Parallel Textbook Multigrid Efficiency

U. Rüde (FAU Erlangen, ulrich.ruede@fau.de)
joint work with B. Gmeiner, M. Huber, H. Stengel, H. Köstler (FAU)
C. Waluga, M. Huber, L. John, B. Wohlmuth (TUM)
M. Mohr, S. Bauer, J. Weismüller, P. Bunge (LMU)

Lehrstuhl für Simulation
FAU Erlangen-Nürnberg

FOURTEENTH COPPER MOUNTAIN CONFERENCE
ON
ITERATIVE METHODS

March 20 – March 25, 2016
Two questions:

What is the minimal cost of solving a PDE?
(such as Poisson’s or Stokes’ equation in 3D)
- asymptotic results of the form
  \[
  \text{Cost} \leq C n^p \quad \text{(Flops)}
  \]
  with unspecified constant C are inadequate to predict the real performance

Flops as metric become questionable

How do we quantify true cost?
(i.e. resources needed)
- Number of flops?
- Memory consumption?
- Memory bandwidth? (aggregated?)
- Communication bandwidth?
- Communication latency?
- Power consumption?
Two Multi-PetaFlops Supercomputers

**JUQUEEN**
- Blue Gene/Q architecture
- 458,752 PowerPC A2 cores
- 16 cores (1.6 GHz) per node
- 16 GiB RAM per node
- 5D torus interconnect
- 5.8 PFlops Peak
- 448 TByte memory
- TOP 500: #11

**SuperMUC**
- Intel Xeon architecture
- 147,456 cores
- 16 cores (2.7 GHz) per node
- 32 GiB RAM per node
- Pruned tree interconnect
- 3.2 PFlops Peak
- TOP 500: #23

SuperMuc: 3 PFlops
How big a PDE problem can we solve?

- 400 TByte main memory = $4 \times 10^{14}$ Bytes = 5 vectors each with $10^{13}$ elements
- 8 Byte = double precision

- even with a sparse matrix format, storing a matrix of dimension $10^{13}$ is not possible on Juqueen
  - matrix-free implementation necessary

Which algorithm?
- multigrid
  - asymptotically optimal complexity: Cost = $C \times N$
  - $C$ „moderate“
- does it parallelize well?
  - overhead?

equal, not „≤“
### Energy

<table>
<thead>
<tr>
<th>computer generation</th>
<th>gigascale: $10^9$ FLOPS</th>
<th>terascale $10^{12}$ FLOPS</th>
<th>petascale $10^{15}$ FLOPS</th>
<th>exascale $10^{18}$ FLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>desired problem size DoF=$N$</td>
<td>$10^6$</td>
<td>$10^9$</td>
<td>$10^{12}$</td>
<td>$10^{15}$</td>
</tr>
<tr>
<td>energy estimate (kWh) 1 Njoule x $N^2$ all-to-all communication</td>
<td>0.278 Wh 10 min of LED light</td>
<td>278 kWh 2 weeks blow drying hair</td>
<td>278 GWh 1 month electricity for Denver</td>
<td>278 PWh 100 years world electricity production</td>
</tr>
<tr>
<td>TerraNeo prototype (kWh)</td>
<td>0.13 Wh</td>
<td>0.03 kWh</td>
<td>27 kWh</td>
<td>?</td>
</tr>
</tbody>
</table>

At extreme scale: optimal complexity is a must!
Hierarchical Hybrid Grids (HHG)

B. Bergen, F. Hülseman, U. Rüde, G. Wellein: ISC Award 2006: „Is $1.7 \times 10^{10}$ unknowns the largest finite element system that can be solved today?“, SuperComputing, 2005.

- Parallelize „plain vanilla“ multigrid for tetrahedral finite elements
  - partition domain
  - parallelize all operations on all grids
  - use clever data structures
  - matrix free implementation
- Do not worry (so much) about coarse grids
  - idle processors?
  - short messages?
  - sequential dependency in grid hierarchy?
- Elliptic problems always require global communication. This cannot be accomplished by
  - local relaxation or
  - Krylov space acceleration or
  - domain decomposition without coarse grid
HHG: A modern architecture for FE computations

Geometrical Hierarchy: Volume, Face, Edge, Vertex
Copying to update ghost points

Linearization & memory representation

MPI message to update ghost points

Process boundary
Towards a holistic performance engineering methodology:

Parallel Textbook Multigrid Efficiency

Brandt, A. (1998). Barriers to achieving textbook multigrid efficiency (TME) in CFD.
Textbook Multigrid Efficiency (TME)

“Textbook multigrid efficiency means solving a discrete PDE problem with a computational effort that is only a small (less than 10) multiple of the operation count associated with the discretized equations itself.” [Brandt, 98]

- work unit (WU) = single elementary relaxation
- classical algorithmic TME-factor:
  ops for solution/ ops for work unit
- new parallel TME-factor to quantify
  - algorithmic efficiency
  - combined with implementation scalability
Textbook Multigrid Efficiency (TME)

Full multigrid method (FMG):

- linear tetrahedral elements
- One $V(2,2)$-cycle with SOR smoother on each new level reduces the algebraic error to 70% of discretization error.
- Textbook efficiency for 3D Poisson problem: $6.5 \text{ WU}$

Residual reduction for multigrid $V$- and $W$-cycles vs. work units (WU)


extended TME paradigm for parallel performance analysis

- Analyse cost of an elementary relaxation
  - micro-kernel benchmarks for smoother
    - optimize node level performance
    - justify/remove any performance degradation
  - aggregate performance for whole parallel system defines a WU

- Measure parallel solver performance

- Compute TME factor as overall measure of efficiency
  - analyse discrepancies
  - identify possible improvements

- Optimize TME factor
Parallel TME

\( \mu_{\text{sm}} \)  # of elementary relaxation steps on single core/sec
\( U \)  # cores
\( U \mu_{\text{sm}} \)  aggregate peak relaxation performance

\[
T_{\text{wu}}(N, U) = \frac{N}{U \mu_{\text{sm}}} \quad \text{idealized time for a work unit}
\]

\( T(N, U) \)  time to solution for \( N \) unknowns on \( U \) cores

Parallel textbook efficiency factor

\[
E_{\text{ParTME}}(N, U) = \frac{T(N, U)}{T_{\text{wu}}(N, U)} = T(N, U) \frac{U \mu_{\text{sm}}}{N}
\]

combines algorithmic and implementation efficiency.
TME Efficiency Analysis: RB-GS Smoother

```c
for (int i=1; i < (tsize-j-k-1); i=i+2) {
}
```

This loop should be executed on **single SuperMuc core** at

- **720 M updates/sec** *(in theory* - peak performance)*
- **\( \mu_{sm} = 176 \text{ M} \)** updates/sec *(in practice - memory access bottleneck; RB-ordering prohibits vector loads)*

Thus **whole SuperMuc** should perform

- **\( U \mu_{sm} = 147456 \times 176 \text{ M} \approx 26T \) updates/sec**
Execution-Cache-Memory Model (ECM)


ECM model for the 15-point stencil on SNB core.

Arrow indicates a 64 Byte cache line transfer.

Run-times represent 8 elementary updates.
TME and Parallel TME results

<table>
<thead>
<tr>
<th>Setting/Measure</th>
<th>$E_{\text{TME}}$</th>
<th>$E_{\text{SerTME}}$</th>
<th>$E_{\text{NodeTME}}$</th>
<th>$E_{\text{ParTME1}}$</th>
<th>$E_{\text{ParTME2}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid points</td>
<td>-</td>
<td>$2 \cdot 10^6$</td>
<td>$3 \cdot 10^7$</td>
<td>$9 \cdot 10^9$</td>
<td>$2 \cdot 10^{11}$</td>
</tr>
<tr>
<td>Processor cores $U$</td>
<td>-</td>
<td>1</td>
<td>16</td>
<td>4096</td>
<td>16384</td>
</tr>
<tr>
<td>(CC) - FMG(2,2)</td>
<td>6.5</td>
<td>15</td>
<td>22</td>
<td>26</td>
<td>22</td>
</tr>
<tr>
<td>(VC) - FMG(2,2)</td>
<td>6.5</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>(SF) - FMG(2,1)</td>
<td>31</td>
<td>64</td>
<td>100</td>
<td>118</td>
<td>-</td>
</tr>
</tbody>
</table>

Three model problems:
- scalar constant (CC) & scalar variable (VC) coefficients
- Stokes solved via Schur complement (SF)

Full multigrid with #iterations such that asymptotic optimality maintained

TME = 6.5 (algorithmically for scalar cases)
ParTME around 20 for scalar PDE, and ≥100 for Stokes
Application to Earth Mantle Convection Models


HHG Solver for Stokes System
Motivated by Earth Mantle convection problem


\[-\nabla \cdot (2\eta\varepsilon(u)) + \nabla p = \rho(T)g,\]
\[\nabla \cdot u = 0,\]
\[\frac{\partial T}{\partial t} + u \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = \gamma.\]

\[\begin{align*}
\mathbf{u} & \quad \text{velocity} \\
\rho & \quad \text{dynamic pressure} \\
T & \quad \text{temperature} \\
\nu & \quad \text{viscosity of the material} \\
\varepsilon(u) = \frac{1}{2}(\nabla u + (\nabla u)^T) & \quad \text{strain rate tensor} \\
\rho & \quad \text{density} \\
\kappa, \gamma, g & \quad \text{thermal conductivity, heat sources, gravity vector}
\end{align*}\]

Scale up to \(\sim 10^{12}\) nodes/DOFs
\[\Rightarrow\text{resolve the whole Earth Mantle globally with 1km resolution}\]

Stokes equation:
\[-\text{div}(\nabla \mathbf{u} - p\mathbf{I}) = \mathbf{f}, \quad \text{div}\mathbf{u} = 0\]

FEM Discretization:
\[\begin{align*}
\mathbf{a}(\mathbf{u}_l, \mathbf{v}_l) + \mathbf{b}(\mathbf{v}_l, p_l) &= \mathbf{L}(\mathbf{v}_l) \quad \forall \mathbf{v}_l \in \mathbf{V}_l, \\
\mathbf{b}(\mathbf{u}_l, q_l) - \mathbf{c}(p_l, q_l) &= 0 \quad \forall q_l \in \mathbf{Q}_l,
\end{align*}\]

with:
\[\begin{align*}
\mathbf{a}(\mathbf{u}, \mathbf{v}) &= \int_{\Omega} \nabla \mathbf{u} : \nabla \mathbf{v} \, d\mathbf{x}, \\
\mathbf{b}(\mathbf{u}, q) &= -\int_{\Omega} \text{div}\mathbf{u} \cdot q \, d\mathbf{x}
\end{align*}\]

Schur-complement formulation:
\[\begin{bmatrix}
\mathbf{A}_l & \mathbf{B}_l^T \\
0 & \mathbf{C}_l + \mathbf{B}_l\mathbf{A}_l^{-1}\mathbf{B}_l^T
\end{bmatrix}
\begin{bmatrix}
\mathbf{u}_l \\
p_l
\end{bmatrix}
= \begin{bmatrix}
\mathbf{f}_l \\
\mathbf{B}_l\mathbf{A}_l^{-1}\mathbf{f}_l
\end{bmatrix}\]
Coupled Flow-Transport Problem

\[-\nabla u + \nabla p = -Ra \hat{r}\]
\[\text{div } u = 0\]
\[\partial_t T + u \cdot \nabla T = Pe^{-1} \Delta T\]

- \(6.5 \times 10^9\) DoF
- 10000 time steps
- run time 7 days
- Mid-size cluster: 288 compute cores in 9 nodes of LSS at FAU
Comparison of Stokes Solvers

Three solvers:
- Schur complement CG (pressure correction) with MG
- MINRES with MG preconditioner for Vector-Laplace
- MG for saddle point problem with Uzawa type smoother

Time to solution without coarse grid solver

residual reduction by $10^{-8}$
with coarse grid solver the difference shrinks

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Threads</th>
<th>DoFs</th>
<th>SCG iter</th>
<th>time s</th>
<th>MINRES iter</th>
<th>time s</th>
<th>Uzawa iter</th>
<th>time s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>$6.6 \cdot 10^7$</td>
<td>28</td>
<td>136.30</td>
<td>65</td>
<td>115.05</td>
<td>7</td>
<td>58.80</td>
</tr>
<tr>
<td>6</td>
<td>192</td>
<td>$5.3 \cdot 10^8$</td>
<td>26</td>
<td>134.26</td>
<td>64</td>
<td>130.56</td>
<td>7</td>
<td>64.40</td>
</tr>
<tr>
<td>48</td>
<td>1536</td>
<td>$4.3 \cdot 10^9$</td>
<td>26</td>
<td>135.06</td>
<td>62</td>
<td>128.34</td>
<td>7</td>
<td>65.03</td>
</tr>
<tr>
<td>384</td>
<td>12288</td>
<td>$3.4 \cdot 10^{10}$</td>
<td>26</td>
<td>135.41</td>
<td>62</td>
<td>128.34</td>
<td>7</td>
<td>64.96</td>
</tr>
<tr>
<td>3072</td>
<td>98304</td>
<td>$2.7 \cdot 10^{11}$</td>
<td>26</td>
<td>139.55</td>
<td>62</td>
<td>133.30</td>
<td>7</td>
<td>66.08</td>
</tr>
<tr>
<td>24576</td>
<td>786432</td>
<td>$2.2 \cdot 10^{12}$</td>
<td>28</td>
<td>154.06</td>
<td>64</td>
<td>139.52</td>
<td>8</td>
<td>78.24</td>
</tr>
</tbody>
</table>
Exploring the Limits …


Multigrid with Uzawa Smoother

Optimized for Minimal Memory Consumption

- $10^{13}$ Unknowns correspond to 80 TByte for the solution vector
- Juqueen has 450 TByte Memory
- matrix free implementation essential

<table>
<thead>
<tr>
<th>nodes</th>
<th>threads</th>
<th>DoFs</th>
<th>iter</th>
<th>time</th>
<th>time w.c.g.</th>
<th>time c.g.</th>
<th>in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>80</td>
<td>$2.7 \cdot 10^9$</td>
<td>10</td>
<td>685.88</td>
<td>678.77</td>
<td>1.04</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>640</td>
<td>$2.1 \cdot 10^{10}$</td>
<td>10</td>
<td>703.69</td>
<td>686.24</td>
<td>2.48</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>5120</td>
<td>$1.2 \cdot 10^{11}$</td>
<td>10</td>
<td>741.86</td>
<td>709.88</td>
<td>4.31</td>
<td></td>
</tr>
<tr>
<td>2560</td>
<td>40960</td>
<td>$1.7 \cdot 10^{12}$</td>
<td>9</td>
<td>720.24</td>
<td>671.63</td>
<td>6.75</td>
<td></td>
</tr>
<tr>
<td>20480</td>
<td>327680</td>
<td>$1.1 \cdot 10^{13}$</td>
<td>9</td>
<td>776.09</td>
<td>681.91</td>
<td>12.14</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions and Outlook

- Resilience with multigrid
- Multigrid scales to Peta and beyond
- HHG: lean and mean implementation, excellent time to sol.
- Reaching $10^{13}$ DoF