GPU Cluster Computing for FEM with Applications in CFD and CSM

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

Institut für Angewandte Mathematik
TU Dortmund, Germany
dominik.goeddeke@math.tu-dortmund.de

Mini-Symposium: GPU Computing in CFD
ECCOMAS-CFD 2010
Lisbon, Portugal, June 17, 2010

FEAST –

Hardware-oriented Numerics

FEAST – hardware-oriented numerics

Hardware-oriented numerics

- Much more than good implementation of good numerics
- Balancing of (often) contradictory efficiency requirements
  - Numerical efficiency (convergence rates)
  - Hardware efficiency (MFLOP/s rates)
  - Parallel efficiency (scalability)
- Goal: Maximise *total efficiency*

FEAST – Finite Element Analysis and Solution Tools

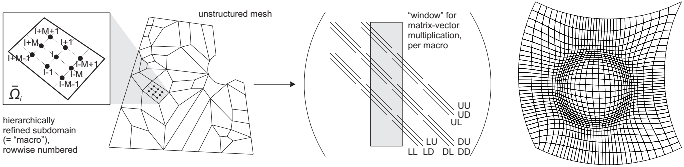
- Toolbox and infrastructure for large-scale finite element discretisations *and* parallel multilevel solvers
- Applications are built on top of FEAST
- Not maximum performance for one particular application, but high and scalable performance for many problems

Discretisation approach

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 ($\Rightarrow$ fast)
r-adaptivity

Unstructured macro mesh of generalised tensor product patches



Exploit local structure for tuned linear algebra and tailored multigrid components

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)
- Schwarz smoother encapsulates local irregularities: Robust multigrid ('gain a digit') with strong smoothers
- Embed in Krylov to alleviate Block-Jacobi character

global BiCGStab
preconditioned by
  global multilevel (V 1+1)
  additively smoothed by
    for all Ω_i: **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

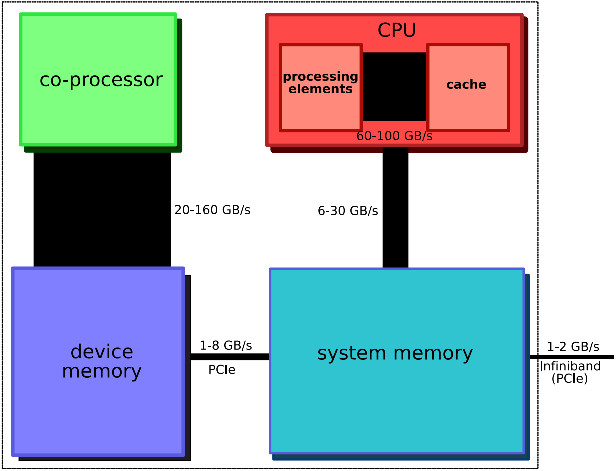
- Linearised elasticity with compressible material (2x2 blocks)
- Saddle point problems: Stokes (3x3 with zero blocks), elasticity with (nearly) incompressible material, Navier-Stokes with stabilisation (3x3 blocks)

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \mathbf{f}, \quad \begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{B}_1 \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{B}_2 \\ \mathbf{B}_1^{\mathsf{T}} & \mathbf{B}_2^{\mathsf{T}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{p} \end{pmatrix} = \mathbf{f}, \quad \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{B}_1 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{B}_2 \\ \mathbf{B}_1^{\mathsf{T}} & \mathbf{B}_2^{\mathsf{T}} & \mathbf{C}_C \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{p} \end{pmatrix} = \mathbf{f}$$

$\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ correspond to scalar (elliptic) operators
$\Rightarrow$ Tuned linear algebra **and** tuned solvers

GPU integration into FEAST

Motivation: Bandwidth in a CPU/GPU node



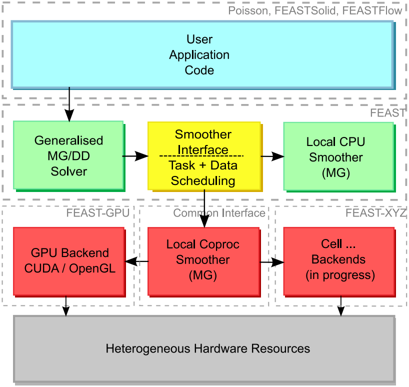
Minimally invasive integration

GPUs as accelerators for the most computationally intense parts

CPUs execute outer MLDD solver

No changes to applications required!

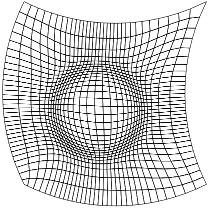
global BiCGStab
preconditioned by
  global multilevel (V 1+1)
  additively smoothed by
    for all Ω_i: **local multigrid**
  coarse grid solver: UMFPACK

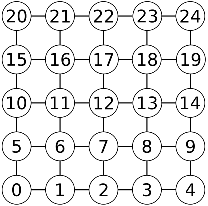


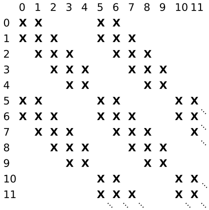
Fine-grained parallelisation of multigrid on GPUs

Fundamental building block in FEAST

- Local (geometric) multigrid on a $N = M \times M$ patch/subdomain
- Exploit generalised tensor product property



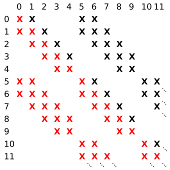




In the following

- Parallelisation of inherently sequential, numerically strong smoothers (preconditioners) for more than 100 000 threads on the GPU

Gauß-Seidel smoother

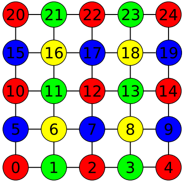


Starting point

- Explicit coupling, but inherently sequential (*'natural order GS'*)
- Exact parallelisation (wavefronts) not efficient

Decouple dependencies via colouring $\Rightarrow$ indep. parallel work

- Standard-GS Update: 'left-bottom'
- Red: Jacobi
- Green: Coupling with left
- Blue: Coupling with bottom
- Yellow: Coupling with left and bottom



Analysis

- Parallel efficiency: 4 sweeps with $\approx N/4$ parallel work each
- Numerical efficiency: Full coupling only in last sweep

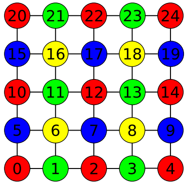
Gauß-Seidel smoother

Observation

- After decoupling via colours, the ‘Standard-GS’-Update is suboptimal

Better inexact parallelisation: ‘All-Colour-Coupling’

- Rot: Jacobi
- Green: Coupling with left and right
- Blue: Coupling with top and bottom
- Yellow: Full coupling (8 neighbours)
- More computation than standard colouring



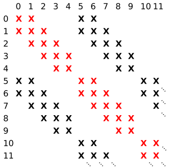
Analysis

- Corresponds to renumbering of the mesh points
- Convergence rates of sequential GS are recovered
- Total efficiency: $\rightarrow$ later

Tridiagonal smoother

Starting point

- Good for 'line-wise' anisotropies
- *'Alternating Direction Implicit'* technique (ADI) alternately acts line- and column-wise
- CPU implementation: Thomas-Algorithm



Observations

- One independent tridiagonal system per mesh row
- Top-level parallelisation: All mesh rows
- Implicit coupling: Wavefront and colouring techniques not applicable

Tridiagonal and TriGS smoothers

Cyclic reduction for tridiagonal systems

- Exact, stable (w/o pivoting) and cost-efficient
- Problem: Classical formulation parallelises computation but not memory accesses (multibank, shared memory)
- Developed a better formulation, 2-4x faster (published in TPDS, online version available)

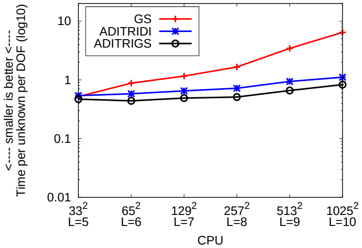
TriGS smoother

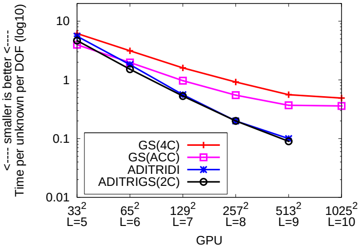
- Combination of tridiagonal and Gauß-Seidel smoother: Shift known results from previous row to right hand side and solve remaining tridiagonal system per row
- ADI-TRIGS: most robust generalised TP smoother in FEAST
- Difference to tridiagonal solvers: Mesh rows depend sequentially on each other
- Use colouring to decouple the dependencies between rows

Evaluation: Total efficiency on CPU and GPU

Test problem (one subdomain)

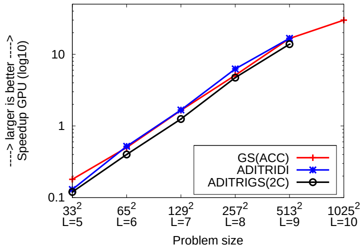
- Generalised Poisson with anisotropic diffusion
- Total efficiency: Time per unknown per digit (μs)
- Mixed precision iterative refinement multigrid solver
- Strong smoothers required





GPU only saturated for sufficiently large problem sizes

Speedup GPU vs. CPU



Summary for local problems

- Factor 10-30 (dep. on precision and smoother selection) speedup over already highly tuned CPU implementation
- Same functionality on CPU and GPU
- Balancing of numerical and parallel efficiency (hardware-oriented numerics)

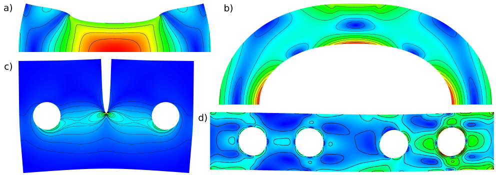
Results: FEAST on large GPU clusters

Linearised elasticity

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \mathbf{f}$$

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

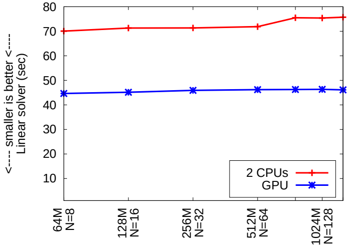
global multivariate BiCGStab
block-preconditioned by
Global multivariate multilevel (V 1+1)
additively smoothed (block GS) by
for all Ω_i: solve $\mathbf{A}_{11}\mathbf{c}_1 = \mathbf{d}_1$ by
local scalar multigrid
update RHS: $\mathbf{d}_2 = \mathbf{d}_2 - \mathbf{A}_{21}\mathbf{c}_1$
for all Ω_i: solve $\mathbf{A}_{22}\mathbf{c}_2 = \mathbf{d}_2$ by
local scalar multigrid
coarse grid solver: UMFPACK

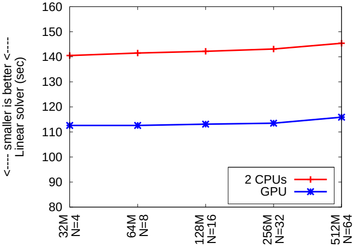


Weak scalability

Simultaneous doubling of problem size and resources

- Left: Poisson, 160 dual-CPU nodes, max. $1.3 \cdot 10^9$ DOF
- Right: Linearised elasticity, 64 dual-CPU nodes, max. $0.5 \cdot 10^9$ DOF

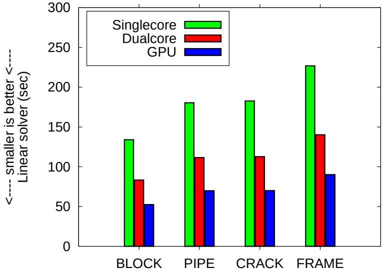


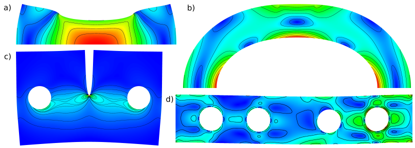


Results

- No loss of weak scalability despite local acceleration
- 1.3 billion unknowns (no stencil!) on 160 GPUs in less than 50 s

Speedup linearised elasticity





- USC cluster in Los Alamos, 16 dualcore nodes
- Problem size 128 M DOF
- Dualcore 1.6x faster than singlecore (memory wall)
- GPU 2.6x faster than singlecore, 1.6x than dualcore

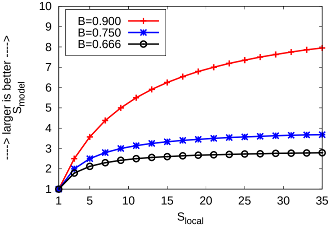
Speedup analysis

Theoretical model of expected speedup

- Integration of GPUs increases resources
- Correct model: strong scaling within each node
- Acceleration potential of the elasticity solver: $R_{\text{acc}} = 2/3$ (remaining time in MPI and the outer solver)
- $S_{\text{max}} = \frac{1}{1-R_{\text{acc}}}$ $\qquad S_{\text{model}} = \frac{1}{(1-R_{\text{acc}})+(R_{\text{acc}}/S_{\text{local}})}$

This example

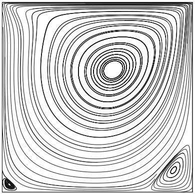
Accelerable fraction R_{acc}	66%
Local speedup S_{local}	9x
Modeled speedup S_{model}	2.5x
Measured speedup S_{total}	2.6x
Upper bound S_{max}	3x

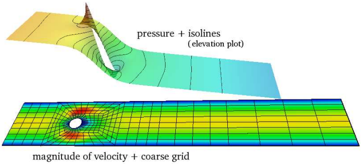


Stationary laminar flow (Navier-Stokes)

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{B}_1 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{B}_2 \\ \mathbf{B}_1^\mathsf{T} & \mathbf{B}_2^\mathsf{T} & \mathbf{C} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{g} \end{pmatrix}$$

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 Ω_i: solve for $\mathbf{u}_1$ with **local MG**
     - for all Ω_i: solve for $\mathbf{u}_2$ with **local MG**
  2) update RHS: $\mathbf{d}_3 = -\mathbf{d}_3 + \mathbf{B}^\mathsf{T}(\mathbf{c}_1, \mathbf{c}_2)^\mathsf{T}$
  3) scale $\mathbf{c}_3 = (\mathbf{M}_p^\mathsf{L})^{-1}\mathbf{d}_3$





Stationary laminar flow (Navier-Stokes)

Solver configuration

- Driven cavity: Jacobi smoother sufficient
- Channel flow: ADI-TRIDI smoother required

Speedup analysis

	R_{acc}		S_{local}		S_{total}	
	L9	L10	L9	L10	L9	L10
DC Re250	52%	62%	9.1x	24.5x	1.63x	2.71x
Channel flow	48%	–	12.5x	–	1.76x	–

Shift away from domination by linear solver (fraction of FE assembly and linear solver of total time, max. problem size)

DC Re250		Channel	
CPU	GPU	CPU	GPU
12:88	31:67	38:59	**68:28**

Conclusions and future work

Conclusions and future work

- Hardware-oriented numerics
  - Balance conflicting efficiency goals (numerics, hardware, scalability)
  - Make code future-proof with respect to long-term hardware trends
- Integration of GPUs into FEAST
  - Multicolouring and cyclic reduction for strong parallel multigrid smoothers
  - Accelerate many applications without modifying them, rather than delivering *the optimal speedup for one specific problem*
- Significant speedups in walltime to solution for large CSM and CFD problems on GPU clusters
- Future work
  - Investigate finite element assembly on GPUs (see Chris Cecka's talk later today)
  - Design solution schemes with higher acceleration potential (increase locality)

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