CFD, PDEs, and HPC
A Thirty Year Perspective

Paul Fischer
University of Illinois, CS & MechSE
Argonne National Laboratory, MCS

Katherine Heisey
Stefan Kerkemeier
James Lottes
Oana Marin
Elia Merzari
Misun Min
Aleks Obabko
Philipp Schlatter
Martin Schmitt
Ananias Tomboulides
Active users group (> 250)

Turbulence in a heat-exchanger inlet.
Industrial Example

- 12 hour turnaround for result on the left:
  - 6 hours to mesh, 6 hours to run on 16K cores
- 3 Days for result on the right (mostly meshing...)
Outline

- Turbulence
- PDE discretization and solve strategies
- HPC
Incompressible Navier-Stokes Equations

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

\[\nabla \cdot u = 0\]

Key algorithmic / architectural issues:

- Unsteady evolution implies many timesteps, significant reuse of preconditioners, data partitioning, etc.

- \( \text{div } u = 0 \) implies \textit{long-range global coupling at each timestep} \\
  \rightarrow \text{iterative solvers} \\
  \text{communication intensive}

- Small dissipation \rightarrow \text{large number of scales} \rightarrow \text{large number of gridpoints for high Reynolds number } Re
Navier-Stokes Time Advancement

\[ \frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \frac{1}{Re} \nabla^2 u \]
\[ \nabla \cdot u = 0 \]

- Nonlinear term: explicit
  - $k$ th-order backward difference formula / extrapolation ($k = 2$ or $3$)
  - $k$ th-order characteristics (Pironneau ’82, MPR ‘90)

- Linear Stokes problem: pressure/viscous decoupling:
  - 3 Helmholtz solves for velocity (“easy” w/ Jacobi-precond.CG)
  - Poisson equation for pressure (computationally dominant)
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 – $10^5 \times$ 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 Comparisons

- 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.

---

\(\text{LES & RANS:}\)

\(V \oplus z=0.5\)

\(\text{exhaust}\)

\(\text{incoming jet pair}\)

\(V \oplus z=0.95\)

---

\(^1\) 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
- Visualization shows change due to jet / cross-flow interaction
- MAX2 results \textbf{not} predicted by RANS

Red case \( 10^4 \times \) cheaper than blue case
Sublaminar Drag in Curved Pipe Flow
— Noorani & Schlatter ’12

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

10% drag reduction!

Tangential Velocity (symmetry plane) shows clear wave pattern

$Re = 3400$
**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 turbulent!*

- A brief taxonomy gives some insight to the fluid dynamics computational landscape
Incompressible Navier-Stokes Equations

\[
\frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \frac{1}{Re} \nabla^2 u \\
\nabla \cdot u = 0
\]

- **Physics: Low Mach-number flow:**
  - Interesting speed \( u \ll \text{sound speed}, a \) (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**
Some Turbulence Examples

Optimizing Heat Transfer with Wire-Coil Inserts  J. Collins, ANL

Film Cooling
Duggleby et al., TAMU

Heat Transfer: Exp. and Num.

Pipe Flow:
Re₁ = 550
Re₂ = 1000
G. El Khoury, KTH
DNS Separation in an Asymmetric Diffuser

- Flow separation and recovery
- DNS at Re=10,000: E=127750, N=11, 100 convective time units
- Comparison w/ exptl results of Cherry et al.

Ohlsson, Schlatter, F., and Henningson, JFM (2010)
DNS of Flow around a NACA4412 Wing Profile

- $Re_c = 400,000$ with 5° angle of attack.
- 3.2 billion gridpoints
- Locally structured data
  - (within each high-order element)
- Globally unstructured mesh

Goals: - cycle-to-cycle var.
- thermal analysis

Streamlines: 90°CA

Axial Mean Vel. Axial RMS Vel.

Very good agreement for all available experimental data!
Compression: Significant Increase in Range of Scales at TDC

— M. Schmitt, ETHZ 2014

Impacts thermal boundary layer, initial conditions for ignition.
DNS of Turbulence in the TCC Model

Starting with an .stl file, mesh is made with CUBIT.

The lower panel shows the mesh motion.

Lower-right shows a very fine mesh used for the intake port.
Vortex Breakdown at $Re_D = 15,000$

- These are extremely well resolved calculations performed on Mira.
- Note the highly-resolved filamental horseshoe vortices around the base of the valve stem that ultimately break down into a hairpin vortex chain.
Influence of Reynolds Number

- The Reynolds number has a significant impact on the scales of motion.

- The Reynolds number in the intake port of the TCC engine peaks at around $Re=45000$ at 670 RPM.

- The Reynolds number in the combustion chamber is about $Re=15000$. 

$Re_D=30,000$

$Re_D=45,000$
\( \lambda_2 \) criterion (Jeong & Hussein '95) involves velocity gradients.

\[
\lambda_2 (S^2 + \Omega^2) < 0, \quad S := \frac{1}{2} \left[ \nabla u + (\nabla u)^T \right], \quad \Omega := \frac{1}{2} \left[ \nabla u - (\nabla u)^T \right]
\]

- Places stringent demands on resolution since velocity is only \( C^0 \)
Several meshes, but largest is 3.9 million elements of order N=6:

- 216 x 3.9 million = 840 million points
- 0-180 CAD at 400 RPM
- 100 hours on 131072 cores (~ 1/6 Mira)

Can we run faster?
- Better algorithms?
  - Lower $n$ (via high-order discretizations)
  - Faster solvers (lower iteration counts)
- More parallelism?
Some Relatively Deep Considerations

- High-Order:
  - reduction in problem size and, hopefully, costs.

- Fast multilevel solvers:
  - coarse-grid solve is a big challenge in parallel

- More parallelism
  - How much more?
Influence of Scaling on Discretization

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

- Dispersion errors accumulate linearly with time:

  $\sim |\text{correct speed} - \text{numerical speed}| \times t$  
  \hspace{1cm} \text{(for each wavenumber)}

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

- For fixed final error $\varepsilon_f$, 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)
High-Order Spatial Discretizations

Example: Spectral element method (Patera 84, Maday & Patera 89)

- Variational method, similar to FEM, using GL quadrature.

- Domain partitioned into $E$ high-order hexahedral elements

- Trial and test functions represented as $N$th-order tensor-product polynomials within each element. ($N \sim 4 -- 15$, typ.)

  - $n \sim EN^3$ gridpoints in 3D

  - Fast operator evaluation: $O(n)$ storage, $O(nN)$ work

- Converges exponentially fast with $N$ for smooth solutions.
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)

\[ v_x = 1 - e^{\lambda x} \cos 2\pi y \]
\[ v_y = \frac{\lambda}{2\pi} e^{\lambda x} \sin 2\pi y \]
\[ \lambda := \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2} \]
Excellent transport properties, even for non-smooth solutions

Convection of non-smooth data on a 32x32 grid. 
($K_1 \times K_1$ spectral elements of order $N$). 

(cf. Gottlieb & Orszag 77)
Nonlinear Example: NREL Channel Flow Study
Sprague et al., 2010

- **Accuracy:** Comparison to several metrics in turbulent DNS, $\text{Re}_\tau = 180$ (MKM’99)

![Graph showing accuracy comparison between MKM (1999), SEM, and FV methods]

- $7^{th}$-order SEM needs an order-of-magnitude fewer points than $2^{nd}$-order FV.
Nonlinear Example: NREL Channel Flow Study

Sprague et al., 2010

Benefits in linear problems carry over to nonlinear case.

- **Test case:** DNS $Re_t = 180$ (MKM’ 99)

- **Results:** Properly implemented SEM and FV have the same cost per gridpoint
Some Relatively Deep Considerations

- High-Order:
  - reduction in problem size and, hopefully, costs.

- Fast multilevel solvers:
  - coarse-grid solve is a big challenge in parallel

- More parallelism
  - How much more?
Fast Multilevel Solvers
Multigrid with Additive Schwarz-Based Smoothing

\[ z = M r = \sum_{e=1}^{E} R_e^T A_e^{-1} R_e r + R_0^T A_0^{-1} R_0 r \]

Local Overlapping Smoother: FEM-based Poisson problems with homogeneous Dirichlet boundary conditions, \( A_e \).
Use fast diagonalization.

Coarse Grid Solve: Poisson problem using linear finite elements on entire spectral element mesh, \( A_0 \) (GLOBAL).

(Dryja & Widlund 87, Pahl 93, Lottes & F 05)
Fast Solvers for p-Multigrid

- **Schwarz Smoothers**: fast diagonalization method (Rice et al. 64, Couzy 95, F.02)
  - **Exploit local tensor-product structure**:
    
    \[
    A_e^{-1} = (S \otimes S)(I \otimes A_x + A_y \otimes I)^{-1}(S \otimes S)^T
    \]
  - Complexity < \( A_p \)

- **p-multigrid schedule**:
  
  \[
  N_f = N \\
  N_1 = 3 \\
  N_0 = 1 \text{ (coarse-grid solve)}
  \]

- **Coarse-grid solve**: Direct, \( XX^T \) (F. & Tufo 01) \( \rightarrow \) \( P \sim 100,000 \) or less
  - **Custom AMG**: (Lottes 08/11) \( \rightarrow \) \( P \sim 10^5 \ldots 10^9 \)

  Communication intensive!
Putting At All Together: 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
More Parallelism?
More Parallelism?

- Current simulation has 3.9 million elements.
  - Therefore, max value of P is 3.9 M.
  - \( n/P \sim 216 \) points per processor.
  - Too Low.

Why?

- Back of the envelope analysis suggests a minimum of \( n/P \sim 2000 \) points per MPI rank on Mira to realize 50% parallel efficiency – for any finite difference/volume/element code. (Based on communication cost overhead, rate of local work, etc.)

- Current simulations are at \( n/p \sim 8.4M/262144 \sim 3200 \).
217 Pin Problem, N=9, E=3e6:

- 2 billion points
- BGQ – 524288 cores
  - 1 or 2 ranks per core
- A mixture of CG / multigrid
- 60% parallel efficiency at 1 million ranks
- 2000 points/rank at 1 million ranks
Scaling Questions

- Will this scaling continue?
- Is this the best we can do?

- What, exactly, is better, or even good?
  - Good node performance
  - Strong scaling to large processor counts.

- Strong scaling is ultimately limited by costs that do not go to zero as $n/P \to 0$:

  $$t \sim c_1 \frac{n}{P} + c_2 + c_3 \log P$$

  - $c_2 \sim \text{communication overhead}$
  - $\sim \text{other overhead (memory latency on GPU)}$
  - $\sim \text{Amdahl}$
  - $c_3 \sim \text{can be mitigated by communication hardware}$

- We are interested in understanding what is setting the limits on strong scaling applications, because that sets the limits on speed.
Last Part of Talk

Performance Analysis

- How can we judge if we’re getting any speed up?
  - *Obvious answer is to fix the problem size, and then run on ever larger numbers of processors (or fewer, until you can’t fit into memory).*

- Second part is to ask yourself, why I am not seeing more speed-up for the given problem size, n, and number of processors, P?
Two Run-Time Scenarios

- **Fully Populated Cluster**: job on every node.

- **Supercomputing Center**: job not using all nodes.
  - Why stop at $P$ nodes, instead of $2P$??

- Our question:

- Study model Poisson problem to get insight.
  
  *(PDE + HPC)*
Model Problem: Poisson with finite differences

Consider complexity estimates for 3D Poisson with several iterative solvers.

- $n/P$ points on each processor

Data from neighbor allows stencil update
**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.

- Seek conditions where communication is subdominant, \( T_A > T_C \):
  - \( T_A(P,n) = T(1,n) / P \) the parallel work
  - \( T_C(P,n) \) the total communication cost = sum \( t_c(m) \)
Linear Communication Model

Blue Waters Ping-Pong Test

Linear communication model:

\[ t_c (m) = \alpha^* + \beta^* m, \]

\( m = \) number of 64-bit words

Nondimensionalize by \( t_a \) \([c = a^*b]\) :

\[ t_c (m) = \left( \alpha + \beta \right) m \times t_a \]

\( \alpha = \alpha^* / t_a \), \( \beta = \beta^* / t_a \)
**Linear Communication Model**

- Simple (simplistic?) and general:
  - Determined by two asymptotes \((\alpha, \beta)\)
  - Does not capture contention effects.
    - *Are these important? Probably, on rich nodes.*
  - Our approach: model *ideal* performance and investigate departures from ideal.

- More complex models depend on
  - \(P\)
  - Runtime topology – rarely known (except on Bluegene)
  - Other jobs in system (except on Bluegene)
  - Node architecture
    - *(e.g., Gropp, Olson, Samfass 2016: Modeling MPI Communication Performance on SMP Nodes: Is it Time to Retire the Ping Pong Test?)*
Modeling MPI Communication Performance on SMP Nodes: Is it Time to Retire the Ping Pong Test?

William Gropp
Department of Computer Science
University of Illinois at Urbana-Champaign
wgropp@illinois.edu

Luke N. Olson
Department of Computer Science
University of Illinois at Urbana-Champaign
lukeo@illinois.edu

Philipp Samfass
Department of Computer Science
University of Illinois at Urbana-Champaign
samfass2@illinois.edu

ABSTRACT

The “postal” model of communication \[ T = \alpha + \beta n \]
for sending \( n \) bytes of data between two processes with latency \( \alpha \) and bandwidth \( 1/\beta \), is perhaps the most commonly used communication performance model in parallel computing. This performance model is often used in developing and evaluating parallel algorithms in high-performance computing, and was an effective model when it was first proposed. Consequently, numerous tests of “ping pong” communication have been developed in order to measure these parameters in the model. However, with the advent of multicore nodes connected to a single (or a few) network interfaces, the model has become a poor match to modern hardware.

understand performance issues in applications. One of the earliest, sometimes called the postal model \([3, 8]^2\), represents the communication time to send \( n \) bytes as

\[ T = \alpha + \beta n, \] \hspace{1cm} (1)

where the term \( \alpha \) denotes the total latency and \( \beta \) the inverse of the asymptotic bandwidth (for arbitrarily large \( n \), measured in seconds per byte). While numerous other performance models have been proposed [16] and have found some use, the postal model remains the most common performance model used to analyze message-passing programs.

Consequently, many benchmarks measure \( \alpha \) and \( \beta \) by sending a message to another process, followed by sending the
Scaling Limits for PDE-Based Simulation

Paul F. Fischer*† Katherine Heisey† Misun Min‡

We analyze algorithm/architecture performance characteristics that have a direct impact on the scalability of present-day and future turbulent flow simulations on large-scale parallel computers.

I. Introduction

Parallel computing is founded on the principle that, given enough work for a given problem, one can subdivide the computation across $P$ processors and realize an effective $P$-fold reduction in time to solution. On today’s architectures, any PDE-based or particle-based simulation that uses a billion gridpoints or particles can easily be distributed across two compute nodes and run in half the time—for essentially the same power—compared with running on just a single node. This computational scenario, running a problem of fixed size in half the time on two processors or nearly one-$P$th the time on $P$ processors, is termed strong scaling and is the focus of this paper. Specifically, we explore the basic question of how far one can scale a given problem, defined by its computational resolution $n$ (e.g., the number of gridpoints), when using $P$ processors.

The relevance of the strong scaling question is expressed succinctly in the equation

$$S_P = \eta P S_1,$$

(1)
Linear Communication Model

Simple (simplistic?) and general:
- Determined by two asymptotes \((\alpha, \beta)\)
- Does not capture contention effects.
  - *Are these important? Probably, on rich nodes.*
- Our approach: model *ideal* performance and investigate departures from ideal.

More complex models depend on
- \(P\)
- Runtime topology – rarely known (except on Bluegene)
- Other jobs in system (except on Bluegene)
- Node architecture
  - *(e.g., Gropp, Olson, Samfass 2016: Modeling MPI Communication Performance on SMP Nodes: Is it Time to Retire the Ping Pong Test?)*
Switch to 3-trip messaging:
- Source: “n bytes coming, ready?”
- Target: “ready”
- Source: n bytes → target
Linear Communication Model – \( P \) dependence

- The switch point is important in the strong-scale limit \( n/P \to 0 \)
  - Communication costs are significant (surface/volume large)
  - Messages are short:
    - \( m < m_{\text{switch}} \): \( \text{Cost} \sim \alpha \)
    - \( m > m_{\text{switch}} \): \( \text{Cost} \sim 3 \alpha \)

- Switch point is reduced when \( P \) increases
  - Need to reserve \( P \) buffers of size \( b \): local memory demand scales as \( (P \times b) \)
  - Inveighs against strong scaling.

- For PDEs, we don’t need a lot of large buffers.
  - Rare is the 3D application where ranks talk actively to more than 100 other ranks.
  - \textbf{We should boost buffer sizes on most active rank-to-rank exchanges.}
  - \textbf{Reduce (eliminate) unused set-aside buffers.}
  - Maybe already in place? (A question to MPI experts in the audience..)
Switch to 3-trip messaging:
- Source: “n bytes coming, ready?”
- Target: “ready”
- Source: n bytes → target

Larger P → smaller buffers
- “Short messages” are shorter
Linear Communication Model – $P$ dependence

- Going from $P=1024$ to $P=32768$ on BW
  - Switchpoint between short and long messages moves from 250 words to 25 words.

- If you have $10^3 = 1000$ points/rank, need switchpoint to be $\sim 100 = 10^2$. 
Another argument for (or against) the simple linear communication model…

System noise is difficult to quantify
Linear Communication Model over Several Decades
Nondimensionalized!

Linear communication model:
\[ t_c(m) = \alpha^* + \beta^* m, \quad m: \text{64-bit words} \]

Nondimensionalize by \( t_a \) \([c = a*b]: \)
\[ t_c(m) = (\alpha + \beta m) \cdot t_a \]

\[ \alpha = \alpha^* / t_a, \quad \beta = \beta^* / t_a \]
## 30 Years of Nondimensional Machine Parameters

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

- $m_2 := \frac{\alpha}{\beta} \sim \text{message size} \rightarrow \text{twice cost of single-word message}$
- $t_a$ based on noncached matrix-matrix products of order 10—13
Scalability Estimates: Jacobi Iteration

Point Jacobi iteration (7-point stencil): \[ u_i = \frac{1}{a_{ii}} \left( f_i - \sum_{j \neq i} a_{ij} u_j \right) \]

— Work: \[ T_{aJ} \sim 14 \frac{n}{P} t_a \]
— Communication: \[ T_{cJ} \sim \left( 6 + \frac{n}{P} \right)^{2/3} \frac{1}{m_2} \alpha t_a \]

— For fixed \( \frac{n}{P} \), Jacobi complexity is \( P \)-independent \textbf{assuming \( P \) independent message costs; no contention.}

— However, algorithmic scaling is poor (iteration count scales as \( n^{2/3} \))
  – a more communication intensive approach is required
  – conjugate gradient iteration, multigrid, etc.

  – \textit{Jacobi is nonetheless a reasonable surrogate for explicit timesteppers}
Complexity Models for Iterative Solvers

- Point Jacobi iteration (7-point stencil, 3D):
  - Work: \( T_{aJ} \sim 14 \frac{n}{P} t_a \)
  - Communication: \( T_{cJ} \sim (6 + (\frac{n}{P})^{2/3} \frac{1}{m_2}) \alpha t_a \)

- Conjugate gradient iteration (7-point stencil): (alt: Chebyshev iteration)
  - Work: \( T_{aCG} \sim 27 \frac{n}{P} t_a \)
  - Communication: \( T_{cCG} \sim T_{cJ} + 4 \log_2 P \alpha t_a \)

- Geometric Multigrid:
  - Work: \( T_{aMG} \sim 50 \frac{n}{P} t_a \)
  - Communication: \( T_{cMG} \sim (8 \log_2 \frac{n}{P} + 30/m_2 (\frac{n}{P})^{2/3} + 8 \log_2 P) \alpha t_a \)
Scaling Estimates: Jacobi

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

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

\[\begin{align*}
\alpha &= 2300 \\
\beta &= 12.6 \\
m_2 &= 185
\end{align*}\]

\[\frac{n/P}{\approx 2000}\]

Jacobi scaling is independent of \( P \).

- Of course, need occasional all_reduce to check convergence…
- Also, not a scalable algorithm (but, similar to explicit timestepper)
Scaling Estimates: Conjugate Gradients

\[ z = D^{-1} r \]
\[ r = r^t z \]
\[ p = z + \beta p \]
\[ w = A p \]
\[ \sigma = w^t p \]
\[ x = x + \alpha p \]
\[ r = r - \alpha p \]

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.
Scaling Estimates: Conjugate Gradients

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

\[P = 10^6, \quad \log_2 P = 20, \quad (n/P) \approx 8500\]

\[P = 10^9, \quad \log_2 P = 30, \quad (n/P) \approx 12000\]

- 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 $4\alpha$!
Eliminating log $P$ term in CG

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

$n/P \approx 1200$

- 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).

  $\rightarrow$ 10x faster run, but no reduction in power consumption.
Nek/BGP Communication Cost Distribution vs Rank

- Billion-point 217-pin bundle simulation on P=65536

![Graph showing communication costs]

- Coarse solve time
- Neighbor exchange
- mpi_all_reduce

- Neighbor vs. all_reduce:  $50\alpha$ vs $4\alpha$  ($4\alpha$, not $16 \times 4\alpha$)
Scaling Estimates: Multigrid

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

\[
\frac{n}{P} (P = 10^3) \approx 13,000
\]

\[
\frac{n}{P} (P = 10^6) \approx 17,000
\]

\[
\frac{n}{P} (P = 10^9) \approx 22,000
\]

- Replacing \(8 \alpha \log_2 P\) with \(16 \alpha\) yields \(\frac{n}{P} \sim 9000\), which is > 2x gain in scalability.

**Such gains could be realized through hardware support in the network interface card (NIC) for scan / reduce operation**

- Some vendors have indicated a move in this direction. (yay!)

- Further savings might be possible by reducing the first term. (Algorithmic issues addressed by Bell, Dalton, Olson, 2013.) This is an excellent co-design opportunity.
Measured and Modeled Multigrid Performance

Measured BG/P Multigrid Performance


t_{A} = t_{C}

Modeled BG/P Multigrid Performance

1.1: Left: Measured scalability for 3D geometric multigrid, \(\frac{t_{A} + t_{C}}{t_{A}}\) as a function of varying processor counts, \(P\). Right: Modeled scalability for 3D geometric multigrid using 1.1.
Returning to Original Scaling Question

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

- 2 billion points
- BGQ – 524288 cores
  - 1 or 2 ranks per core
- A mixture of CG / multigrid
- 60% parallel efficiency at 1 million processes
- 2000 points/process
What about GPUs or more Complex Nodes?

- A major concern here is the “\( n_{1/2} \)"
  - \( n_{1/2} := \text{the value of } n \text{ (or, } n/P) \text{ where the node performance reaches } \frac{1}{2} \text{ its peak for your application.} \)

- In addition to communication overhead, \( n_{1/2} \) sets the strong scale limit on complex nodes.

- This is an old vector-architecture concern that is of critical importance for strong scaling on today and tomorrow’s architectures.

---

Example of a High $n_{1/2}$

- Electromagnetics code on GPUs of OLCF’s Titan.
- Even with $P=1$, $n/P$ is limited to be $> 125,000$ to get saturated performance per node (for CFD, this number is higher).
\[ n^{1/2} \quad \textbf{Requirements for a Candidate Node to Yield Speedup} \]

\[
S_P = \eta \cdot S_1 \cdot P \quad \text{(speed, in mflops)}
\]

\[
S_1 = \text{observed saturated speed, in flops, } n \gg 1
\]

- Let \( n \) be total problem size (gridpoints, say), and \( n_{1/2} \) be the local problem size such that

\[
S_1(n_{1/2}) = \frac{1}{2}S_1(n_{\text{sat}})
\]

- Let \( W := w \cdot n \) be the total number of flops and \( w \) be the number of flops per gridpoint.

- Choose \( P = n/n_{1/2} \) \hspace{1cm} (50\% efficiency)

- Time to solution:

\[
T_P = \frac{w \cdot n}{S_P} = \frac{w \cdot n}{\eta S_1 P}
\]

\[
= \frac{w \cdot n}{\eta S_1(n/n_{1/2})} = \frac{w \cdot n_{1/2}}{1/2S_1}
\]

\[
= 2w \left( \frac{n_{1/2}}{S_1} \right)
\]
Summary: Getting Good Performance for CFD, PDEs, HPC

- Know your model & physics (if possible):
  - RANS vs LES – $10^4$ – $10^5$ savings

- PDEs:
  - Discretizations can yield 10x savings
  - Fast solvers 10-100x

- HPC:
  - Understand your single-node performance, over a range of n!
  - Examine your communication costs.
  - All of this involves modeling and measuring.

- The HPC part can be almost (!) as much fun as the fluids part!
Thank You!