Parallel Solution of a Finite Element Problem with 17 Billion Unknowns

B. Bergen (LSS Erlangen/Los Alamos)
T. Gradl (LSS Erlangen)
F. Hülsemann (LSS Erlangen/EDF)
U. Rüde (LSS Erlangen, ruede@cs.fau.de)

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

Boston, July 2006
Overview

**Motivation**

**Applications of Large-Scale FE Simulations**
- Direct and Inverse Bio-Electric Field Problems
- 3D image registration/ image processing/ analysis
- Flow Induced Noise and Acoustics
- Multiphysics-Problems in Nano Technology
- Simulation of High Temperature Furnaces

**Scalable Finite Element Solvers**
- **Scalable Algorithms:** Multigrid
- **Scalable Architecture:** Hierarchical Hybrid Grids (HHG)
- Results

**Outlook**
- Towards Peta-Scale FE Solvers
Part I
Motivation
Architecture Example
Our Pet Dinosaur (*now in retirement*)

- 8 Proc and 8 GB per node
  - 1344 CPUs (168*8)
  - 2016 Gflop total
  - Very sensitive to data structures
- Currently being replaced by HLRB-II:
  - 4096 Processor SGI Altix
  - upgrade to >60 Tflop in 2007
- KONWIHR = Competence Network for Scientific Supercomputing
  - Project Gridlib (2001-2005)
  - Development of Hierarchical Hybrid Grids (HHG)

HLRB-I: Hitachi SR 8000 at the Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (No. 5 at time of installation in 2000)
gridlib/HHG MFlops rates for matrix-vector multiplication on one node on the Hitachi

compared with *highly tuned* JDS results for sparse matrices
HHG Motivation II: DiMe - Project

Data Local Iterative Methods for the Efficient Solution of Partial Differential Equations

www10.informatik.uni-erlangen.de/de/Research/Projects/DiME/

- Cache-optimizations for sparse matrix codes
- High Performance Multigrid Solvers
- Efficient LBM codes for CFD
Memory for the discretization is often the limiting factor on the size of problem that can be solved.

**Common Practice**
- Add resolution by applying regular refinement to an unstructured input grid
- Refinement does not add new information about the domain ⇒ CRS is overkill (or any other sparse matrix structure)

**Missed Opportunity**
- Regularity of structured patches is not exploited

**HHG**
- Develop new data structures that **exploit regularity** for enhanced performance
- Employ **stencil-based discretization** techniques on structured patches to reduce memory usage
Part II

Applications of Large Scale Finite Element Simulations
Example: Flow Induced Noise

Flow around a small square obstacle

Relatively simple geometries, but fine resolution for resolving physics

Images by M. Escobar (Dept. of Sensor Technology + LSTM)
Large Scale Acoustic Simulations

- A concert hall may have $>10\,000\,m^3$ volume
  - need a resolution of $<1\text{cm}$ to resolve audible spectrum of wavelengths
  - need $>10^6$ cells per $1\,m^3$
  - The concert hall will require to solve a system with $>10^{10}$ unknowns per (implicit) time step

- Flow induced noise (KONWIHR)
  - generated by turbulence and boundary effects
  - high frequency noise requires a very fine resolution of acoustic field
  - far field computations $\Rightarrow$ large domains
Part III - a

Towards Scalable FE Software

*Multigrid Algorithms*
What is Multigrid?

- Has nothing to do with „grid computing“
- A general methodology
  - multi - scale (actually it is the „original“)
  - many different applications
  - developped in the 1970s - ...
- Useful e.g. for solving elliptic PDEs
  - large sparse systems of equations
  - iterative
  - convergence rate independent of problem size
  - asymptotically optimal complexity -> algorithmic scalability!
  - can solve e.g. 2D Poisson Problem in ~ 30 operations per gridpoint
  - efficient parallelization possible
  - best (maybe the only?) basis for fully scalable FE solvers
Goal: solve $A^h u^h = f^h$ using a hierarchy of grids

Relax on $A^h u^h = f^h$

Residual $r^h = f^h - A^h u^h$

Restrict $r^H = I^H_h r^h$

Correct $u^h \leftarrow u^h + e^h$

Interpolate $e^h = I^H_h e^H$

Solve $A^H e^H = r^H$

by recursion
Multigrid Details

- Finite Element induced
  - interpolation
  - restriction
  - Galerkin coarsening

- V(2,2) cycle

- line-wise Gauss-Seidel smoother
  - lines in red-black ordering
  - neglecting a few dependencies across proc. boundaries (Jacobi-like points)

- mostly standard
Parallel High Performance FE Multigrid

- Parallelize "plain vanilla" multigrid
  - partition domain
  - parallelize all operations on all grids
  - use clever data structures

- Do not worry (so much) about coarse grids
  - idle processors?
  - short messages?
  - sequential dependency in grid hierarchy?

- Why we do not use Domain Decomposition
  - DD without coarse grid does not scale algorithmically and is inefficient for large problems/ many processors
  - DD with coarse grid is still less efficient than multigrid and is as difficult to parallelize
Part III - b

Towards Scalable FE Software

Hierarchical Hybrid Grids
Hierarchical Hybrid Grids (HHG)

- Unstructured input grid
  - resolves geometry of problem domain
- Patch-wise regular refinement
  - generates nested grid hierarchies naturally suitable for geometric multigrid algorithms
- New:
  - Modify storage formats and operations on the grid to exploit the regular substructures
- Does an unstructured grid with 100 000 000 000 elements make sense?

HHG Goal: Ultimate Parallel FE Performance!
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
Common HHG Misconceptions

- Hierarchical hybrid grids (HHG)
  - are not only another block structured grid
- HHG are more flexible (unstructured, hybrid input grids)
  - are not only another unstructured geometric multigrid package
- HHG achieve better performance
  - unstructured treatment of regular regions does not improve performance
Parallel HHG - Framework Design

Goals

To realize good parallel scalability:

- Minimize latency by reducing the number of messages that must be sent
- Optimize for high bandwidth interconnects ⇒ large messages
- Avoid local copying into MPI buffers
HHG for Parallelization

- Use regular HHG patches for partitioning the domain
**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
HHG for Parallelization

SEND BUFFER

RECEIVE BUFFER

LOCAL BUFFER UNUSED

SEND BUFFER

LOCAL BUFFER UNUSED

Partition 0

Partition 1
Part III - c

Towards Scalable FE Software

*Performance Results*
**Node Performance is Difficult!**  (B. Gropp)

DiMe project: Cache-aware Multigrid (1996- ...)

<table>
<thead>
<tr>
<th>Grid Size</th>
<th>17³</th>
<th>33³</th>
<th>65³</th>
<th>129³</th>
<th>257³</th>
<th>513³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>107</td>
<td>134</td>
<td>715</td>
<td>677</td>
<td>490</td>
<td>579</td>
</tr>
<tr>
<td>No Blocking</td>
<td>244</td>
<td>441</td>
<td>995</td>
<td>1065</td>
<td>849</td>
<td>819</td>
</tr>
<tr>
<td>2x Blocking</td>
<td>540</td>
<td>791</td>
<td>1312</td>
<td>1319</td>
<td>1284</td>
<td>1282</td>
</tr>
<tr>
<td>3x Blocking</td>
<td>042</td>
<td>238</td>
<td>2167</td>
<td>2140</td>
<td>2134</td>
<td>2049</td>
</tr>
</tbody>
</table>

- Performance of 3D-MG-Smoother for 7-pt stencil in Mflops on Itanium 1.4 GHz
- Array Padding
- Temporal blocking - in EPIC assembly language
- Software pipelineing in the extreme (M. Stürmer - J. Treibig)

**Node Performance is Possible!**
Single Processor HHG Performance on Itanium for Relaxation of a Tetrahedral Finite Element Mesh

![Graph showing MFLOP/s vs Refinement Level for Itanium 2, 1.6 GHz and Nocona, 3.4 GHz]
Parallel HHG Performance on Altix for Multigrid with a Tetrahedral Finite Element Mesh
HHG: Parallel Scalability

<table>
<thead>
<tr>
<th>#Procs</th>
<th>#DOFS x $10^6$</th>
<th>#Els x $10^6$</th>
<th>#Input Els</th>
<th>GFLOP/s</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>2,144</td>
<td>12,884</td>
<td>6144</td>
<td>100/75</td>
<td>68</td>
</tr>
<tr>
<td>128</td>
<td>4,288</td>
<td>25,769</td>
<td>12288</td>
<td>200/147</td>
<td>69</td>
</tr>
<tr>
<td>256</td>
<td>8,577</td>
<td>51,539</td>
<td>24576</td>
<td>409/270</td>
<td>76</td>
</tr>
<tr>
<td>512</td>
<td>17,167</td>
<td>103,079</td>
<td>49152</td>
<td>762/545</td>
<td>75</td>
</tr>
<tr>
<td>1024</td>
<td>17,167</td>
<td>103,079</td>
<td>49152</td>
<td>1,456/964</td>
<td>43</td>
</tr>
</tbody>
</table>

Parallel scalability of Poisson problem discretized by tetrahedral finite elements - SGI Altix (Itanium-2 1.6 GHz)

See also: ISC Award 2006.
Why Multigrid?

- A direct elimination banded solver has complexity $O(N^{2.3})$ for a 3-D problem.
- This becomes
  
  $$338869529114764631553758 = O(10^{23})$$

  operations for our problem size

- At one Teraflops this would result in a runtime of 10000 years which could be reduced to 10 years on a Petaflops system
Part IV

Outlook
System Configuration
HLRB-II (Phase I)

- 4096 Intel Madison9M (1.6 GHz) cores
  - single core = 4096 cores
- 6 MByte L3
- FSB 533 (8.5 GByte/s for one core)
- 1.33 Byte/Flop = 0.17 Words/Flop
- 17 TByte memory
- 26.2 TF Peak -> 24.5 TF Linpack
- ~ 7.4 TF aggreg. weighted Application Performance (LRZ BM)
- ~ 3.5 TF expected as every-day mean performance
- 256 core single system image
Target: 3328+ Intel Montvale Sockets
  - dual core = 6656+ cores
  - 9 MByte L3

FSB 667 (10.7 GB/s shared by two cores)
~0.1 Words/Flop

40+ TByte memory
>60 TF Peak

13 TF LRZ benchmarks
~ 6 TF expected sustained
512 cores per node

Installation: 2007
Conclusions

Supercomputer Performance is 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: disguise HitMe by within “clever data structures” that introduce a lot of overhead

- 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 (such as multigrid)
  - the more primitive the algorithm the easier to maximize your MachoFlops.
Acknowledgements

Collaborators
- In Erlangen: WTM, LSE, LSTM, LGDV, RRZE, Neurozentrum, Radiologie, etc.
- International: Utah, Technion, Constanta, Ghent, Boulder, ...

Dissertationen Projects
- B. Bergen (HHG development)
- T. Gradl (HHG application)
- J. Treibig (Cache Optimizations)
- M. Stürmer (Architecture Aware Programming)
- U. Fabricius (AMG-Methods and SW-Engineering for parallelization)
- C. Freundl (Parallel Expression Templates for PDE-solver)
- J. Härtlein (Expression Templates for FE-Applications)
- N. Thürey (LBM, free surfaces)
- T. Pohl (Parallel LBM)
- ... and 6 more

20 Diplom- /Master- Thesis

Studien- /Bachelor- Thesis
- Especially for Performance-Analysis/ Optimization for LBM
  - J. Wilke, K. Iglberger, S. Donath
- ... and 23 more

Funding: KONWIHR, DFG, NATO, BMBF

CSE (Graduate) Education: Elitenetzwerk Bayern
- Bavarian Graduate School in Computational Engineering (with TUM, since 2004)
- Special International PhD program: Identifikation, Optimierung und Steuerung für technische Anwendungen (with Bayreuth and Würzburg) since Jan. 2006
Talk is Over

Please wake up!
FE Applications *(in our group)*

- **Solid-state lasers**
  - heat conduction, elasticity, wave equation

- **Bio-electric fields**
  - Calculation of electrostatic or electromagnetic potential in heterogeneous domains

- **Computational Nano Technology**
  - Lattice-Boltzmann simulation of charged particles in colloids
  - Calculation of electrostatic potential in very large domains

- **Acoustics**
  - room acoustics
  - flow induced noise

- **Industrial High Temperature Furnaces**
  - heat transport, radiation, ...
Memory used to represent the discretization is often the limiting factor on the size of problem that can be solved

- **Common Practice**
  - Add resolution by applying regular refinement to an unstructured input grid
  - Refinement does not add new information about the domain $\Rightarrow$ **CRS is overkill** (or any other sparse matrix structure)

- **Missed Opportunity**
  - Regularity of structured patches is not exploited

- **HHG**
  - Develop new data structures that **exploit regularity** for enhanced performance
  - Employ **stencil-based discretization** techniques on structured patches to reduce memory usage
Large Scale Acoustic Simulations

- A concert hall may have >10 000 m³ volume
  - need a resolution of <1cm to resolve audible spectrum of wavelengths
  - need >10⁶ cells per 1 m³
  - The concert hall will require to solve a system with >10¹⁰ unknowns per (implicit) time step

- Flow induced noise (KONWIHR)
  - generated by turbulence and boundary effects
  - high frequency noise requires a very fine resolution of acoustic field
  - far field computations => large domains

HHG will make unprecedented acoustic simulations possible!
Standard sparse matrix data structures (like CRS) generally achieve poor performance!

- **Cache Effects**
  - Difficult to find an ordering of the unknowns that maximizes data locality

- **Indirect Indexing**
  - Precludes aggressive compiler optimizations that exploit instruction level parallelism (ILP)

- **Continuously Variable Coefficients**
  - Overkill for certain class of problems
User-Friendliness: ParExPDE
Parallel Expression Templates for
Partial Differential Equations

- Library for the rapid development of numerical PDE solvers on parallel architectures
- High level intuitive programming interface
- Use of expression templates ensures good efficiency
- Funded by KONWIHR (2001-2004)

User-friendly FE-Application development with excellent parallel efficiency
Conclusions

- High performance simulation still requires “heroic programming”
- Parallel Programming: single node performance is difficult
- Which architecture?
- Which data structures?
- Where are we going?
  - the end of Moore’s law
  - petaflops require >100,000 processors - and we can hardly handle 1000!
  - the memory wall
    - latency
    - bandwidth
    - Locality!
- We are open for collaborations!
Conclusions (1)

- High performance simulation still requires “heroic programming” … but we are on the way to make supercomputers more generally usable

- Parallel Programming is easy, node performance is difficult (B. Gropp)

- Which architecture ?
  - ASCI-type: custom CPU, massively parallel cluster of SMPs
    - nobody has been able to show that these machines scale efficiently, except on a few very special applications and using enormous human effort
  - Earth-simulator-type: Vector CPU, as many CPUs as affordable
    - impressive performance on vectorizable code, but need to check with more demanding data and algorithm structures
  - Hitachi Class: modified custom CPU, cluster of SMPs
    - excellent performance on some codes, but unexpected slowdowns on others, too exotic to have a sufficiently large software base
  - Others: BlueGene, Cray X1, Multithreading, PIM, reconfigurable …, quantum computing, …
Conclusions (2)

Which data structures?
- structured (inflexible)
- unstructured (slow)
- HHG (high development effort, even prototype 50 K lines of code)
- meshless … (useful in niches)

Where are we going?
- the end of Moore’s law
- nobody builds CPUs with HPC specific requirements high on the list of priorities
- petaflops: 100,000 processors and we can hardly handle 1000
- It’s the locality - stupid!
- the memory wall
  - latency
  - bandwidth
- Distinguish between algorithms where control flow is
  - data independent: latency hiding techniques (pipelining, prefetching, etc) can help
  - data dependent
Talk is Over

Please wake up!
In the Future?

What’s beyond Moore’s Law?
Conclusions (1)

High performance simulation still requires “heroic programming” … but we are on the way to make supercomputers more generally usable

Parallel Programming is easy, node performance is difficult (B. Gropp)

Which architecture?

- Large Scale Cluster: custom CPU, massively parallel cluster of SMPs
  - nobody has been able to show that these machines scale efficiently, except on a few very special applications and using enormous human effort

- Earth-simulator-type: Vector CPU, as many CPUs as affordable
  - impressive performance on vectorizable code, but need to check with more demanding data and algorithm structures

- Hitachi Class: modified custom CPU, cluster of SMPs
  - excellent performance on some codes, but unexpected slowdowns on others, too exotic to have a sufficiently large software base

- Others: BlueGene, Cray X1, Multithreading, PIM, reconfigurable …, quantum computing, …
Conclusions (2)

- Which data structures?
  - structured (inflexible)
  - unstructured (slow)
  - HHG (high development effort, even prototype 50 K lines of code)
  - meshless … (useful in niches)

- Where are we going?
  - the end of Moore’s law
  - nobody builds CPUs for HPC-specific requirements
  - petaflops require >100,000 processors - and we can hardly handle 1000!
  - It’s the locality - stupid!
  - the memory wall
    - latency
    - bandwidth
  - Distinguish between algorithms where control flow is
    - data independent: latency hiding techniques (pipelining, prefetching, etc) can help
    - data dependent
Moore's Law in Semiconductor Technology
(F. Hossfeld)

Growth: 52% per year
Growth: 42% per year
Information Density & Energy Dissipation
(adapted by F. Hossfeld from C. P. Williams et al., 1998)

Semiconductor Technology

Energy/logic Operation [pico-Joules]

Year

Atom

s/Bit

Information Density & Energy Dissipation

(adopted by F. Hossfeld from C. P. Williams et al., 1998)
Acknowledgements

Collaborators
- In Erlangen: WTM, LSE, LSTM, LGDV, RRZE, etc.
- Especially for foams: C. Körner (WTM)
- International: Utah, Technion, Constanta, Ghent, Boulder, ...

Dissertationen Projects
- U. Fabricius (AMG-Verfahren and SW-Engineering for parallelization)
- C. Freundl (Parelle Expression Templates for PDE-solver)
- J. Härtlein (Expression Templates for FE-Applications)
- N. Thürey (LBM, free surfaces)
- T. Pohl (Parallel LBM)
- ... and 6 more

16 Diplom- /Master- Thesis

Studien- /Bachelor- Thesis
- Especially for Performance-Analysis/ Optimization for LBM
  - J. Wilke, K. Iglberger, S. Donath
- ... and 21 more

KONWIHR, DFG, NATO, BMBF

Elitenetzwerk Bayern
- Bavarian Graduate School in Computational Engineering (with TUM, since 2004)
- Special International PhD program: Identifikation, Optimierung und Steuerung für technische Anwendungen (with Bayreuth and Würzburg) to start Jan. 2006.
ParExPDE - Performance Results

*Speedup* of ParExPDE on the LSS Cluster

<table>
<thead>
<tr>
<th># Proc.</th>
<th>Number of Unknowns</th>
<th>Wallclock Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$8.51 \cdot 10^6$</td>
<td>291</td>
</tr>
<tr>
<td>2</td>
<td>$8.51 \cdot 10^6$</td>
<td>138</td>
</tr>
<tr>
<td>4</td>
<td>$8.51 \cdot 10^6$</td>
<td>71</td>
</tr>
<tr>
<td>8</td>
<td>$8.51 \cdot 10^6$</td>
<td>39</td>
</tr>
<tr>
<td>16</td>
<td>$8.51 \cdot 10^6$</td>
<td>24</td>
</tr>
<tr>
<td>32</td>
<td>$8.51 \cdot 10^6$</td>
<td></td>
</tr>
</tbody>
</table>

- Multigrid solver ($V(2,2)$ cycles) for Poisson problem with Dirichlet boundary
- LSS Cluster: AMD Opteron 2.2 GHz
**ParExPDE - Performance Results**

*Scaleup* of ParExPDE on the LSS Cluster

<table>
<thead>
<tr>
<th># Proc.</th>
<th>Number of Unknowns</th>
<th>Wallclock Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2.00 \cdot 10^6$</td>
<td>67</td>
</tr>
<tr>
<td>2</td>
<td>$4.02 \cdot 10^6$</td>
<td>68</td>
</tr>
<tr>
<td>4</td>
<td>$8.05 \cdot 10^6$</td>
<td>68</td>
</tr>
<tr>
<td>8</td>
<td>$1.61 \cdot 10^7$</td>
<td>72</td>
</tr>
<tr>
<td>16</td>
<td>$3.22 \cdot 10^7$</td>
<td>72</td>
</tr>
<tr>
<td>32</td>
<td>$6.45 \cdot 10^7$</td>
<td>76</td>
</tr>
</tbody>
</table>

- Multigrid solver ($V(2,2)$ cycles) for Poisson problem with Dirichlet boundary
- LSS Cluster: AMD Opteron 2.2 GHz
Source Localization

3D MRI data
512 x 512 x 256 voxels
segmentation
4 compartments

Localized
Epileptic focus

Dipole localization search algorithm = optimization

Collaborators: Univ. of Utah (Chris Johnson), Ovidius Univ. Constanτa (C. Popa) Bart Vanrumste (Gent, Univ. of Canterbury, New Zealand), G. Greiner, F. Fahlbusch, C. Wolters (Münster)
Source Localization

Localized
Epileptic focus

Dipole localization
search algorithm
= optimization

Collaborators: Univ. of Utah (Chris Johnson), Ovidius Univ. Constanta (C. Popa) Bart Vanrumste (Gent, Univ. of Canterbury, New Zealand), G. Greiner, F. Fahlbusch, C. Wolters (Münster)
Data Requirements

3D MRI data
512 x 512 x 256 voxels
segmentation
4 compartments

Data types to describe isotropic and anisotropic conductivity
HHG Parallel Update Algorithm

for each vertex do
   apply operation to vertex
end for

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

for each element do
   copy from edge/vertex interiors
   apply operation to element
   copy to edge/vertex halos
end for

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

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

for each element do
   copy from edge/vertex interiors
   apply operation to element
   copy to edge/vertex halos
end for

update secondary dependencies
Numerical Models for Source Localization

- Computational expensive treatment of singularities
  - Dipole Models (Blurred, Subtraction, Mathematical/ Zenger Correction)
  - Resolution and complexity of the mesh is important
### Speedup of ParExPDE

<table>
<thead>
<tr>
<th># Proc.</th>
<th>Number of Unknowns</th>
<th>Wallclock Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$8.51 \cdot 10^6$</td>
<td>291</td>
</tr>
<tr>
<td>2</td>
<td>$8.51 \cdot 10^6$</td>
<td>138</td>
</tr>
<tr>
<td>4</td>
<td>$8.51 \cdot 10^6$</td>
<td>71</td>
</tr>
<tr>
<td>8</td>
<td>$8.51 \cdot 10^6$</td>
<td>39</td>
</tr>
<tr>
<td>16</td>
<td>$8.51 \cdot 10^6$</td>
<td>24</td>
</tr>
<tr>
<td>32</td>
<td>$8.51 \cdot 10^6$</td>
<td>14</td>
</tr>
</tbody>
</table>

- Multigrid solver
- $V(2,2)$ cycles for Poisson problem
- Dirichlet boundary

**LSS-Cluster**

**Compute Nodes**
(8x4 CPUs)
CPU: AMD Opteron 848
ParExPDE - Performance Results

**Scaleup** of ParExPDE on the LSS Cluster

<table>
<thead>
<tr>
<th># Proc.</th>
<th>Number of Unknowns</th>
<th>Wallclock Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2.00 \cdot 10^6$</td>
<td>67</td>
</tr>
<tr>
<td>2</td>
<td>$4.02 \cdot 10^6$</td>
<td>68</td>
</tr>
<tr>
<td>4</td>
<td>$8.05 \cdot 10^6$</td>
<td>68</td>
</tr>
<tr>
<td>8</td>
<td>$1.61 \cdot 10^7$</td>
<td>72</td>
</tr>
<tr>
<td>16</td>
<td>$3.22 \cdot 10^7$</td>
<td>72</td>
</tr>
<tr>
<td>32</td>
<td>$6.45 \cdot 10^7$</td>
<td>76</td>
</tr>
</tbody>
</table>

- Multigrid solver
- V(2,2) cycles
- Poisson problem
- Dirichlet boundary
- LSS Cluster: AMD Opteron 2.2 GHz
ParExPDE - Applications

- Simulation of solid-state lasers (Prof. Pflaum)
  - Calculation of heat conduction
  - Calculation of elasticity
  - Solution of Helmholtz equation
Bio-electric Field Computations

Reconstruction of electromagnetic fields from EEG-Measurements: Source Localization

Neurosurgery Kopfklinikum Erlangen

View through operation microscope
Why simulate and not experiment?

Open brain
EEG Measurements for
Localizing
functional brain regions

Simulation based
Virtual operation planning

LEHRSTUHL FÜR INFORMATIK 10 (SYSTEMSIMULATION)
Problem of inverse EEG/MEG

- **Direct Problem:**
  - *Known:* Sources (strength, position, orientation)
  - *Wanted:* Potentials on the head surface

- **Inverse Problem**
  - *Known:* Potentials on the head surface
  - *Wanted:* Sources (strength, position, orientation)

\[
\Phi = \Phi_0 \quad \text{on } \Sigma \subseteq \Gamma_T
\]

\[
\sigma \nabla \Phi \cdot n = 0 \quad \text{on } \Gamma_T
\]

\[
\nabla \cdot \sigma \nabla \Phi = -I_V
\]
Computational Requirements
for Biomedical FE Computations

- Resolution: $1024^3 \approx 10^9$ Voxels
- Complex source models, anisotropic conductivity, inverse problem
- Bio-electro-chemical model for signal propagation
- Coupling to bio-mechanical effects, e.g.
  - contraction of heart muscle
  - blood flow
- Therapy planning, e.g. cardiac resynchronization pacemaker
  - optimal placement of electrodes
  - optimal control of resynchronization
- ParExPDE as scalable parallel solver!
Computational Nano Technology

- Extension of the LBM particle simulation by an electrostatic potential
- Long range particle-particle interactions
- Simulating charged nanoparticles in flow field
- Generates ions in the fluid (three species of fluid: +, -, 0)
- Generates charge distribution
- ParExPDE for computing the (static) potential distribution in each time step
  - $10^9 - 10^{15}$ unknowns
- Compute forces on particles and ionized fluid from potential

Diplomarbeit C. Feichtinger
3. ORCAN

Component based application framework to ease the development of parallel simulations

Central issues are

- Flexibility: Dynamically switch modules
- Maintainability: Enable a longterm usage of a code
- Extensibility: Enable reuse of existing codes and coupling to other solvers (multiphysics)
- Enable a well-organized SW-development process

Modern Software Infrastructure for Technical High Performance Computing