Is 2.44 trillion unknowns the largest finite element system that can be solved today?

U. Rüde (LSS Erlangen, ruede@cs.fau.de)

Lehrstuhl für Informatik 10 (Systemsimulation)
Universität Erlangen-Nürnberg
www10.informatik.uni-erlangen.de

Advances in Numerical Algorithms and High Performance Computing

University College London
April 14-15, 2014
Overview

- Motivation
  - How fast our computers are
  - Some aspects of computer architecture
  - Parallelism everywhere
- Where we stand: Scalable Parallel Multigrid
  - Matrix-Free Multigrid FE solver
  - Hierarchical Hybrid Grids (HHG)
- Multiphysics applications with multigrid and beyond
  - Geodynamics: Terra-Neo
- What I will not talk about today:
  - Electron beam melting
  - Fully resolved 2 and 3-phase bubbly and particulate flows
  - Electroosmotic flows
  - LBM, granular systems, multibody dynamics, GPUs/accelerators, Medical Applications, Image Processing, Real Time Applications

Conclusions
High Performance Computer Systems
(on the way to Exa-Flops)
## Computer Architecture is Hierarchical

### Core
- Vectorization (SSE, AVX), i.e., vectors of 2-8 floating point numbers must be treated in blocks:
- May have their own cache memories
- Access to local (cache) mem fast
- Access to remote mem slow
- Pipelining, superscalar execution
- Each core may need several threads to hide memory access latency

### Node
- Several CPU chips (2) may be combined with local memory to become a node
- Several cores (8) are on a CPU chip
- Within a node we can use „shared memory parallelism“
  - E.g., OpenMP
- Several cores may share second/third level caches
- Memory access bottlenecks may occur
- Sometimes nodes are equipped with accelerators (i.e., graphics cards)

### Cluster
- Thousands of nodes are connected by a fast network
- Different network topologies
- Between nodes message passing must be used
  - E.g., MPI
- High latency
- Low bandwidth
What will Computers Look Like in 2020?

- Super Computer (Heroic Computing)
  - Cost: 200 Million €
  - Parallel Threads: $10^8 - 10^9$
  - $10^{18}$ FLOPS, Mem: $10^{15}$-$10^{17}$ Byte (1-100 PByte)
  - Power Consumption: 20 MW

- Departmental Server (Mainstream Computing for R&D)
  - Cost: 200 000 €
  - Parallel Threads: $10^5 - 10^6$
  - $10^{15}$ FLOPS, Mem: $10^{12}$-$10^{14}$ Byte (1-100 TByte)
  - Power Consumption: 20 KW

- (mobile) Workstation (Computing for the Masses)
  - ... scale down by another factor 100

  but remember: Predictions are difficult ... especially those about the future
## What Supercomputers are Like Today

<table>
<thead>
<tr>
<th></th>
<th>JUQUEEN</th>
<th>SuperMUC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System</strong></td>
<td>IBM Blue Gene/Q</td>
<td>IBM System x iDataPlex</td>
</tr>
<tr>
<td><strong>Processor</strong></td>
<td>IBM PowerPC A2</td>
<td>Intel Xeon E5-2680 8C</td>
</tr>
<tr>
<td><strong>SIMD</strong></td>
<td>QPX (256bit)</td>
<td>AVX (256bit)</td>
</tr>
<tr>
<td><strong>Peak</strong></td>
<td>5872.0 TFlop/s</td>
<td>3185.1 TFlop/s</td>
</tr>
<tr>
<td><strong>Clock</strong></td>
<td>1.6 GHz</td>
<td>2.8 GHz</td>
</tr>
<tr>
<td><strong>Nodes</strong></td>
<td>28 672</td>
<td>9 216</td>
</tr>
<tr>
<td><strong>Node peak</strong></td>
<td>204.8 GFlops/s</td>
<td>358.4 GFlops/s</td>
</tr>
<tr>
<td><strong>S/C/T per Node</strong></td>
<td>1/16/64</td>
<td>2/16/32</td>
</tr>
<tr>
<td><strong>GFlops per Watt</strong></td>
<td>2.54</td>
<td>0.94</td>
</tr>
</tbody>
</table>

*SuperMuc: 3 PFlops*
Let’s try to quantify:

- What are the limiting resources?
  - Floating Point Operations/sec (Flops)
  - Memory capacity
  - Communication/ memory access, ...?

- What is the capacity of a contemporary supercomputer?
  - Flops?
  - Memory?
  - Memory and communication bandwidth?

- What are the resource requirements (e.g. to solve Laplace or Stokes) in
  - Flops/ Dof?
  - Memory/ Dof?

... isn’t it surprising that there are hardly any publications that quantify efficient computing in this form?
Estimating the cost complexity of FE solvers

- $10^6$ unknowns
  - memory requirement:
  - solution vector: 8 M Bytes
  - plus 3 auxiliary vectors: 32 MBytes
  - stiffness & mass matrix, assume #nnz per row 15 (linear tet elements): 240 MBytes
  - can save $O(10)$ cost by matrix-free implementation

- assume asymptotically optimal solver (multigrid for scalar elliptic PDE)
  - 100 Flops/unknown
  - efficiency $\eta=0.1$

- machine with
  - 1 GFlops, 100 MByte, should solve: $3 \times 10^6$ unknowns in 3 seconds
  - 1 PFlops, 100 TByte, should solve: $3 \times 10^{12}$ unknowns in 3 seconds
What good are $10^{12}$ Finite Elements?

- Earth's oceans together have ca. $1.3 \times 10^9$ km$^3$
  - We can resolve the volume of the planetray ocean globally with ca. 100m resolution
- Earth's mantle has $0.91 \times 10^{12}$ km$^3$
  - We can resolve the volume of the mantle with ca. 1 km
- The human recirculatory system contains
  - ca. 0.006 m$^3$ volume
    - discretize with $10^{15}$ finite elements
    - mesh size of 2 μm
    - Exa-Scale: $10^3$ operations per second and per volume.
  - a red blood cell is ca. 7μm large
  - we have ca. $2.5 \times 10^{13}$ red blood cells
    - with an exa-scale system we can spend $4 \times 10^4$ flops per second and per blood cell
Towards Scalable Algorithms and Data Structures
What are the problems?

- Unprecedented levels of parallelism
  - maybe billions of cores/threads needed
- Hybrid architectures
  - standard CPU
  - vector units (SSE, AVX)
  - accelerators (GPU, Intel Xeon Phi)
- Memory wall
  - memory response slow: latency
  - memory transfer limited: bandwidth
- Power considerations dictate
  - limits to clock speed => multi core
  - limits to memory size (byte/flop)
  - limits to address references per operation
  - limits to resilience
Designing Algorithms!
with four strong jet engines.

Would you want to propel a Superjumbo

Large Scale Simulation Software

or with 1,000,000 blow dryer fans?

Moderately Parallel Computing

Massively Parallel MultiCore Systems
The Energy Problem

Elements/nodes

10 elements/node

Inverse matrix

Coulomb interaction

10\times10

Data movements

Each 1 NanoJoule (optimistic)

Together: 10^{15} Joule

Figure 5: With new scaling rules and massive growth in parallelism, data locality is increasingly important. This diagram shows the cost of a double-precision multiply-add at different levels of the memory hierarchy (from the cost of performing the flop to the movement of data operands from registers, different distances on-chip, and distances off-chip.)

The model or FPU and register access is based on the Tensilica LX2 core energy model, the energy consumed for cross-chip wires uses the Orion2 power model which is based on Balfour's model [6], memory access is based on the JEDEC DDRx memory roadmap, and cross-system is based on projections for VCSEL-based optical transceivers.

2.4 New Communication and Synchronization Mechanisms

Increased use of non-conventional (non-cache-coherent) methods for interprocessor communication on chip (e.g. GPU Warps and Intel SCC inter-processor hardware message queues) suggests new approaches are needed for expressing data locality so that runtime systems can manage data movement instead of increasing the burden on the application programmer.

Configurable Caches:

Increased use of configurable caches (automatic vs. user-managed) to explicitly manage vertical data movement. Motivates new APIs for making effective use of these new hardware locality management features in a cross-platform manner.

Coordinated Power Management:

Local power management decisions (such as Intel/Sandybridge thermal-throttling of FPU clock rates) will result in system-wide performance inhomogeneity. This motivates systemwide coordinated power management.

Energy Aware Data Movement/Load-Balancing:

New sources of transient load-imbalances are increasing (such as software-based fault resilience mechanisms, and localized power management), but rebalancing the load may be more costly than tolerating it. Balancing the cost of fixing a load-imbalance vs. tolerating its continuation requires an API for expressing the energy-costs of data movement to the application and runtime.

277 GWh or 240 Kilotons TNT

the picture shows the Badger-Explosion of 1953 with 23 Kilotons TNT

Multigrid for FE on Peta-Scale Computers
Multigrid: V-Cycle

Goal: solve $A^h u^h = f^h$ using a hierarchy of grids

1. **Relax on**
   $$A^h u^h = f^h$$

2. **Residual**
   $$r^h = f^h - A^h u^h$$

3. **Restrict**
   $$r^H = I_{H}^{H} r^h$$

4. **Interpolate**
   $$e^h = I_{H}^{H} e^H$$

5. **Solve**
   $$A^H e^H = r^H$$

6. **Correct**
   $$u^h \leftarrow u^h + e^h$$

... by recursion...
How fast can we make FE multigrid

- 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
Hierarchical Hybrid Grids (HHG)

- Joint work with Frank Hülsemann (now EDF, Paris), Ben Bergen (now Los Alamos), T. Gradl (Erlangen), B. Gmeiner (Erlangen)

**HHG Goal: Ultimate Parallel FE Performance!**

- unstructured coarse refinement grid with
  - regular substructures for
  - efficiency
  - superconvergence effects
- matrix-free implementation using regular substructures
  - constant stencil when coefficients are constant
  - assembly-on-the-fly for variable coefficients
HHG refinement example

Input Grid
HHG Refinement example

Refinement Level one
HHG Refinement example

Refinement Level Two
HHG Refinement example

Structured Interior
HHG Refinement example

Structured Interior
HHG Refinement example

Edge Interior
HHG Refinement example

Edge Interior
Regular tetrahedral refinement

- Structured refinement of tetrahedra
- Use regular HHG patches for partitioning the domain (only 2D for simplification)

Coarse grid with 132k elements, as assigned to supercomputer

Communication of ghost layers
HHG Parallel Update Algorithm

for each vertex do
    apply operation to vertex
end for
update vertex primary dependencies

for each edge do
    copy from vertex interior
    apply operation to edge
    copy to vertex halo
end for
update edge primary dependencies

for each element do
    copy from edge/vertex interiors
    apply operation to element
    copy to edge/vertex halos
end for
update secondary dependencies
<table>
<thead>
<tr>
<th>#Cores</th>
<th>Coarse Grid</th>
<th>Unkn (x 10^6)</th>
<th>Crse Grd Its</th>
<th>Tme to soln</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>3072</td>
<td>1070</td>
<td>20</td>
<td>5,66</td>
</tr>
<tr>
<td>512</td>
<td>6144</td>
<td>2142</td>
<td>25</td>
<td>5,69</td>
</tr>
<tr>
<td>1024</td>
<td>12288</td>
<td>4286</td>
<td>30</td>
<td>5,71</td>
</tr>
<tr>
<td>2028</td>
<td>24576</td>
<td>8577</td>
<td>45</td>
<td>5,75</td>
</tr>
<tr>
<td>4096</td>
<td>49152</td>
<td>17158</td>
<td>60</td>
<td>5,92</td>
</tr>
<tr>
<td>8192</td>
<td>98304</td>
<td>34326</td>
<td>70</td>
<td>5,86</td>
</tr>
<tr>
<td>16384</td>
<td>196608</td>
<td>68669</td>
<td>90</td>
<td>5,91</td>
</tr>
<tr>
<td>32768</td>
<td>393216</td>
<td>137355</td>
<td>105</td>
<td>6,17</td>
</tr>
<tr>
<td>65536</td>
<td>786432</td>
<td>274743</td>
<td>115</td>
<td>6,41</td>
</tr>
<tr>
<td>131072</td>
<td>1572864</td>
<td>549554</td>
<td>145</td>
<td>6,42</td>
</tr>
<tr>
<td>262144</td>
<td>3145728</td>
<td>1099276</td>
<td>280</td>
<td>6,82</td>
</tr>
<tr>
<td>294912</td>
<td>294912</td>
<td>824365</td>
<td>110</td>
<td>3,80</td>
</tr>
</tbody>
</table>

Parallel scalability of scalar elliptic problem in 3D discretized by tetrahedral finite elements.

Times to solution on *Jugene*, Jülich

Largest problem solved: $1.099 \times 10^{12}$ DOFS (6 trillion tetrahedra) on 262000 processors.

Parallel Efficiency of HHG

- JUGENE
- JUQUEEN
- SuperMUC

Problem Size

- 1 Node Card
- 1 Midplane
- 1 Island
- Hybrid Parallel

Parallel Efficiency

- 1.0E+07
- 1.0E+08
- 1.0E+09
- 1.0E+10
- 1.0E+11
- 1.0E+12
- 1.0E+13

131,072 cores
262,144 cores
393,216 cores
HHG Pros and Cons

Pro:

- node performance
  - SIMD, superscalar execution, GPUs
- better accuracy through local superconvergence effects
- well suited for parallelization
- tau-extrapolation for higher order
- locally selected line/plane smoothers for better efficiency

Con:

- only restricted adaptivity possible
- only limited ability to handle complex shapes
- how to solve the coarse grid problem
- high implementation effort
- less flexible
Algorithm Performance Analysis
Towards Quantitative performance prediction of multigrid for tetrahedra

B. Gmeiner, T. Gradl, F. Gaspar, UR:
Optimization of the multigrid-convergence rate on semi-structured meshes by local Fourier analysis, Computers & Mathematics with Applications, 2013

- Goal: Maintain V-cycle convergence rates <0.2 uniformly by using optimally tuned smoothers in each macro-tetrahedron
- Local Mode Analysis (LFA) for the quantitative prediction of
  - smoothing rate
  - two grid convergence rate
  - V-cycle/ W-cycle performance
- Idea: analyse multigrid in (discrete) Fourier space
  - neglecting effects of boundary conditions
  - quantitative analysis of coupling between high and low frequency modes
  - technically complex in particular for multi-color smoothers
- Generalization of classical multigrid analysis technique to tetrahedral meshes
- Analysis as an algorithm design tool
LFA smoothing factors, $\mu^{v_1+v_2}$, LFA two-grid convergence factors, $\rho$, and measured $W$-cycle convergence rates, $\rho_h$, for an optimized tetrahedron.

<table>
<thead>
<tr>
<th>$v_1$, $v_2$</th>
<th>Damped Jacobi</th>
<th>Gauss-Seidel</th>
<th>Four-color</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu^{v_1+v_2}$</td>
<td>$\rho$</td>
<td>$\rho_h$</td>
<td>$\mu^{v_1+v_2}$</td>
</tr>
<tr>
<td>1, 0</td>
<td>0.720</td>
<td>0.602</td>
<td>0.598</td>
</tr>
<tr>
<td>1, 1</td>
<td>0.517</td>
<td>0.362</td>
<td>0.360</td>
</tr>
</tbody>
</table>

- Degenerated Tetrahedra lead to reduced smoothing rates and to poor multigrid convergence rates.
- Moderate degeneration:
  - compensate by locally adapted number of smoothing steps
  - introduce special smoothers where necessary
Coarse mesh with tetrahedra of different type
Refined to beyond $10^8$ elements
Tet degeneracy measure: longest/shortest edge: 0.214

An example

Measured convergence factors for different smoothing strategies.

<table>
<thead>
<tr>
<th>Smoothing strategy</th>
<th>W-cycle</th>
<th>V-cycle</th>
<th>$\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unoptimized</td>
<td>0.64</td>
<td>0.65</td>
<td>1.0</td>
</tr>
<tr>
<td>Exact block-wise solving</td>
<td>0.11</td>
<td>0.18</td>
<td>–</td>
</tr>
<tr>
<td>Adaptive smoothing steps</td>
<td>0.51</td>
<td>0.53</td>
<td>1.0</td>
</tr>
<tr>
<td>+ Additional interface smoothing</td>
<td>0.34</td>
<td>0.34</td>
<td>1.0</td>
</tr>
<tr>
<td>Full damping</td>
<td>0.59</td>
<td>0.56</td>
<td>1.15</td>
</tr>
<tr>
<td>Interior damping</td>
<td>0.44</td>
<td>0.49</td>
<td>1.55/1.0</td>
</tr>
<tr>
<td>+ Additional interface smoothing</td>
<td>0.30</td>
<td>0.30</td>
<td>1.55/1.0</td>
</tr>
<tr>
<td>Adaptive interior damping</td>
<td>0.46</td>
<td>0.51</td>
<td>Variable</td>
</tr>
<tr>
<td>+ Additional interface smoothing</td>
<td>0.31</td>
<td>0.34</td>
<td>Variable</td>
</tr>
<tr>
<td>Combined Methods</td>
<td>0.15</td>
<td>0.19</td>
<td>1.55/1.0</td>
</tr>
</tbody>
</table>
Application in Geophysics
TERRA-NEO
Co-Design of an Exascale Earth Mantle Modeling Framework

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>p</td>
<td>dynamic pressure</td>
</tr>
<tr>
<td>T</td>
<td>\nu</td>
<td>viscosity of the material</td>
</tr>
<tr>
<td>\epsilon(u)</td>
<td>= \frac{1}{2}(\nabla u + (\nabla u)^T)</td>
<td>strain rate tensor</td>
</tr>
<tr>
<td>\rho</td>
<td>\kappa, \gamma, g</td>
<td>density, thermal conductivity, heat sources, gravity vector</td>
</tr>
</tbody>
</table>

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

FEM Discretization:
\[a(u_l, v_l) + b(v_l, p_l) = L(v_l) \quad \forall v_l \in V_l, \]
\[b(u_l, q_l) - c(p_l, q_l) = 0 \quad \forall q_l \in Q_l, \]

with:
\[a(u, v) := \int_{\Omega} \nabla u : \nabla v \, dx, \quad b(u, q) := -\int_{\Omega} \text{div} u \cdot q \, dx \]

Schur-complement formulation:
\[
\begin{bmatrix}
A_l & B_l^T \\
0 & C_l + B_l A_l^{-1} B_l^T
\end{bmatrix}
\begin{bmatrix}
u_l \\
p_l
\end{bmatrix}
= \begin{bmatrix}
f_l \\
B_l A_l^{-1} f_l
\end{bmatrix}
\]

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

DFG SPP 1648/1 - Software for Exascale Computing
Gmeiner, Waluga, Stengel, Wohlmuth, UR:
Starting with an icosahedron refining until $10^{12}$ FE are reached
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 \times 10^{07}</td>
<td>32 km</td>
<td>30 s</td>
<td>89 s</td>
</tr>
<tr>
<td>4</td>
<td>240</td>
<td>1.6 \times 10^{08}</td>
<td>16 km</td>
<td>38 s</td>
<td>114 s</td>
</tr>
<tr>
<td>30</td>
<td>1 920</td>
<td>1.3 \times 10^{09}</td>
<td>8 km</td>
<td>40 s</td>
<td>121 s</td>
</tr>
<tr>
<td>240</td>
<td>1 5360</td>
<td>1.1 \times 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 \times 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 \times 10^{11}</td>
<td>1 km</td>
<td>54 s</td>
<td>170 s</td>
</tr>
</tbody>
</table>

**Graph:**

- Computation
- Communication

**Legend:**

- Pressure correction
- Solve momentum
- Multigrid, finest level
- Multigrid, remaining levels
- Multigrid, CG, residual
- Multigrid, CG, scalar product
Pessimizing the Performance

with greetings from D. Bailey's: *Twelve Ways to Fool the Masses...*

- Bring loops in wrong order, ignore caches, **randomize memory access**, use many small MPI messages
  - $10^{12} \rightarrow 10^{11}$ unknowns

- Do not use a **matrix-free** implementation (keep in mind that a single multiplication with the mass and stiffness matrix can easily cost 50 mem accesses per unknown):
  - $10^{11} \rightarrow 10^{10}$ unknowns

- Gain additional flexibility by using **unoptimized unstructured** grids (indirect mem access costs!)
  - $10^{10} \rightarrow 10^9$ unknowns

- Increase **algorithmic overhead**, e.g. permanently checking convergence, use the most expensive error estimator, etc. etc.
  - $10^9 \rightarrow 10^8$ unknowns ( ... still a large system ... )
Parallel Textbook
Multigrid Efficiency
Textbook Multigrid Efficiency (TME)?

- TME introduced by Brandt to distinguish between „optimal“ and „fast“ algorithms
  - The cost of solution should be less than 10 times the cost of a Gauss-Seidel relaxation (or Matrix-vector multiply) of the system

- We define the
  - algorithmic TME-factor:
    - operations for solution/ operations for a single relaxation sweep
  - parallel TM-scalability factor
    - time for solution / (time for elementary relaxation × #unknowns)

- For the Stokes system: 4 equations (3x velocity, 1 pressure)
  - Idealized sweep is possible at 26/4 ≈ 7 (× 10^{12} grid points/sec)
  - this ignores cost of communication

- Full SuperMuc solves 6.9×10^{11} mesh points in 41 seconds
  - equivalent to >2.44 ×10^{12} unknowns
  - for residual reduction by 10^3
    - sufficient for FMG solution or
    - single time step
for (int i=1; i < (tsize-j-k-1); i=i+2) {
}

This loop should be executed on each SuperMuc core at:
- 720 M updates/sec (in theory - peak performance)
- 176 M updates/sec (in practice - memory access bottleneck; RB-ordering prohibits vector loads)

Thus whole SuperMuc should perform:
- 147456*176M ≈ 26T elementary updates/sec
Do we reach TME?

- Our overall TM-scalability factor is
  - $41 \text{ sec}/(2.44\times10^{12}/26\times10^{12}) \approx 386$
  - the solution is 386 times as expensive as an idealized single relaxation sweep

- Why do we fail reach a TM-scalability factor $\lesssim 10$?
  - have ignored (parallel) overheads, in particular the reduced efficiency of parallel MG on coarser grids
  - pressure correction Schur solver is suboptimal
    - costing 7 CG iterations
    - requiring the CG preconditioner sweep, etc.
    - error reduction by $10^3$ is over-solving
  - Multigrid only as preconditioner for scalar velocity systems
  - Chance for further tuning of parameters, i.e. are V(3,3) cycles optimal?
TM Scalability as Guideline for Performance Engineering

- Quantify the cost of an elementary relaxation
  - through systematic micro-kernel benchmarks
  - optimize kernel performance, justify any performance degradation
  - compute idealized aggregate performance for large scale parallel system

- Evaluate (parallel) solver performance

- Quantify TM-scalability factor
  - as overall measure of efficiency
  - analyse discrepancies
  - identify possible improvements
  - improve TM-scalability factor
What next (1): Towards Asynchronous Iteration

Basic Observation for Iterative Methods
- Deterministic a-priori ordering of traversal („sweeps“) is artificial
- Other orderings may sometimes
  - be easier to parallelize (e.g. red-black, domain partitioning)
  - converge faster
  - be better suited for hardware (vectorization, SIMD, Cache-aware, ...)

For MG:
- Is it possible to use different traversals
  - within a level?
  - between levels?
What next (2): Use Program Generation Techniques and Domain Specific Languages

- Manual Program Optimization is possible, but tedious
- SPP EXA: Exa-Stencils Project
  - Joint work with C. Lengauer, T. Apel (Passau), M. Bolten (Wuppertal), J. Teich, H. Köstler (Erlangen)
- Design and implementation of a DSL for stencil computations within multigrid
- Automatic generation of efficient code for various hardware platforms
  - node level
  - system level
- Designed for the requirements of multilevel algorithms
Conclusions

- Progress in computer technology and carefully designed algorithms enable FE simulations in excess of $10^{12}$ Dofs ... and it keeps growing
- All computer systems are parallel and we are not well prepared for this disruptive change
- We need a new algorithm engineering methodology, based on a better performance analysis and prediction
- **Co-Design** of Apps, Models, Discretization, Solver, Software, and Parallelization
- See e.g. the position papers of the DOE Exascale Mathematics Working Group
  - Workshop in August 2013
  - [https://collab.mcs.anl.gov/display/examath/About](https://collab.mcs.anl.gov/display/examath/About)
Thank you for your attention!

Questions?

Video generated at LSS with the massively parallel waLberla Software framework for Lattice Boltzmann based multi-physics applications.

Slides, reports, thesis, animations available for download at:
www10.informatik.uni-erlangen.de
Some examples of Coupled Multi Physics Simulations
Motivating Example: Simulation of Electron Beam Melting Process (Additive Manufacturing)

EU-Project Fast-EBM
- ARCAM (Sweden)
- TWI (Cambridge)
- WTM (FAU)
- ZISC (FAU)

Generation of powder bed
Energy transfer by electron beam
- modeling penetration depth

Heat transfer
Flow dynamics
- Melting/solidification
- phase transition
- surface tension
- fluid flow
- wetting, capillary forces

Joint work with C. Körner, M. Markl, R. Ammer
Simulation of Electron Beam Melting

Simulating powder bed generation using the rigid body dynamics

High speed camera shows melting step for manufacturing a hollow cylinder

LBM Simulation

Inter-node Strong Scaling SuperMUC: 16 - 4096 tasks

- 16-256 tasks (512x384x48 cells)
- 64-1024 tasks (512x384x48 cells)
- 256-4096 tasks (2048x1536x48 cells)
Bubbly Flows and Foaming Processes

- 1000 Bubbles
- $510 \times 510 \times 530 = 1.4 \times 10^8$ lattice cells
- 70,000 time steps
- 77 GB
- 64 processes
- 72 hours
- 4,608 core hours

Visualization

- 770 images
- Approx. 12,000 core hours for rendering

Best Paper Award for Stefan Donath (LSS Erlangen) at ParCFD, May 2009 (Moffett Field, USA)
Simulation of Metal Foams

- Example application:
  - Engineering: metal foam simulations

- Based on LBM:
  - Free surfaces
  - Surface tension
  - Disjoining pressure to stabilize thin liquid films
  - Parallelization with MPI and load Balancing

- Collaboration with C. Körner (Dept. of Material Sciences, Erlangen)

- Other applications:
  - Food processing
  - Fuel cells
Charged Particles in Fluid Flows

- LBM for Hydrodynamics
- Rigid Multibody Dynamics for particulate phase
- Multigrid (FV discretization) for electrostatic effects
- Ion transport and double layer effects still neglected
Electrostatic Potential and Force on Particles

- Electric potential described by Poisson equation, with particle’s charge density on RHS:

\[ -\Delta \Phi(\vec{x}) = \frac{\rho_{\text{particles}}(\vec{x})}{\epsilon_r \epsilon_0} \]

- Discretized by finite volumes.
- Solved with cell-centered multigrid solver.
- Subsampling for computing overlap degree to set RHS accordingly.

\[ \vec{F}_q = -q_{\text{particle}} \cdot \nabla \Phi(\vec{x}) \]
Agglomeration of charged particles on charged plane in fluid flow.

Channel: 2.56 x 5.76 x 2.56 mm
Dx=10µm, Dt=4·10^{-5}s, τ=1.7

Particle radius: 60µm
Particle charge: 8000e
Inflow velocity: 1mm/s
Other walls: No-slip BCs

Potential: Bottom -100V, Top 0V
Other walls: homogen. Neumann BCs

Computed on 144 cores (12 nodes) of RRZE - LiMa
71.600 time steps
64^3 unknowns per core
6 MG levels