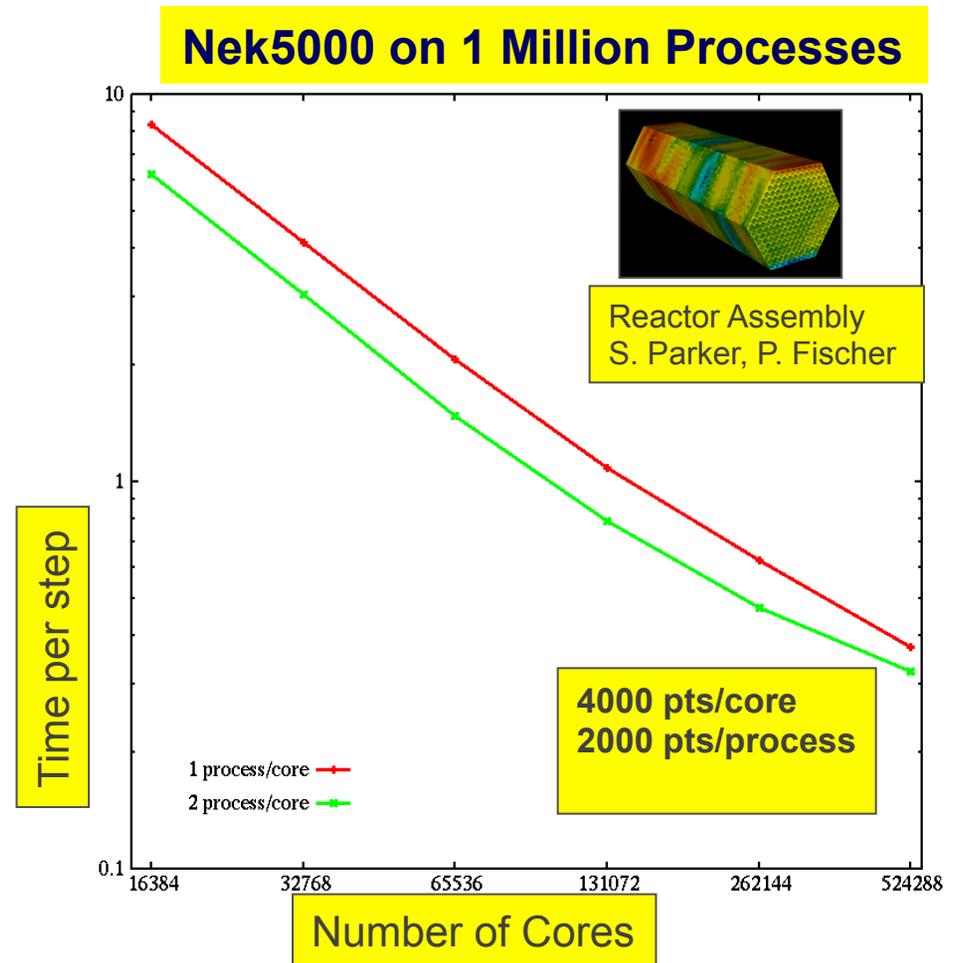
- 3 million 7th-order spectral elements (n=1.01 billion)
- 16384–131072 processors of IBM BG/P
- 15 iterations per timestep; 1 sec/step @ P=131072
- Coarse grid solve < 10% run time at P=131072



Nearest Neighbor Scaling to 1 Million Ranks

217 Pin Problem, $N=9$, $E=3e6$:

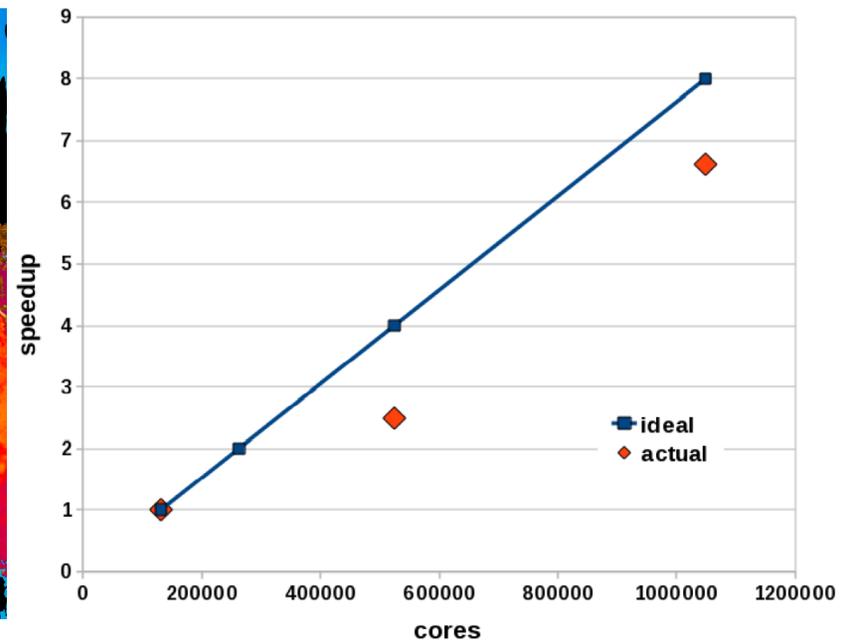
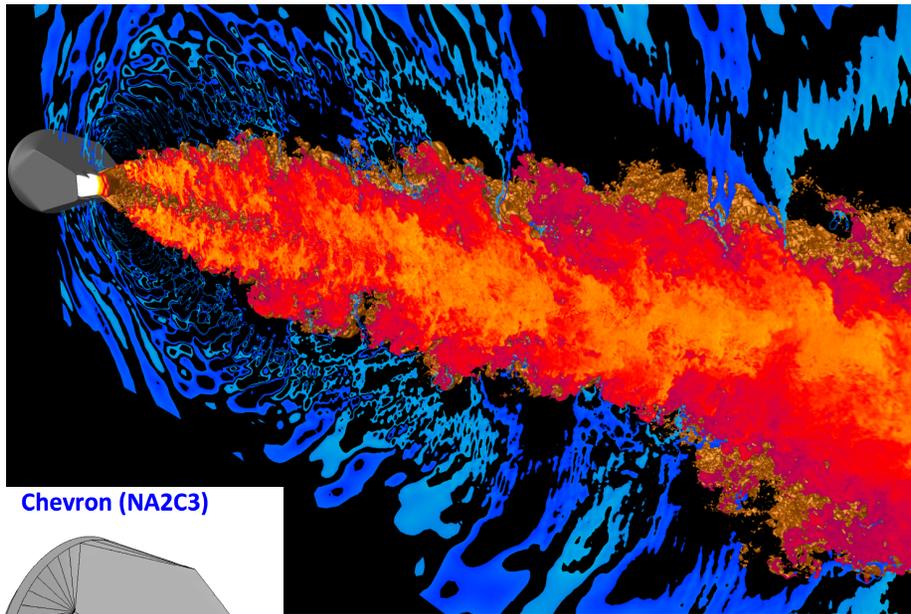
- 60% parallel efficiency @ $P = 10^6$
- 2000 points/process
- 2 billion points
- BGQ – 524288 cores
 - 1 or 2 ranks per core
- A mixture of CG / multigrid



Nearest Neighbor Example - 1 Million Cores

Joe Nichols, Stanford, 2013

- CharLES: explicit, compressible flow solver:
 - turbulence + acoustics; jet noise reduction
- $n/P = 650$
- 83% parallel efficiency @ $P = 1048576$



Parallel Successes

Innovation often arises when simulations become routine, which allows computational scientists to truly experiment.

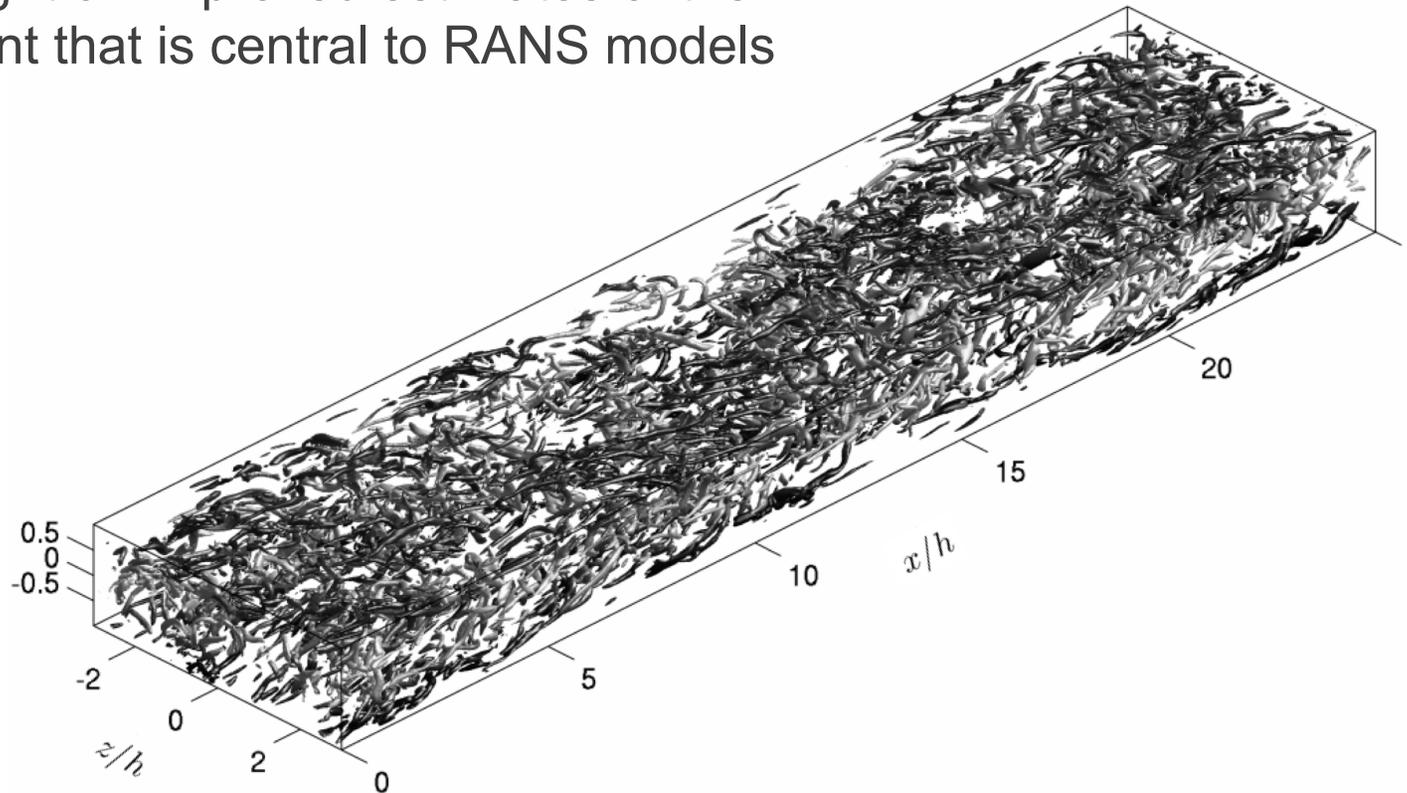
Parallel computing has yielded developments in many areas, including:

- Architectures
- Algorithms
- Physics Modeling
 - Two quick examples



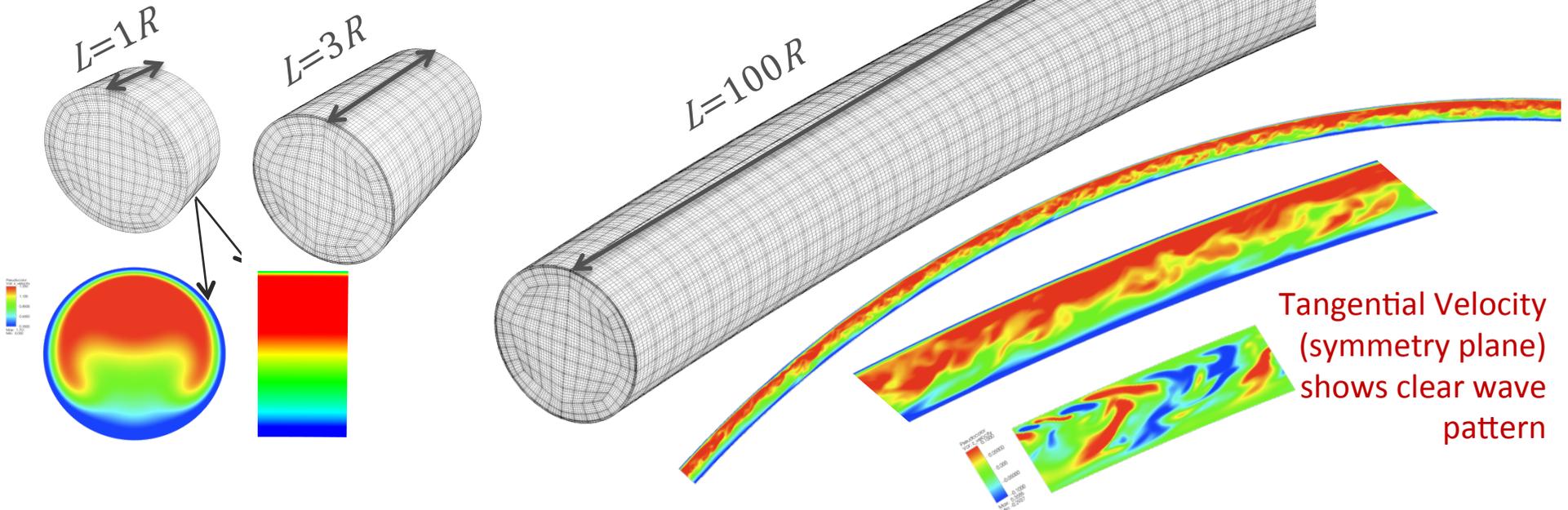
Physical “Channel Flow”

- Vinuesa & Nagib (2013) have found numerically / experimentally that discrepancies between classic (doubly-periodic) channel flow and experiments are attributable to persistent secondary flows that drain mean-flow energy, even for high aspect-ratio ducts.
- Results shed light on improved estimates of the Karman constant that is central to RANS models



Sublaminar Drag in Curved Pipe Flow

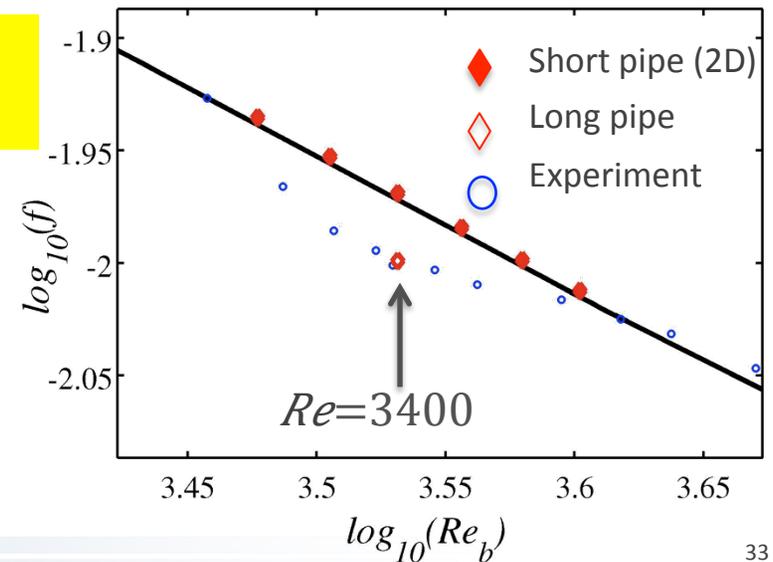
- Noorani & Schlatter '12



Tangential Velocity (symmetry plane) shows clear wave pattern

- DNS results are being used to calibrate new RANS models in commercial engineering codes.

10% drag reduction!

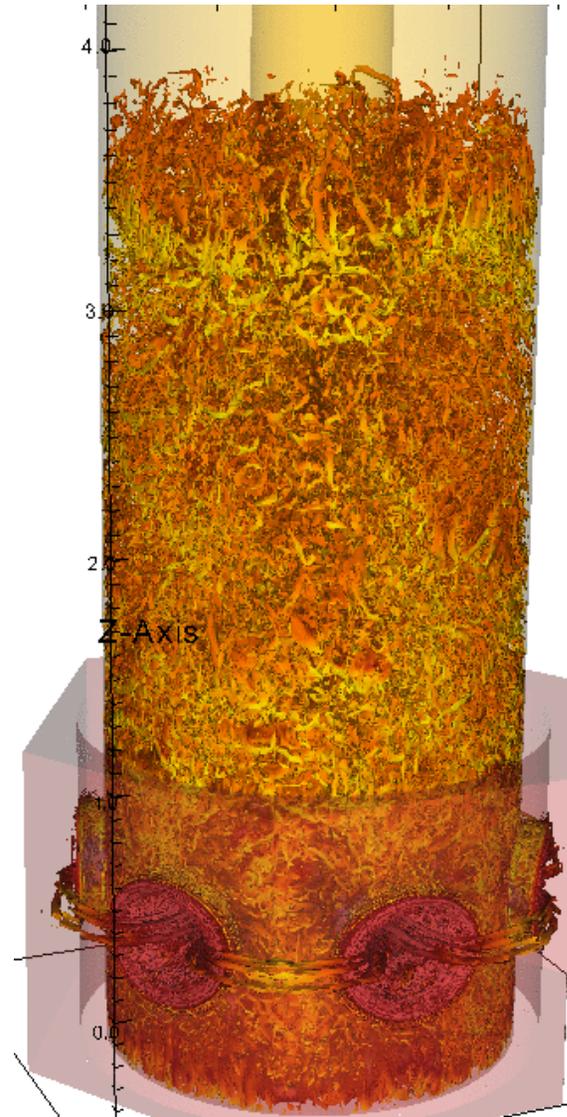


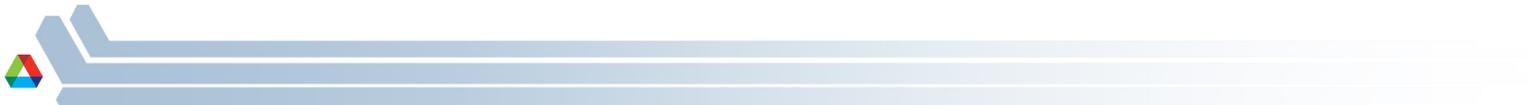
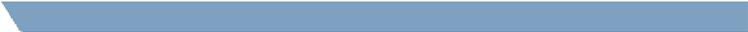
Summary

- Parallel computing is central to the advancement of CFD, in both engineering and research applications.
- Algorithmic / Parallel Scaling Analysis:
 - CG and multigrid scale to $P=10^6$ with current architecture parameters at reasonable granularity in the range of 2,000-20,000 points/core.
 - Estimate that $N \sim 10^{13}$ is the minimum problem size for an exascale fluid simulation.
 - Hardware support for parallel prefix is one possible avenue to finer-grained parallelism
 - Spectral methods require moderately high dimension meshes ($d > 2$)



Thank You!





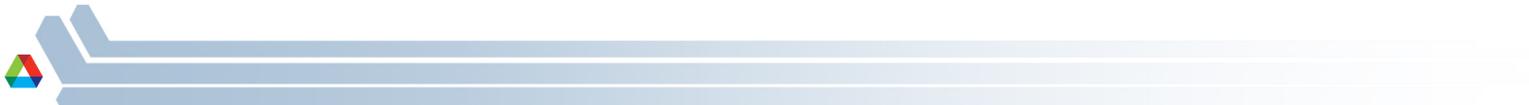
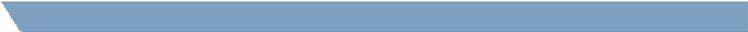
Parallel Successes

Innovation often arises when simulations become routine, which allows computational scientists to truly experiment.

Parallel computing has yielded developments in many areas, including:

- Architectures
 - distributed memory: bandwidth scales as P (!)
 - network topologies; short-cuts, etc.
- Algorithms
 - Private-memory model (resolves memory-processor affinity)
 - Schwarz methods (in particular, RAS)
- Physics
 - Two quick examples





Influence of Scaling on Discretization

Large problem sizes enabled by peta- and exascale computers allow propagation of small features (size λ) over distances $L \gg \lambda$. If speed ~ 1 , then $t_{final} \sim L / \lambda$.

- Dispersion errors accumulate linearly with time:

$$\sim |\text{correct speed} - \text{numerical speed}| * t \quad (\text{for each wavenumber})$$

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

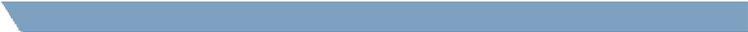
- For fixed final error ε_f , require: numerical dispersion error $\sim (\lambda / L) \varepsilon_f, \ll 1$.

High-order methods can efficiently deliver small dispersion errors.

(Kreiss & Oliger 72, Gottlieb et al. 2007)

- Cost per grid point is comparable (equal) to low-order methods.
- Number of points an order-of-magnitude less.





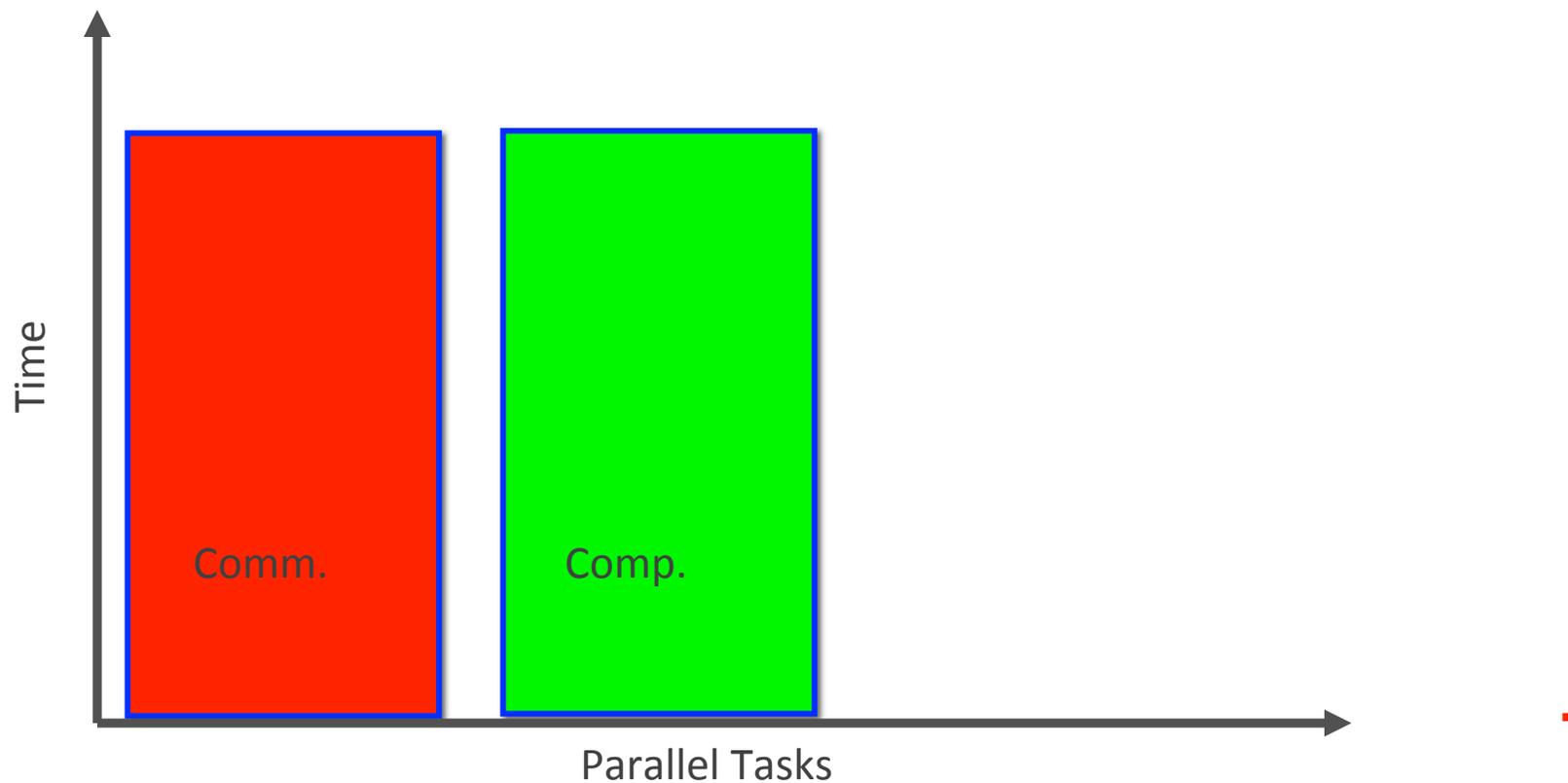
Granularity

- Memory constraints are pushing towards fine-grained parallelism.
- Is it realistic to expect an order of magnitude shift in granularity from current practice?
- Presently, granularity is an elastic parameter that is explored by users every day:
 - User runs on P processors and observes a given run time, T .
 - User runs on $P'=2P$ processors and observes time $T' \sim T/E$, $E < 2$, and decides whether they can tolerate the drop in parallel efficiency.
 - User repeats experiment until E is too small.



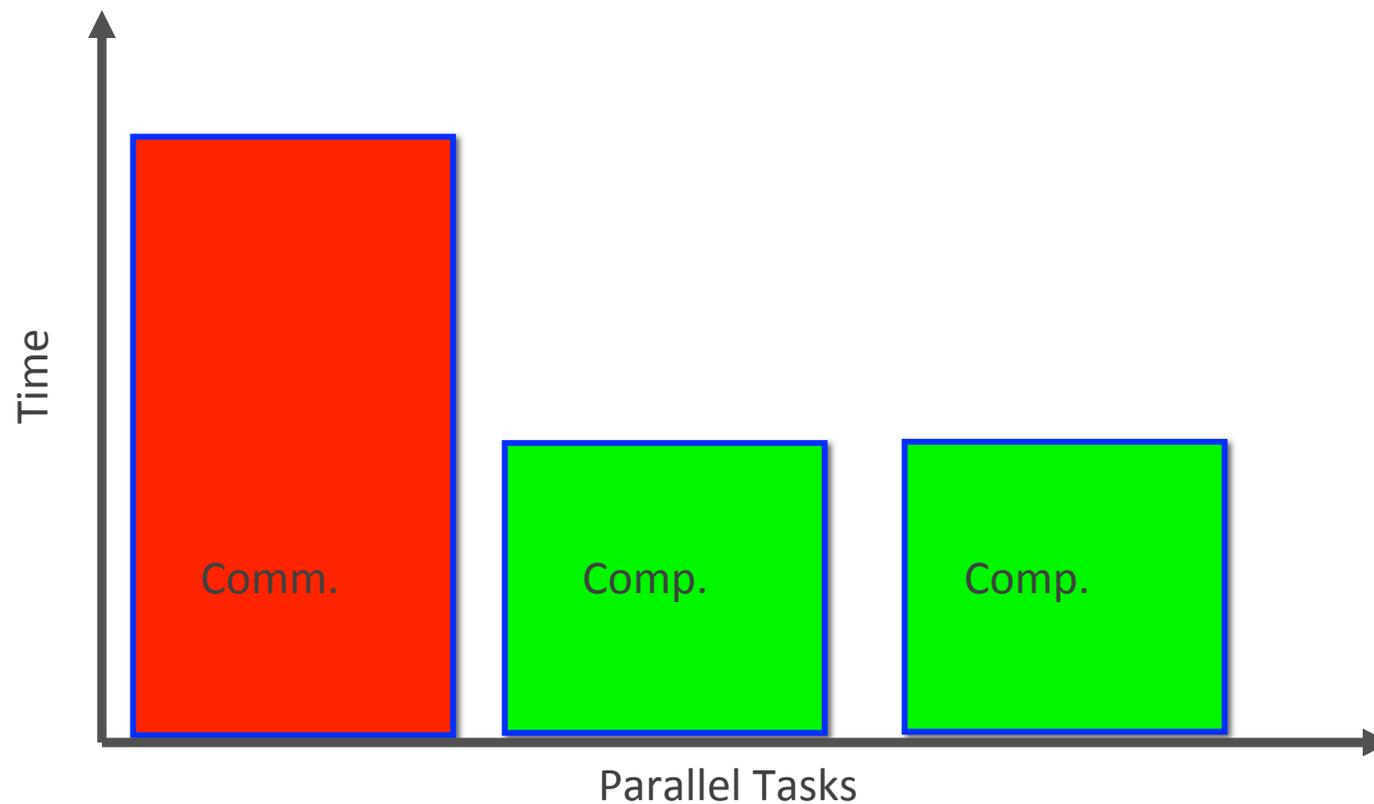
Multicore Does Not Imply Fine-Grained Parallelism

- Consider an operation with balanced communication / computation.
- Assume we can reduce the time spent on work through multicore.



Multicore Does Not Imply Fine-Grained Parallelism

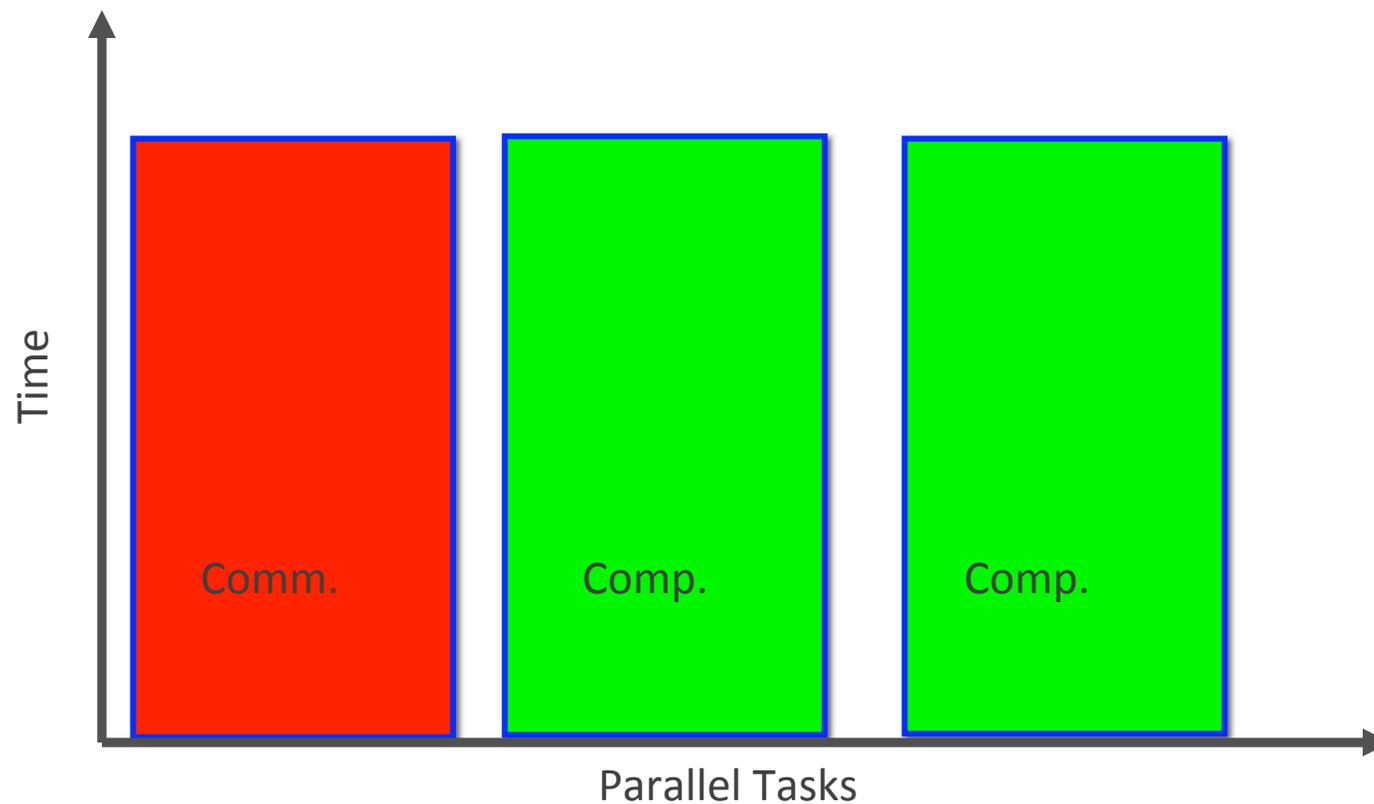
- Internode latency is not reduced by adding more cores.
- Simulation becomes communication dominated.



Multicore Does Not Imply Fine-Grained Parallelism

- To restore computation / communication balance, we must double the work per core.

→ *Net granularity per core is unchanged.*



Granularity

- What ultimately limits granularity?
 - i.e., for fixed problem size, n , how far can we push P ?
 - We have modeled several basic PDE kernels to answer this question.
 - On present-day architectures with reasonably rich (i.e., $> 1D$ mesh), internode latency limits strong scaling to $n/P \sim 10^4$
- If latency could be reduced by 10x, one could strong scale a broad range of applications to $10P$, instead of P .
- Result would be a 10-fold reduction in time to solution
(but no reduction in energy usage).
- *Unfortunately, latency is not readily reduced – but alternative strategies could help.*



