GPU Cluster Computing for FEM

Dominik Göddeke
Sven H.M. Buijssen, Hilmar Wobker and Stefan Turek

Angewandte Mathematik und Numerik
TU Dortmund, Germany
dominik.goeddeke@math.tu-dortmund.de

GPU Computing in Computational Science and Engineering
International Workshop on Computational Engineering
Special Topic Fluid-Structure Interaction
Herrsching, October 14, 2009
FEAST – Hardware-oriented Numerics
Structured vs. unstructured grids

**Fully adaptive grids**
- Maximum flexibility
- ‘Stochastic’ numbering
- Unstructured sparse matrices
- Indirect addressing, very slow.

**Locally structured grids**
- Logical tensor product
- Fixed banded matrix structure
- Direct addressing (⇒ fast)
- $r$-adaptivity

Unstructured macro mesh of tensor product subdomains

Exploit local structure for tuned linear algebra and tailored multigrid smoothers
Solver approach

**ScaRC – Scalable Recursive Clustering**

- Hybrid multilevel domain decomposition method
- Minimal overlap by extended Dirichlet BCs
- Inspired by parallel MG ("best of both worlds")
  - Multiplicative between levels, global coarse grid problem (MG-like)
  - Additive horizontally: block-Jacobi / Schwarz smoother (DD-like)
- Hide local irregularities by MGs within the Schwarz smoother
- Embed in Krylov to alleviate Block-Jacobi character

```
global BiCGStab  
preconditioned by  
global multilevel (V 1+1)  
additively smoothed by  
  for all Ωᵢ: local multigrid  
coarse grid solver: UMFPACK
```
Multivariate problems

Block-structured systems

- Guiding idea: Tune scalar case once per architecture instead of over and over again per application
- Equation-wise ordering of the unknowns
- Block-wise treatment enables multivariate ScaRC solvers

Examples

- Linearised elasticity with compressible material (2x2 blocks)
- Saddle point problems: Stokes (3x3 with zero blocks), elasticity with (nearly) incompressible material, Navier-Stokes with stabilisation (3x3 blocks)

\[
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2
\end{pmatrix}
= f,
\begin{pmatrix}
A_{11} & 0 & B_1 \\
0 & A_{22} & B_2 \\
B_1^T & B_2^T & 0
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
p
\end{pmatrix}
= f,
\begin{pmatrix}
A_{11} & A_{12} & B_1 \\
A_{21} & A_{22} & B_2 \\
B_1^T & B_2^T & C_C
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
p
\end{pmatrix}
= f
\]

\(A_{11}\) and \(A_{22}\) correspond to scalar (elliptic) operators
⇒ Tuned linear algebra \textbf{and} tuned solvers
Co-processor integration into FEAST
Bandwidth in a CPU/GPU node

- CPU:
  - Processing elements: 40 GB/s
  - Cache

- Co-processor

- System memory: 6-15 GB/s

- Device memory: 20-160 GB/s

- Infiniband to next node: 1-2 GB/s

- 1-8 GB/s
## Mixed Precision Multigrid

<table>
<thead>
<tr>
<th>Level</th>
<th>Core2Duo (double)</th>
<th>GTX 280 (mixed)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time(s)</td>
<td>MFLOP/s</td>
</tr>
<tr>
<td>7</td>
<td>0.021</td>
<td>1405</td>
</tr>
<tr>
<td>8</td>
<td>0.094</td>
<td>1114</td>
</tr>
<tr>
<td>9</td>
<td>0.453</td>
<td>886</td>
</tr>
<tr>
<td>10</td>
<td>1.962</td>
<td>805</td>
</tr>
</tbody>
</table>

- Poisson on unitsquare, Dirichlet BCs, TP grid, *not a matrix stencil*
- *Converges to wrong solution in single precision*
- 1M DOF, multigrid, FE-accurate in less than 0.1 seconds!
- 27x faster than CPU, exactly same results as pure double
- 1.7x faster than pure double on GPU
- defect calculation alone: 46.5 GFLOP/s, 50x speedup (single vs. single)
Minimally invasive integration

**global BiCGStab**
preconditioned by
**global multilevel** (V 1+1)
additively smoothed by
**for all** $\Omega_i$: **local multigrid**
coarse grid solver: UMFPACK

All outer work: CPU, double
Local MGs: GPU, single
Same accuracy and functionality mandatory
Oblivious of the application
Some results
Linearised elasticity

\[
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2
\end{pmatrix}
= f
\]

\[
\begin{pmatrix}
(2\mu + \lambda)\partial_{xx} + \mu\partial_{yy} & (\mu + \lambda)\partial_{xy} \\
(\mu + \lambda)\partial_{yx} & \mu\partial_{xx} + (2\mu + \lambda)\partial_{yy}
\end{pmatrix}
\]

global multivariate BiCGStab  
block-preconditioned by  
Global multivariate multilevel (V 1+1)  
additively smoothed (block GS) by  
for all \(\Omega_i\): solve \(A_{11}c_1 = d_1\) by  
local scalar multigrid  
update RHS: \(d_2 = d_2 - A_{21}c_1\)  
for all \(\Omega_i\): solve \(A_{22}c_2 = d_2\) by  
local scalar multigrid  
coarse grid solver: UMFPACK
Accuracy

Cantilever beam, aniso 1:1, 1:4, 1:16
Hard, very ill-conditioned CSM test
CG solver: > 2x iterations per refinement
GPU-ScaRC solver: same results as CPU

<table>
<thead>
<tr>
<th>refinement</th>
<th>Iterations</th>
<th>Volume</th>
<th>y-Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>aniso04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L=8</td>
<td>CPU</td>
<td>GPU</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>1.6087641E-3</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>4</td>
<td>1.6087641E-3</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4.5</td>
<td>1.6087641E-3</td>
</tr>
<tr>
<td>aniso16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L=8</td>
<td>6</td>
<td>6</td>
<td>6.7176398E-3</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>6</td>
<td>6.7176427E-3</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5.5</td>
<td>6.7176516E-3</td>
</tr>
</tbody>
</table>
Weak scalability

- Outdated cluster, dual Xeon EM64T singlecore
- one NVIDIA Quadro FX 1400 per node (one generation behind the Xeons, 20 GB/s BW)
- Poisson problem (left): up to 1.3 B DOF, 160 nodes
- Elasticity (right): up to 1 B DOF, 128 nodes
Absolute speedup

![Graph showing speedup comparison between CPU and GPU for different tasks.

- **16 nodes, Opteron 2214 dualcore**
- **NVIDIA Quadro FX 5600 (76 GB/s BW), OpenGL**
- **Problem size 128 M DOF**
- **Dualcore 1.6x faster than singlecore**
- **GPU 2.6x faster than singlecore, 1.6x than dual**
Acceleration analysis

**Speedup analysis**

- Addition of GPUs increases resources
- ⇒ Correct model: strong scalability inside each node
- Accelerable fraction of the elasticity solver: 2/3
- Remaining time spent in MPI and the outer solver

**Accelerable fraction** $R_{acc}: \ 66\%$

**Local speedup** $S_{local}: \ 9x$

**Total speedup** $S_{total}: \ 2.6x$

**Theoretical limit** $S_{max}: \ 3x$
Stationary Navier-Stokes

\[
\begin{pmatrix}
A_{11} & A_{12} & B_1 \\
A_{21} & A_{22} & B_2 \\
B_1^T & B_2^T & C
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
p
\end{pmatrix}
= \begin{pmatrix}
f_1 \\
f_2 \\
g
\end{pmatrix}
\]

- 4-node cluster
- Opteron 2214 dualcore
- GeForce 8800 GTX (90 GB/s BW), CUDA
- Driven cavity and channel flow around a cylinder

**fixed point iteration**
assemble linearised subproblems and solve with **global BiCGStab** (reduce initial residual by 1 digit)
Block-Schurcomplement preconditioner
1) approx. solve for velocities with **global MG** ($V_{1+0}$), additively smoothed by
   for all $\Omega_i$: solve for $u_1$ with **local MG**
2) update RHS: $d_3 = -d_3 + B^T(c_1, c_2)^T$
3) scale $c_3 = (M_p^L)^{-1}d_3$
Navier-Stokes results

### Speedup analysis

<table>
<thead>
<tr>
<th></th>
<th>$R_{\text{acc}}$</th>
<th>$S_{\text{local}}$</th>
<th>$S_{\text{total}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L9</td>
<td>L10</td>
<td>L9</td>
</tr>
<tr>
<td>DC Re100</td>
<td>41%</td>
<td>46%</td>
<td>6x</td>
</tr>
<tr>
<td>DC Re250</td>
<td>56%</td>
<td>58%</td>
<td>5.5x</td>
</tr>
<tr>
<td>Channel flow</td>
<td>60%</td>
<td>–</td>
<td>6x</td>
</tr>
</tbody>
</table>

**Important consequence:** Ratio between assembly and linear solve changes significantly

<table>
<thead>
<tr>
<th></th>
<th>DC Re100</th>
<th>DC Re250</th>
<th>Channel flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain accel.</td>
<td>29:71</td>
<td>50:48</td>
<td>11:89</td>
</tr>
<tr>
<td>plain accel.</td>
<td>25:75</td>
<td>13:87</td>
<td>26:74</td>
</tr>
</tbody>
</table>
Conclusions
Conclusions

- Hardware-oriented numerics prevents existing codes being worthless in a few years
- Mixed precision schemes exploit the available bandwidth without sacrificing accuracy
- GPUs as local preconditioners in a large-scale parallel FEM package
- Not limited to GPUs, applicable to all kinds of hardware accelerators
- Minimally invasive approach, no changes to application code
- Excellent local acceleration, global acceleration limited by ‘sequential’ part
- Future work: Design solver schemes with higher acceleration potential without sacrificing numerical efficiency
Acknowledgements

Collaborative work with

FEAST group (TU Dortmund)

Robert Strzodka (Max Planck Institut Informatik)

Jamaludin Mohd-Yusof, Patrick McCormick (Los Alamos National Laboratory)

http://www.mathematik.tu-dortmund.de/~goeddeke

Supported by DFG, projects TU 102/22-1, TU 102/22-2, TU 102/27-1, TU102/11-3; and BMBF, *HPC Software für skalierbare Parallelrechner*: SKALB project 01IH08003D