GPU Cluster Computing for Finite Element Applications

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

Applied Mathematics
TU Dortmund, Germany
dominik.goeddeke@math.tu-dortmund.de

SIAM Conference on Computational Science and Engineering
Emerging Manycore Architectures Minisymposium
Miami, March 6, 2009

The big picture

Scientific computing is in the middle of a paradigm shift

ILP wall memory wall characteristic feature size

heat power consumption leaking voltage

Hardware evolves towards parallelism and heterogeneity

multicore CPUs Cell BE processor GPUs

Emerging manycore architectures

accelerators algorithm design for 10000s of threads

FEAST –

Hardware-oriented Numerics

Mesh structure

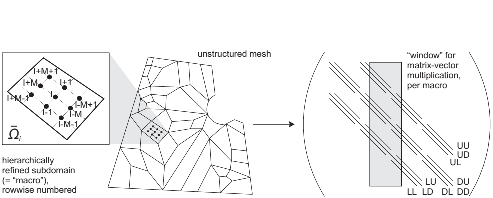
Fully adaptive grids

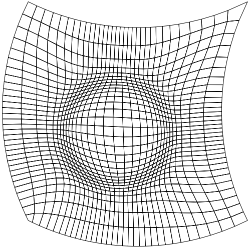
Maximum flexibility
‘Stochastic’ numbering
Unstructured sparse matrices
Indirect addressing, very slow.

Locally structured grids

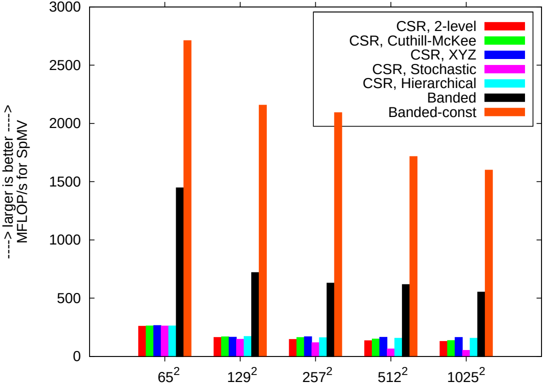
Logical tensor product
Fixed banded matrix structure
Direct addressing ($\Rightarrow$ fast)
r-adaptivity

Unstructured macro mesh of tensor product subdomains





Example: SpMV on TP grid



- Opteron X2 2214, 2.2 GHz, 2x1 MB L2 cache, one thread
- 50 vs. 550 MFLOP/s for interesting large problem size
- Caching of coefficient vector, full streaming bandwidth for A
- `const`: constant coefficients $\Rightarrow$ stencil

ScaRC – Scalable Recursive Clustering

- Minimal overlap by extended Dirichlet BCs
- Hybrid multilevel domain decomposition method
- Inspired by parallel MG (”best of both worlds”)
  - Multiplicative vertically (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 Ω_i: **local multigrid**
  coarse grid solver: UMFPACK

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

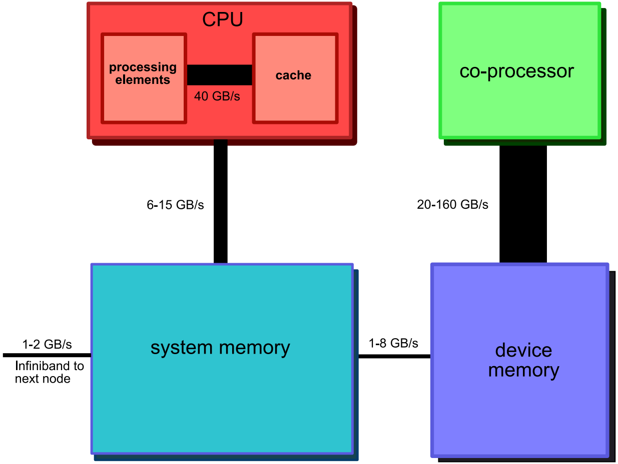
- Linearised elasticity with compressible material
- Stokes
- Saddle point problems: Elasticity with (nearly) incompressible material, Navier-Stokes with stabilisation

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

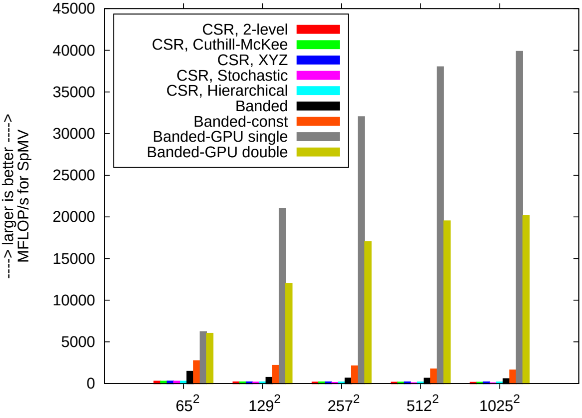
$\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ correspond to scalar (elliptic) operators
$\Rightarrow$ Tuned linear algebra **and** tuned solvers

Co-processor integration into FEAST

Bandwidth in a CPU/GPU node



Example: SpMV on TP grid



40 GFLOP/s, 140 GB/s with CUDA on GeForce GTX 280
'only' 13 GFLOP/s on 8800 GTX (90 GB/s peak)

Mixed precision approach

	single precision		double precision	
Level	Error	Reduction	Error	Reduction
2	2.391E-3		2.391E-3	
3	5.950E-4	4.02	5.950E-4	4.02
4	1.493E-4	3.98	1.493E-4	3.99
5	3.750E-5	3.98	3.728E-5	4.00
6	1.021E-5	3.67	9.304E-6	4.01
7	6.691E-6	1.53	2.323E-6	4.01
8	2.012E-5	0.33	5.801E-7	4.00
9	7.904E-5	0.25	1.449E-7	4.00
10	3.593E-4	0.22	3.626E-8	4.00

- Poisson $-\Delta\mathbf{u} = \mathbf{f}$ on $[0,1]^2$ with Dirichlet BCs, MG solver
- Bilinear conforming Finite Elements (Q_1) on cartesian mesh
- L_2 error against analytical reference solution
- Residuals indicate convergence, but results are completely off
- Mixed precision solver: double precision Richardson, preconditioned with single precision MG ('gain one digit')
- Same results as entirely in double precision

Example: Multigrid on TP grid

	Core2Duo (double)		GTX 280 (mixed)		
Level	time(s)	MFLOP/s	time(s)	MFLOP/s	speedup
7	0.021	1405	0.009	2788	2.3x
8	0.094	1114	0.012	8086	7.8x
9	0.453	886	0.026	15179	17.4x
10	1.962	805	0.073	21406	26.9x

- Poisson on unitsquare, Dirichlet BCs, *not only a matrix stencil*
- 1M DOF, multigrid, FE-accurate in less than 0.1 seconds!
- 27x faster than CPU
- 1.7x faster than pure double on GPU
- 8800 GTX (correction loop on CPU): 0.44 seconds on level 10

Minimally invasive integration

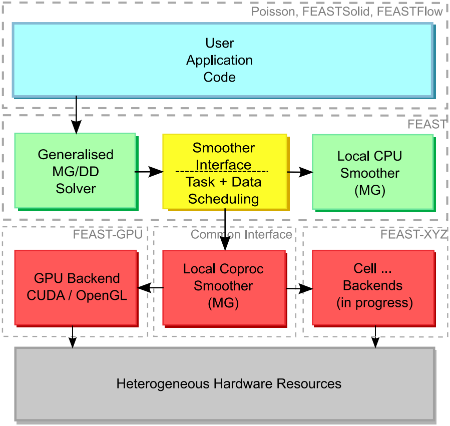
global BiCGStab
preconditioned by
global multilevel (V 1+1)
additively smoothed by
for all Ω_i: **local multigrid**
coarse grid solver: UMFPACK

All outer work: CPU, double

Local MGs: GPU, single

GPU is preconditioner

Applicable to many co-processors



Minimally invasive integration

General approach

- Balance acceleration potential and integration effort
- Accelerate many different applications built on top of one central FE and solver toolkit
- Diverge code paths as late as possible
- No changes to application code!
- Retain all functionality
- Do not sacrifice accuracy

Challenges

- Heterogeneous task assignment to maximise throughput
- Limited device memory (modeled as huge L3 cache)
- Overlapping CPU and GPU computations
- Building dense accelerated clusters

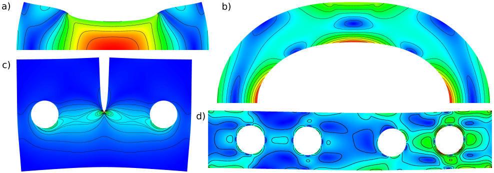
Some results

Linearised elasticity

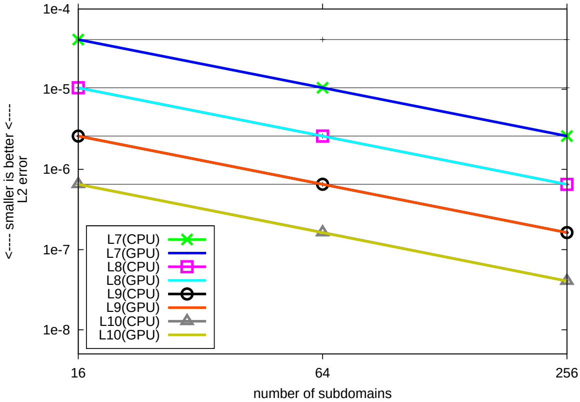
$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \mathbf{f}$$

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

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

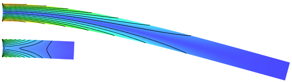


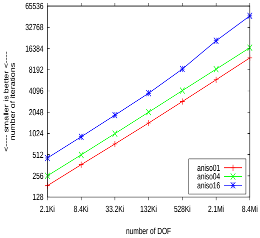
Accuracy (I)



- **Same results for CPU and GPU**
- L_2 error against analytically prescribed displacements
- Tests on 32 nodes, 512 M DOF

Accuracy (II)

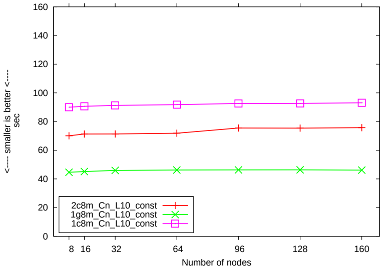


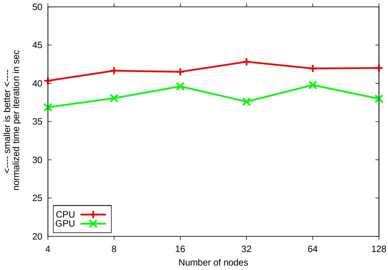


Cantilever beam, aniso 1:1, 1:4, 1:16
Hard, very ill-conditioned CSM test
CG solver: > 2x iterations per refinement
GPU-ScaRC solver: same results as CPU

aniso04	Iterations		Volume		y-Displacement	
refinement L	CPU	GPU	CPU	GPU	CPU	GPU
8	4	4	1.6087641E-3	1.6087641E-3	-2.8083499E-3	-2.8083499E-3
9	4	4	1.6087641E-3	1.6087641E-3	-2.8083628E-3	-2.8083628E-3
10	4.5	4.5	1.6087641E-3	1.6087641E-3	-2.8083667E-3	-2.8083667E-3
aniso16						
8	6	6	6.7176398E-3	6.7176398E-3	-6.6216232E-2	-6.6216232E-2
9	**6**	**5.5**	6.7176427E-3	6.7176427E-3	-6.621655**1**E-2	-6.621655**2**E-2
10	5.5	5.5	6.7176516E-3	6.7176516E-3	-6.621750**1**E-2	-6.621750**2**E-2

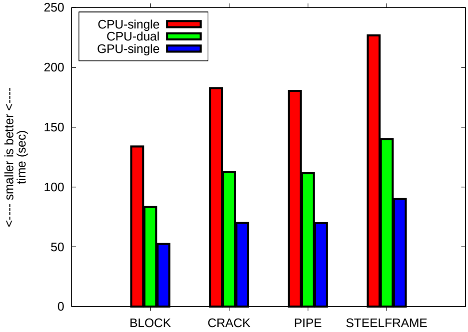
Weak scalability





- Outdated cluster, dual Xeon EM64T,
- 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

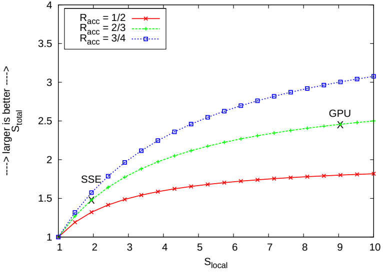


- 16 nodes, Opteron X2 2214,
- NVIDIA Quadro FX 5600 (76 GB/s BW), OpenGL
- Problem size 128 M DOF
- Dualcore 1.6x faster than singlecore
- GPU 2.6x faster than singlecore, 1.6x than dual

Speedup analysis

- Addition of GPUs increases resources
- $\Rightarrow$ 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

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

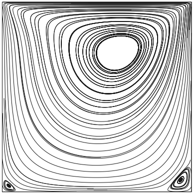
- 4-node cluster
- Opteron X2 2214
- GeForce 8800 GTX (90 GB/s BW), CUDA
- Driven cavity and channel flow around a cylinder

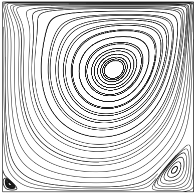
fixed point iteration
solving linearised subproblems with
global BiCGStab (reduce initial residual by 1 digit)
Block-Schurcomplement preconditioner
1) approx. solve for velocities with
global MG (V 1+0), additively smoothed by

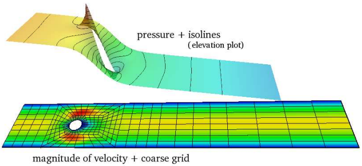
for all Ω_i: solve for $\mathbf{u_1}$ with
local MG

for all Ω_i: solve for $\mathbf{u_2}$ with
local MG

2) update RHS: $\mathbf{d_3} = -\mathbf{d_3} + \mathbf{B^T}(\mathbf{c_1}, \mathbf{c_2})^\mathbf{T}$
3) scale $\mathbf{c_3} = (\mathbf{M_p^L})^{-1}\mathbf{d_3}$







Speedup analysis

	R_{acc}		S_{local}		S_{total}	
	L9	L10	L9	L10	L9	L10
DC Re100	41%	46%	6x	12x	1.4x	1.8x
DC Re250	56%	58%	5.5x	11.5x	1.9x	2.1x
Channel flow	60%	–	6x	–	1.9x	–

Important consequence: Ratio between assembly and linear solve changes significantly

DC Re100		DC Re250		Channel flow	
plain	accel.	plain	accel.	plain	accel.
29:71	50:48	11:89	25:75	13:87	26:74

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

Collaborative work with

FEAST group (TU Dortmund)

Robert Strzodka (Max Planck Institut Informatik)

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

Max Planck Center for visual computing and communication

Los Alamos NATIONAL LABORATORY EST. 1943

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

Supported by Deutsche Forschungsgemeinschaft, project TU 102/22-1