New Math for Exascale Computational Science?

Ulrich Ruede
Lehrstuhl fuer Simulation

Department of Computer Science
University of Erlangen

DOE ExaScale Math Workshop
Washington, DC
August 2013
Where we Are: Gaps in Math Research for ExaScale

- Mathematics is possibly the most important contributor to Exa-Scale Science
  - optimal asymptotic complexity necessary for scalability
- However
  - algorithms often underperform by orders of magnitude: performance abyss
  - tweaking bad codes is not enough
  - fundamental deficiencies

- Nothing is more practical than a good theory, but
  - misconception about the role of qualitative rigorous theory
  - $\|e\| \leq C h^p$ is not sufficient for efficiency when $C$ is a unspecified constant
- We need in addition
  - more quantitative theory
  - systematic algorithmic benchmarking
  - new cost metric = better complexity models: not just $O(N)$ Flops
    - must account for data movement or energy consumption, etc.
Performance Oriented Algorithm Design (1)

- **predict** a-priori what performance is achievable:
  - white box performance model
- **evaluate** concrete algorithm/implementation against prediction
- **account** for discrepancies
  - refine assumptions
  - **improve** performance model
  - **improve** implementation
  - **improve** solver, discretization, model
- With other words:

  performance driven co-design!
Performance Oriented Algorithm Design (2)

Example: Multigrid solves 2D Poisson in $30N$ operations ($O(h^2)$ 5-pt stencil)

- consequently, we should solve
  - on a 100 Gflops CPU: a 1000x1000 grid in 0.0003 seconds
- **If not**, we must question
  - the discretization
  - the solver
  - the implementation
  - the performance model
  - ...
- There are good reasons to be slower, but we must **use scientific methodology to understand** why and how
- When we burn 20 MW and miss the best published result by 4 orders of magnitude then it is not enough to state „our code is not fully optimized yet“.
- **Where are** the math papers publishing concrete complexity constants for other problems (Possion 3D, Stokes, Lame, ...)?
- **New theory** must relate algorithms better to hardware performance characteristics (such as mem bandwidth, peak flops, energy consumption, ...)

Ulrich Ruede               Exa-Scale Mathematics Workshop, Washington, DC, August 2013
Is Multigrid Doomed?

Multigrid for structured refinement of an unstructured tetrahedral mesh

Use regular HHG patches for partitioning the domain

communication of ghost layers

Coarse grid with 132k elements, as assigned to supercomputer
Multigrid Scalability on JuQueen for Stokes

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Threads</th>
<th>Grid points</th>
<th>Resolution</th>
<th>Time: (A)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>$2.1 \cdot 10^7$</td>
<td>32 km</td>
<td>30 s</td>
<td>89 s</td>
</tr>
<tr>
<td>4</td>
<td>240</td>
<td>$1.6 \cdot 10^8$</td>
<td>16 km</td>
<td>38 s</td>
<td>114 s</td>
</tr>
<tr>
<td>30</td>
<td>1920</td>
<td>$1.3 \cdot 10^9$</td>
<td>8 km</td>
<td>40 s</td>
<td>121 s</td>
</tr>
<tr>
<td>240</td>
<td>15 360</td>
<td>$1.1 \cdot 10^{10}$</td>
<td>4 km</td>
<td>44 s</td>
<td>133 s</td>
</tr>
<tr>
<td>1 920</td>
<td>122 880</td>
<td>$8.5 \cdot 10^{10}$</td>
<td>2 km</td>
<td>48 s</td>
<td>153 s</td>
</tr>
<tr>
<td>15 360</td>
<td>983 040</td>
<td>$6.9 \cdot 10^{11}$</td>
<td>1 km</td>
<td>54 s</td>
<td>170 s</td>
</tr>
</tbody>
</table>

Coupled with Energy Equation: Buoyancy Driven Flow

Largest run: fully implicit - coupled
4.1 trillion tetrahedra
3.4 trillion DOF

Ulrich Ruede
Exa-Scale Mathematics Workshop, Washington, DC, August 2013
Evaluation

• Matrix free implementation essential
• HHG - semistructured grid essential
• Careful design and tuning from node to system level
• Can solve $>10^{12}$ DOF Stokes Eqn with multigrid in $<1$ min
• good parallel efficiency on $\sim$1 million parallel threads
• reaches $<10\%$ peak performance on multi-PetaScale machines
• node performance not bandwidth limited
• limit by instruction throughput (complex load/store & mem alignment pattern)
• non-scalable (CG-type) coarse grid solver limits overall scalability - though this is not as dramatic as common wisdom would tell
• Stokes-Solver via Schur complement (pressure correction) will be improved
• Better discretizations are out there
  • but naively implemented LBB-stable FE would drive complexity through the roof!
The **recirculatory system** contains
- ca. 0.006 m$^3$ volume
- discretize with $10^{12}$ finite volumes
- mesh size of 0.02 mm
- $10^6$ operations per second and per volume.
- ca. $2.5 \times 10^{13}$ red blood cells
- $4 \times 10^4$ flops per second and blood cell

The **brain** has
- ca. $10^{11}$ Neurons
- $10^7$ flops per sec and neuron
- ca. 0.0015 m$^3$ volume
- discretize with $10^{14}$ finite elements
- resolve the brain with a mesh size ~0.0025 mm
- $10^4$ operations per second and per element.
Conclusions (inspired by David Bailey)

**Exa-Scale will be Easy!**

- If parallel efficiency is bad, choose a slower serial algorithm
  - it is probably easier to parallelize
  - and will make your speedups look much more impressive
- Introduce the “CrunchMe” variable for getting high Flops rates
  - advanced method: disguise CrunchMe by using an inefficient (but compute-intensive!) algorithm from the start
- Introduce the “HitMe” variable to get good cache hit rates
  - advanced version: Implement HitMe in the „Hash-Brown Lookaside Table for the Multi-Threatened Cash-Filling Clouded Tree“ data structure
  - ... impressing yourself and others
- Never cite “time-to-solution”
  - who cares whether you solve a real-life problem anyway
  - it is the MachoFlops that interest the people who pay for your research
- Never waste your time by trying to use a complicated algorithm in parallel
  - Use Primitive Algorithm => Easy to Maximize your MachoFlops
  - A few billion CPU hours can easily save you days of reading in boring math books
We need the magic bullet!

Animation by Simon Bogner, LSS with massively parallel LBM