technische universität
dortmund

fakultät für mathematik
LS III (IAM)

UCHPC

UnConventional High Performance Computing for Finite Element Simulations

S. Turek, Chr. Becker, S. Buijssen, D. Göddeke, H.Wobker
(FEAST Group)

Institut für Angewandte Mathematik, TU Dortmund

http://www.mathematik.tu-dortmund.de/LS3
http://www.featflow.de
http://www.feast.tu-dortmund.de

Motivation

technische universität dortmund

- The 'free ride' is over, paradigm shift in HPC:
  - memory wall (in particular for sparse Linear Algebra problems)
  - physical barriers (heat, power consumption, leaking voltage)
  - applications no longer run faster automatically on newer hardware
- Heterogeneous hardware: commodity CPUs plus co-processors
  - graphics cards (GPU)
  - Cell BE processor
  - HPC accelerators (e.g. ClearSpeed)
  - reconfigurable hardware (FPGA)
- Finite Element Methods (FEM) and Multigrid solvers: most flexible, efficient and accurate simulation tools for PDEs.

Aim of this Talk

technische universität dortmund

High Performance Computing

meets

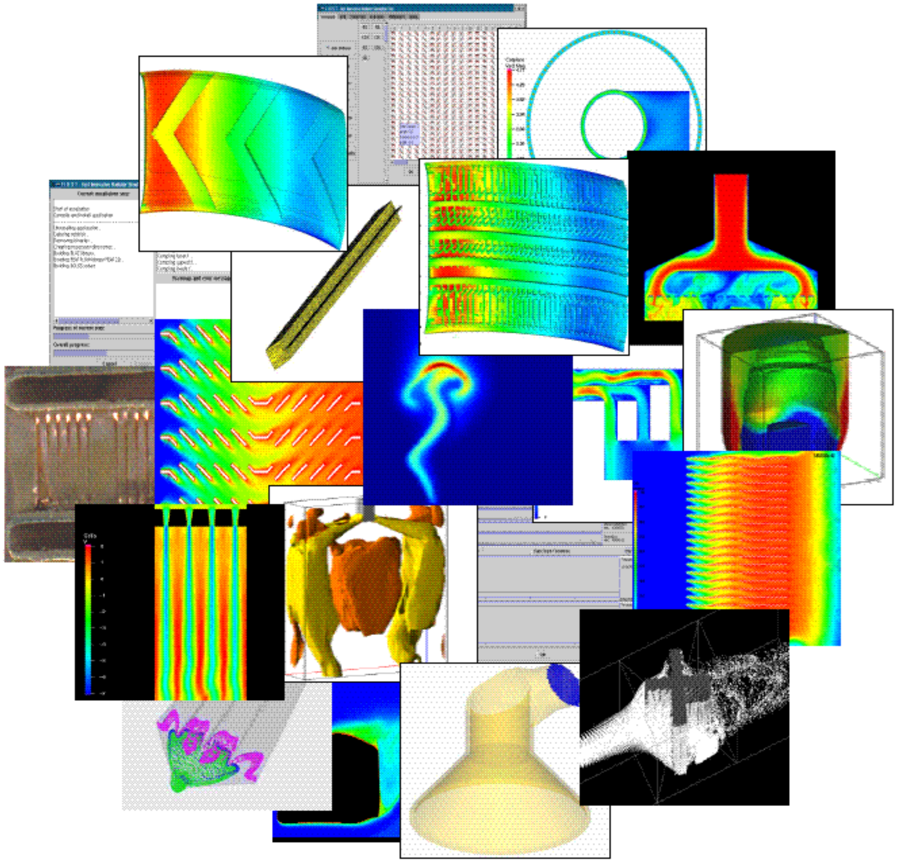
Hardware-oriented Numerics

on

Unconventional Hardware

for

Finite Element Methods



1) Hardware-Oriented Numerics

technische universität dortmund

What is **'Hardware-Oriented Numerics'**?

- It is more than '*good Numerics*' and '*good Implementation*' on High Performance Computers
- Critical quantity: **'Total Numerical Efficiency'**

Total Numerical Efficiency

technische universität dortmund

- '**High** (guaranteed) **accuracy** for user-specific quantities with minimal #d.o.f. (~ *N*) via **fast and robust solvers** – for a wide class of parameter variations – with **optimal numerical complexity** (~ *O*(*N*)) while exploiting a significant percentage of the **available huge sequential/ parallel GFLOP/s rates** at the same time'
- **FEM Multigrid solvers** with **a posteriori error control** for **adaptive meshing** are a candidate
- Is it easy to achieve high **'Total Numerical Efficiency'**?

Example: Fast Poisson Solvers

technische universität dortmund

- **Fast Multigrid Methods as general philosophy**
  - 'Optimized' versions for **scalar PDE** problems (≈Poisson problems) on **general meshes** should require ca. **1000 FLOPs** per unknown (in contrast to LAPACK for dense matrices with $O(N^3)$ FLOPs)

- Problem size 10^6 : Much less than 1 sec on PC (???)
- Problem size 10^{12}: Less than 1 sec on PFLOP/s computer

→ **More realistic (and much harder) 'Criterion' for Petascale Computing in Technical Simulations**

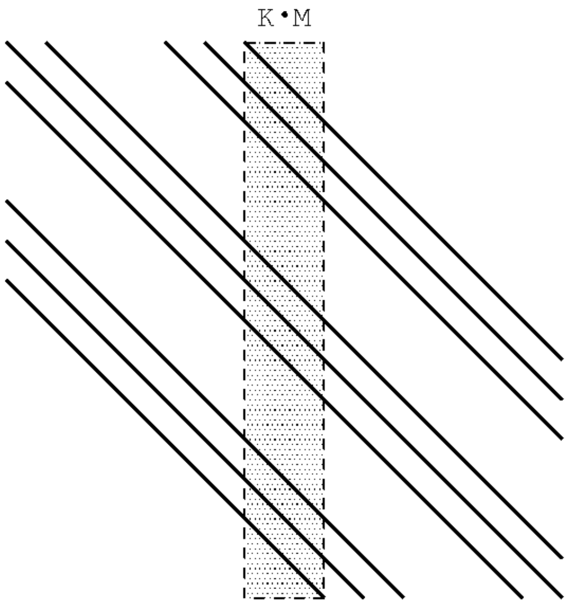
Main Component: 'Sparse' MV Application

technische universität dortmund

- Sparse **Matrix-Vector techniques** ('indexed DAXPY')

```
   DO 10 IROW=1,N
     DO 10 ICOL=KLD(IROW),KLD(IROW+1)-1
10     Y(IROW)=DA(ICOL)*X(KCOL(ICOL))+Y(IROW)
```

- Sparse Banded **MV techniques** on **generalized TP grids**



Grid Structure

<u>**Fully adaptive grids**</u>

Maximum flexibility

'Stochastic' numbering

Unstructured sparse matrices

Indirect addressing, very slow.

<u>**Locally structured grids**</u>

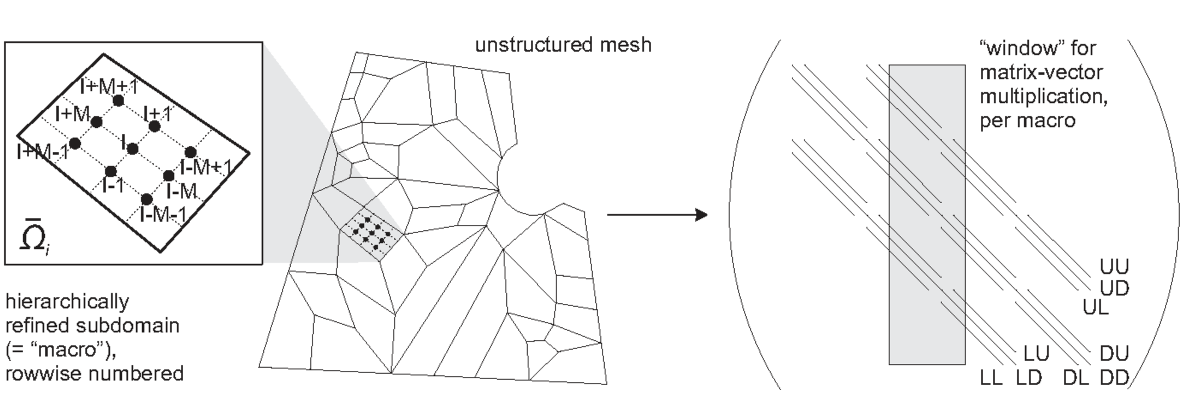
Logical tensor product

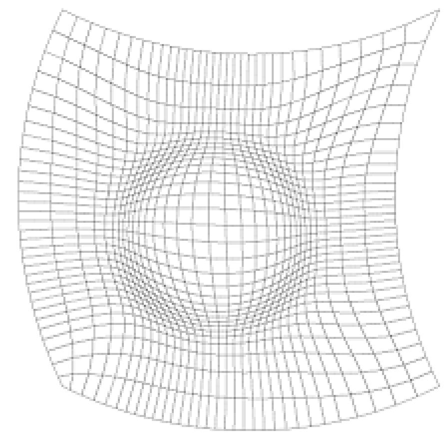
Fixed banded matrix structure

Direct addressing ($\Rightarrow$ fast)

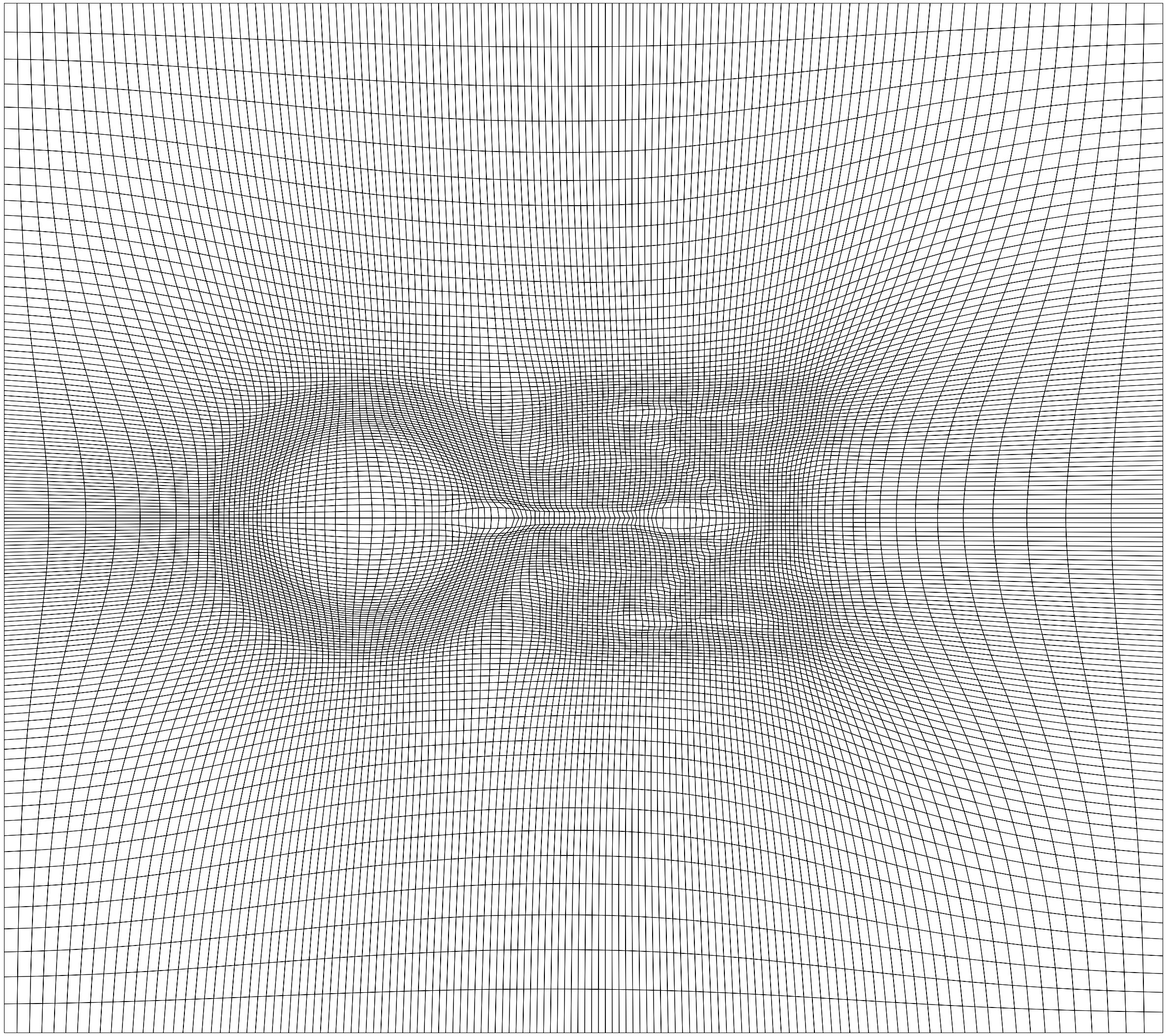
r-adaptivity

<u>Unstructured macro mesh of tensor product subdomains</u>

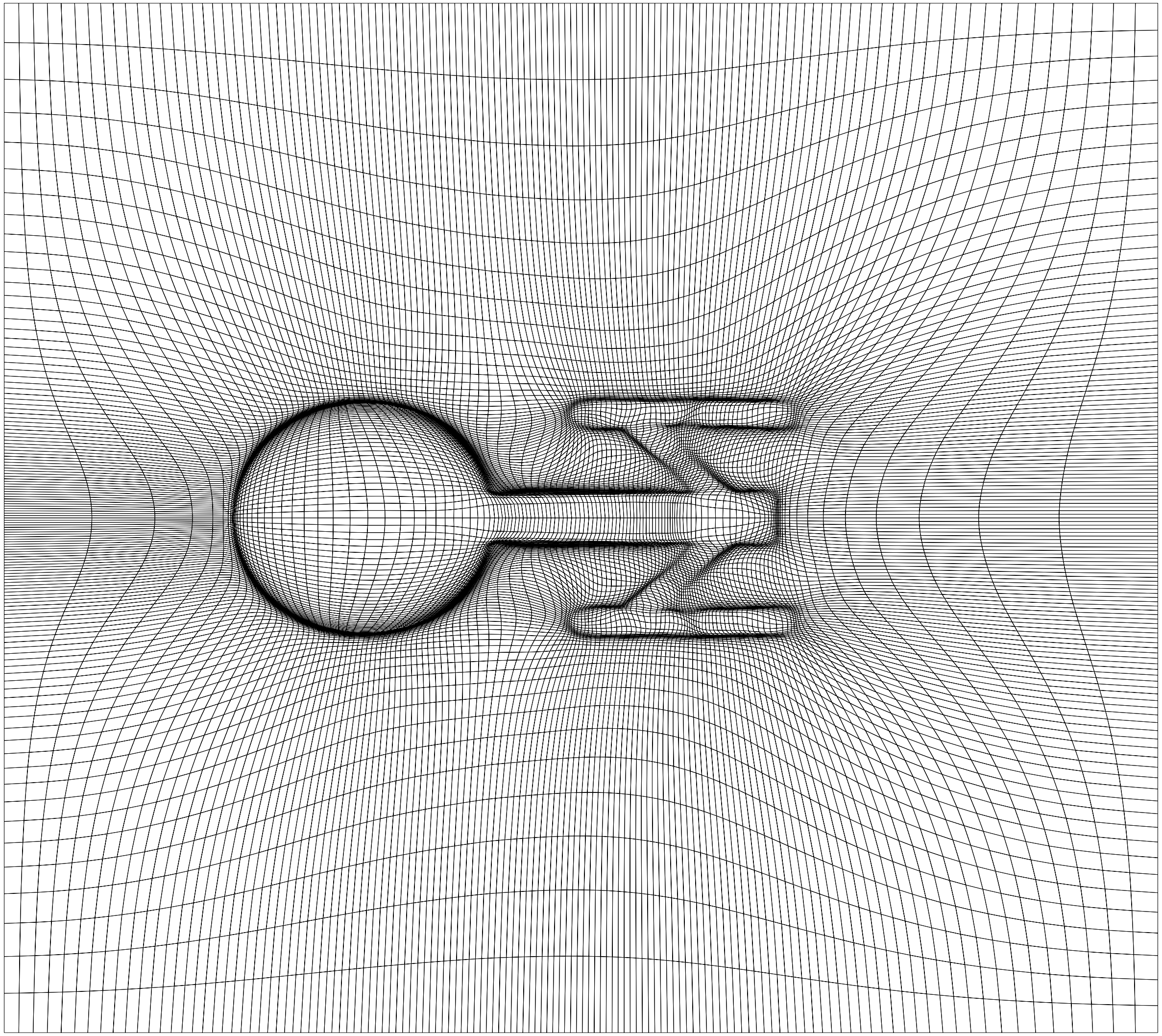




Generalized Tensorproduct Meshes

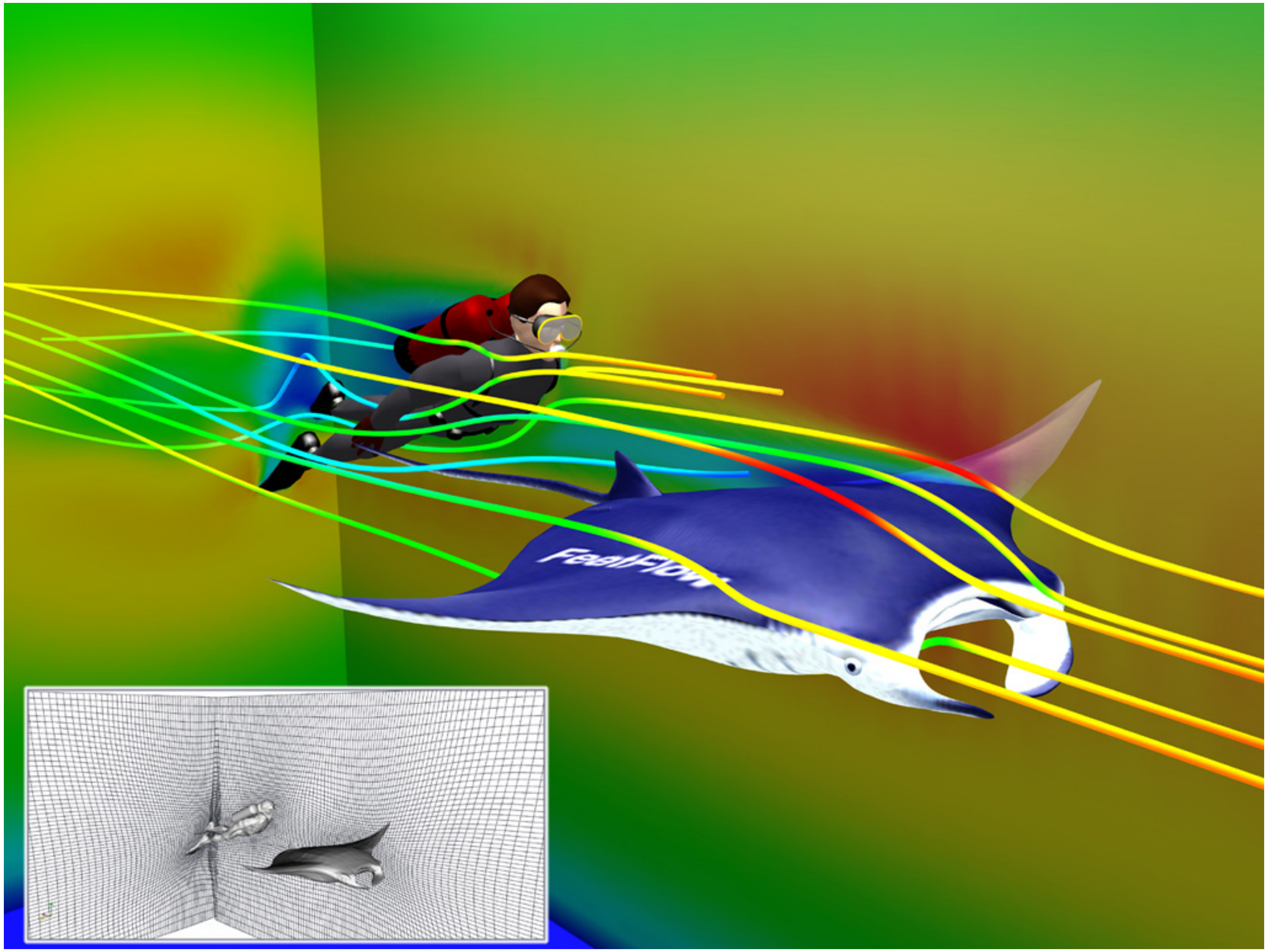


Generalized Tensorproduct Meshes



Generalized Tensorproduct Meshes

technische universität
dortmund

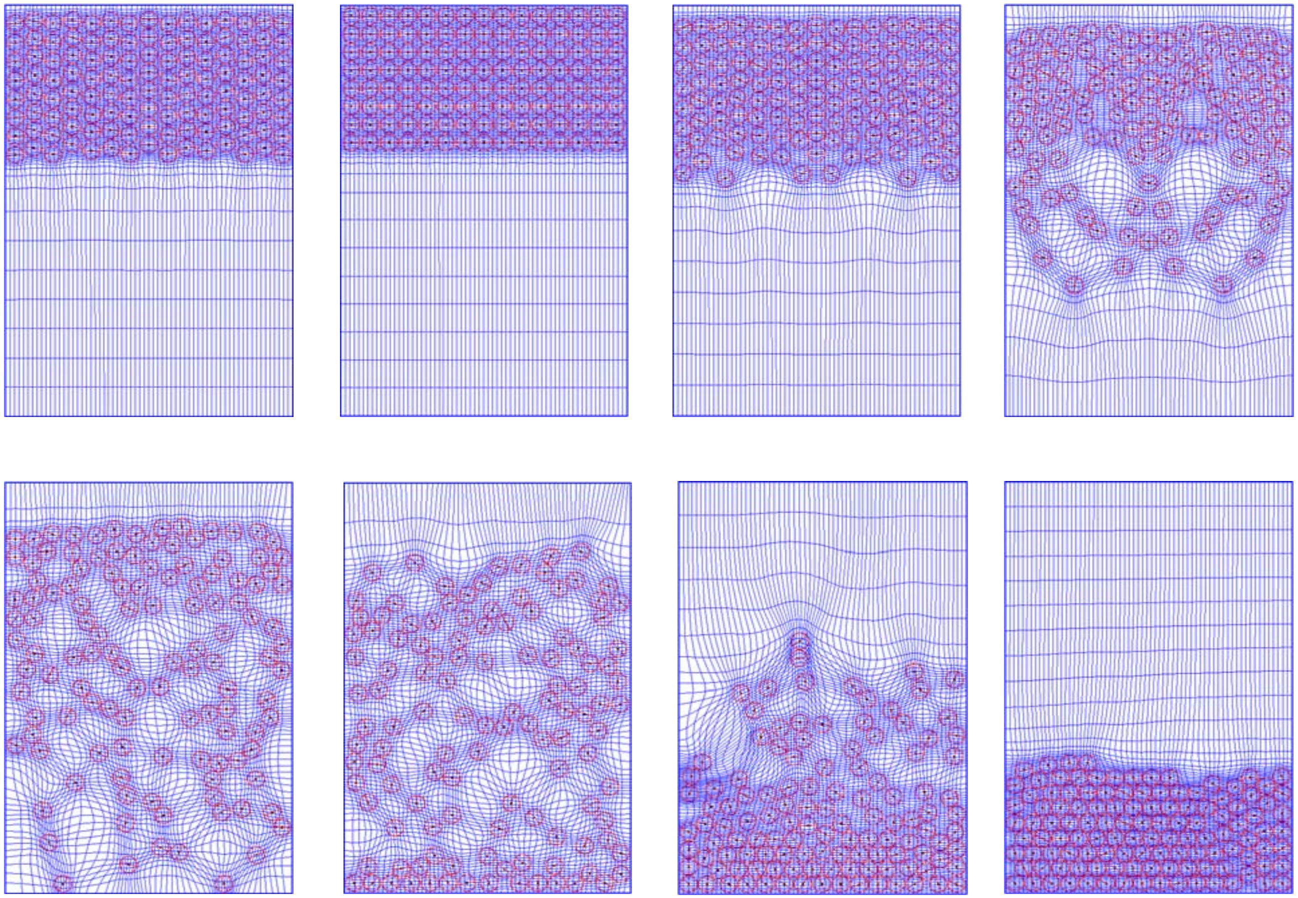


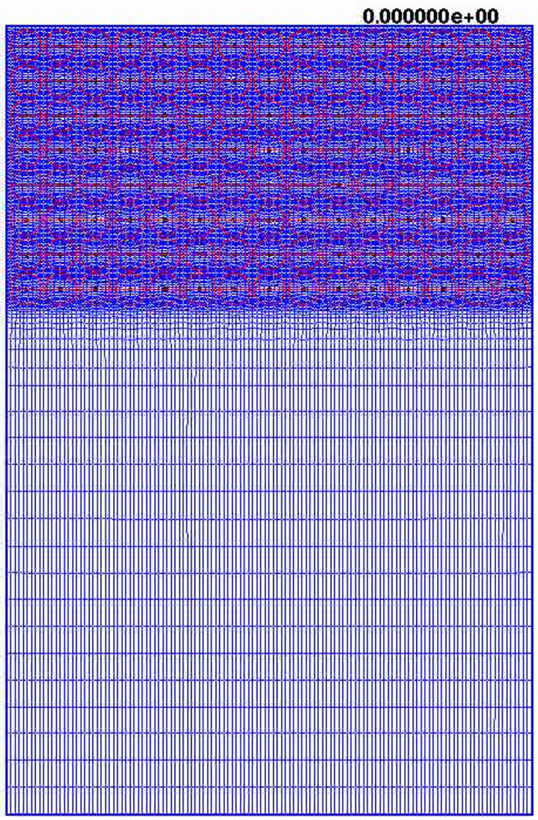
…with appropriate Fictitious Boundary techniques in FEATFLOW…..

Generalized Tensorproduct Meshes

technische universität dortmund

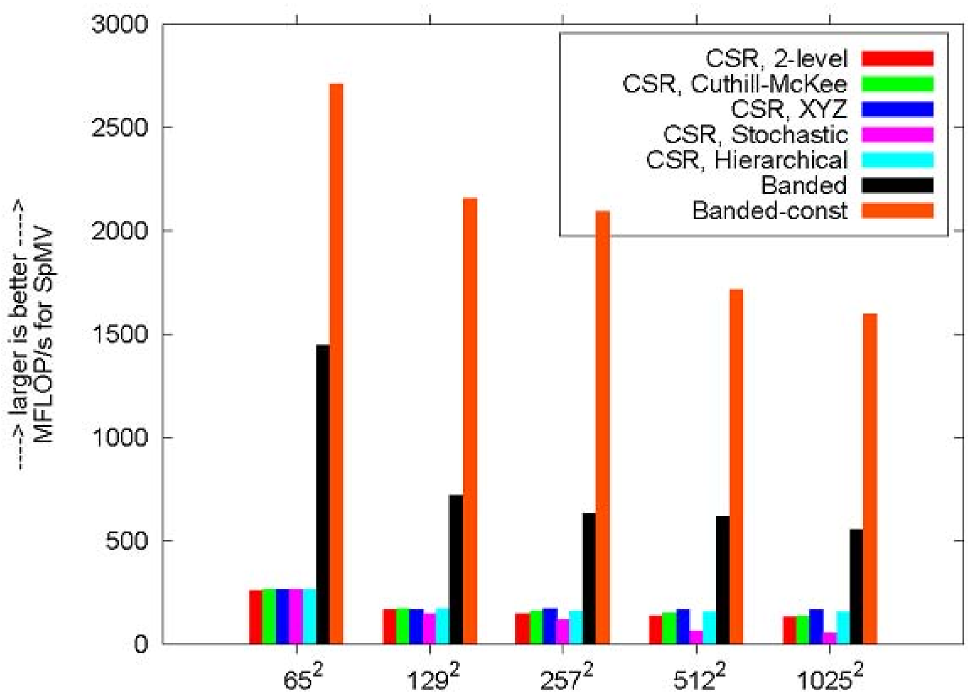
….dynamic CFD problems…..





Example: SpMV on TP Grid

technische universität dortmund



- 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

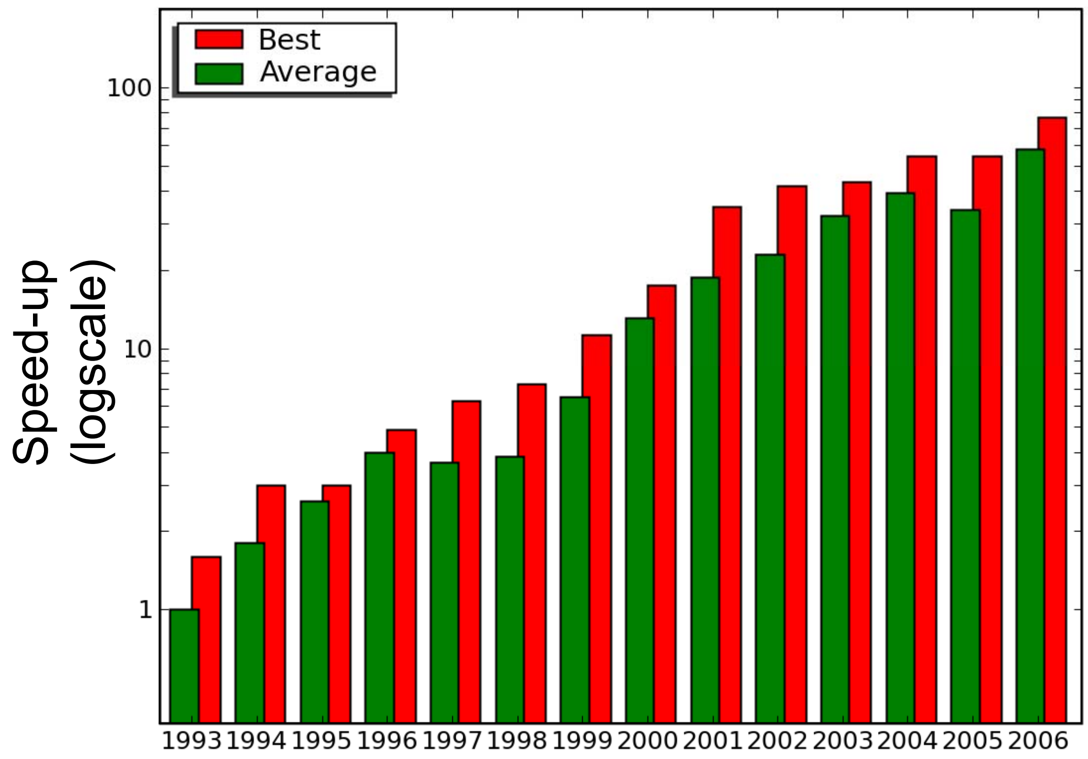
Observation I: Sparse MV Multiplication

Numbering	**4K DOF**	**66K DOF**	**1M DOF**
Stochastic	127	116	**50**
Hierarchical	251	159	**154**
Banded	1445	627	**550**
Stencil (const)	2709	2091	**1597**

In realistic scenarios, MFLOP/s rates are

- poor, and
- problem size dependent

Observation II: Full CFD Simulations



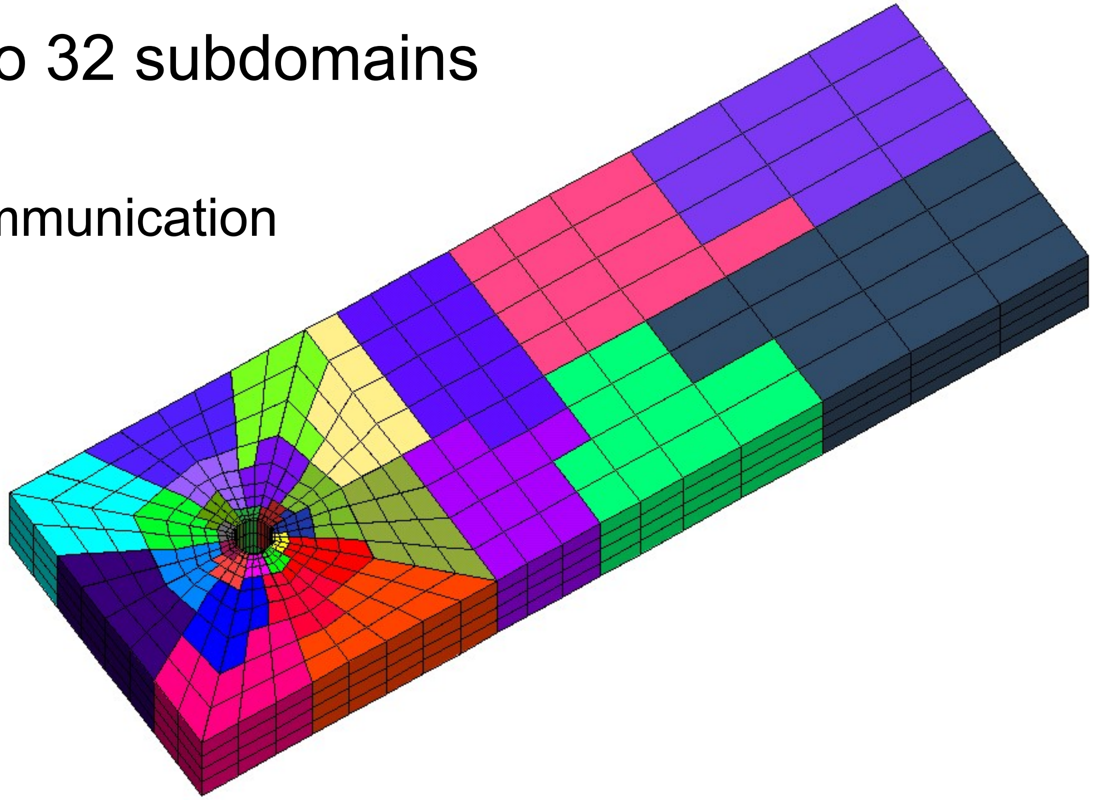
Speed-up of 100x for free in 10 years

Stagnation for standard simulation tools on conventional hardware

Observation III: Parallel Performance

technische universität dortmund

- Mesh partitioned into 32 subdomains
  - Problems due to communication
  - Numerical behavior vs. anisotropic meshes



	1 P.	2 P.	4 P.	8 P.	16 P.	32 P.	64 P.
%Comm.	10%	24%	36%	45%	47%	55%	56%
# PPP-IT	2.2	3.0	3.9	4.9	5.2	5.7	6.2

Summary

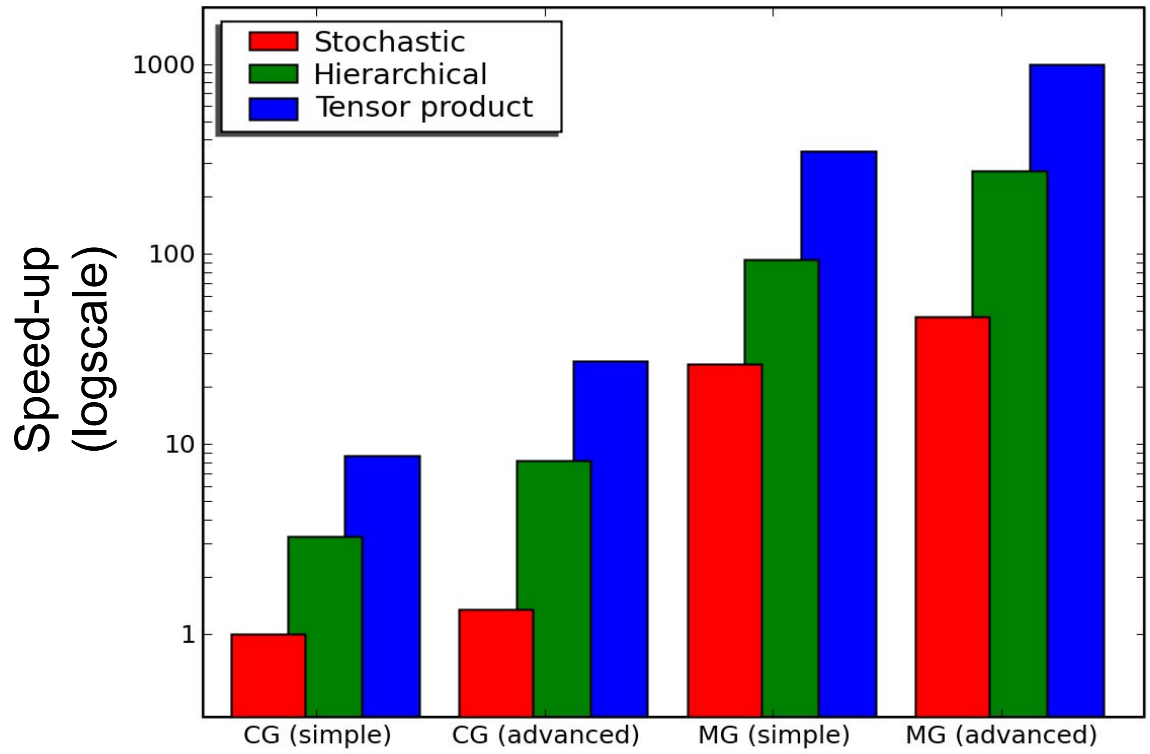
technische universität dortmund

- It is (almost) impossible to reach **Single Processor Peak Performance** with modern (= high numerical efficiency) FEM simulation tools
- Memory-intensive data/matrix/solver structures?
- **Parallel Peak Performance** with modern Numerics even harder, already for moderate processor numbers

Hardware-oriented Numerics (HwoN)

technische universität dortmund

FEM for 8 Mill. unknowns on general domain, 1 CPU, Poisson Problem in 2D



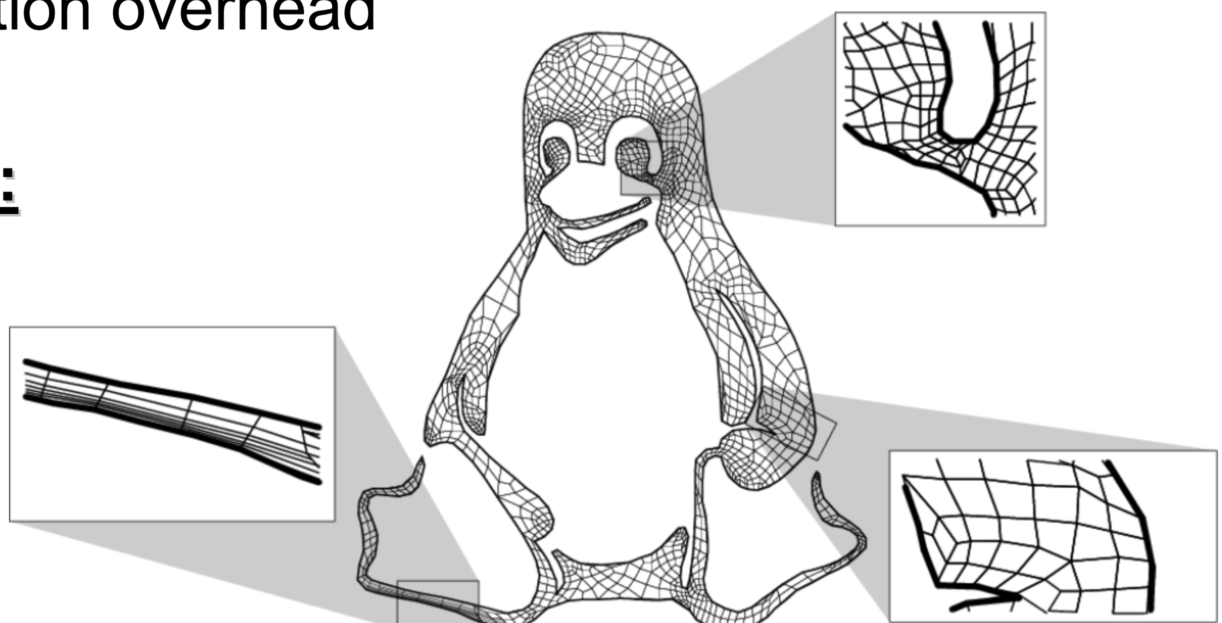
Dramatic improvement (factor 1000) due to **better Numerics** AND **better data structures/ algorithms**

FEAST – Realization of HwoN

technische universität dortmund

- **ScaRC** solver: Combine advantages of (parallel) domain decomposition and multigrid methods
- Cascaded multigrid scheme
- Hide anisotropies locally to increase robustness
- Globally unstructured – locally structured
- Low communication overhead

FEAST applications:
FEASTFlow (CFD)
FEASTSolid (CSM)
FEASTLBM (SKALB Project)



Solver Structure

technische universität dortmund

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

(Preliminary) State-of-the-Art

technische universität dortmund

- **Numerical efficiency?**
  → OK
- **Parallel efficiency?**
  → OK (tested up to 256 CPUs on NEC SX-8, commodity clusters)
- **Single processor efficiency?**
  → OK (for CPU)
- **'Peak' efficiency?**
  → NO
  → Special ***unconventional*** FEM Co-Processors

2) UnConventional HPC

technische universität
dortmund

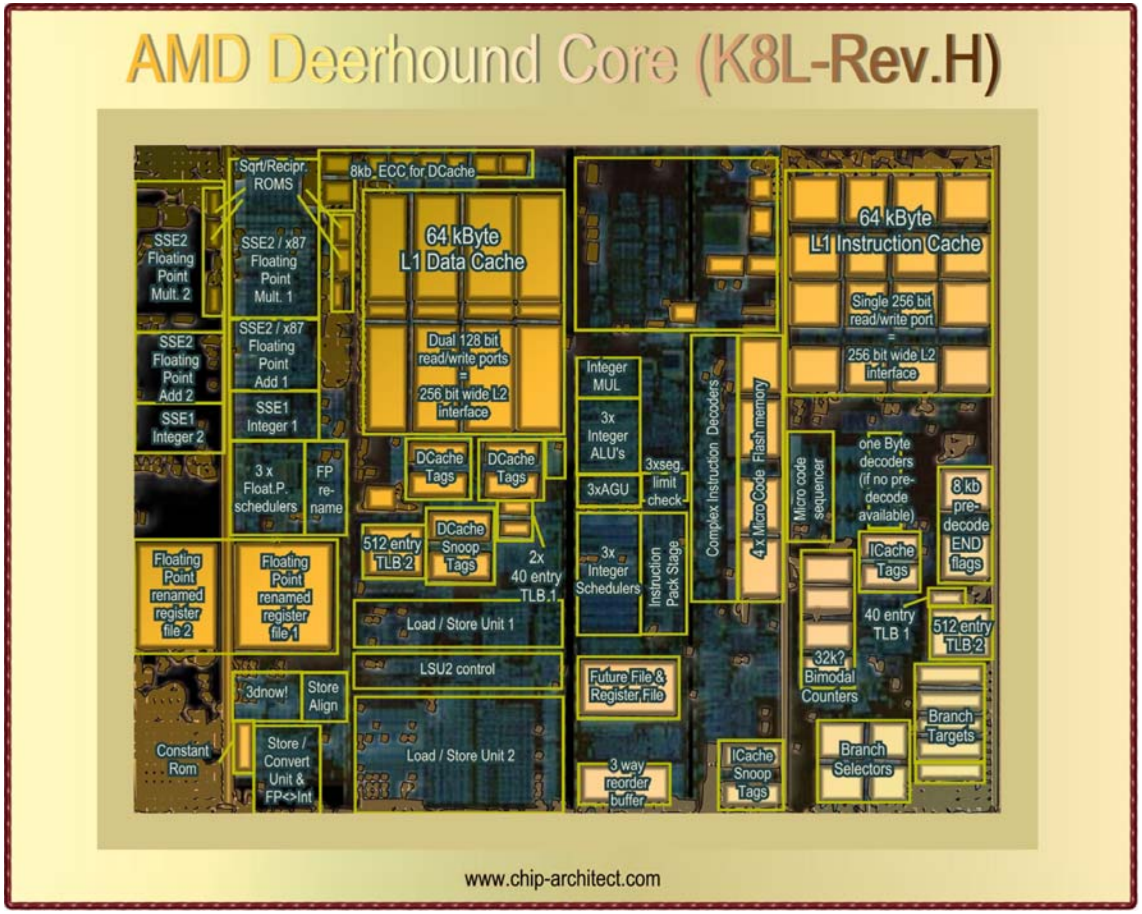
- Cell multicore processor (PS3),
7 synergistic processing units
@ 3.2 GHz, 218 GFLOP/s,
Memory @ 3.2 GHz

- GPU (NVIDIA GTX 285):
240 cores @ 1.476 GHz,
1.242 GHz memory bus (160 GB/s)
≈ 1.06 TFLOP/s

UnConventional High Performance Computing (UCHPC)

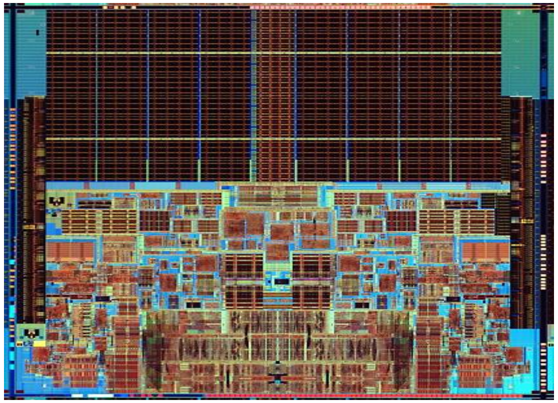
Why are GPUs and Cells so fast?

technische universität dortmund



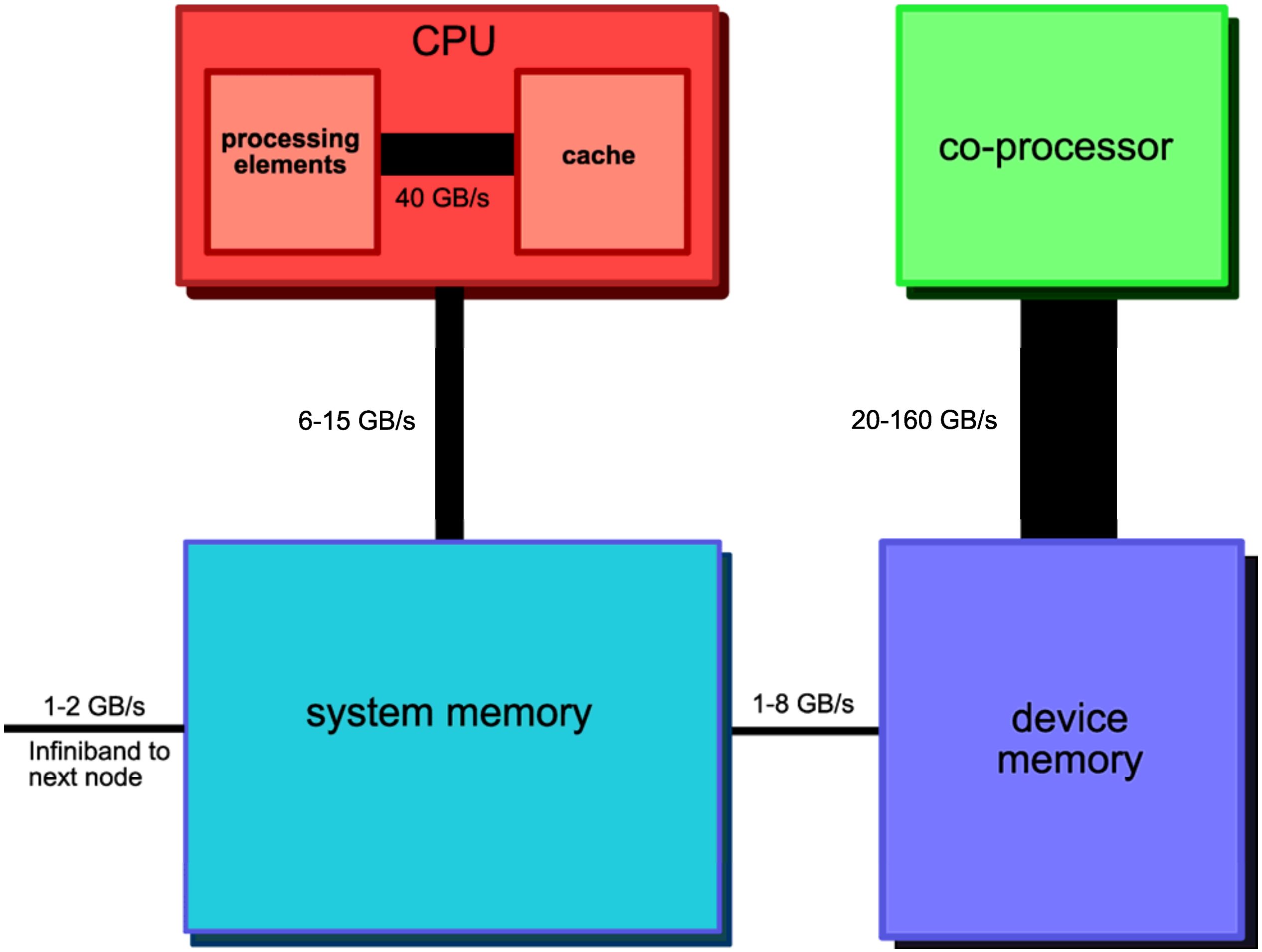
CPUs devote most of the transistors to caches and data movement for general purpose applications

GPUs and **Cells** are more “transistor-efficient“ w.r.t. floating point operations



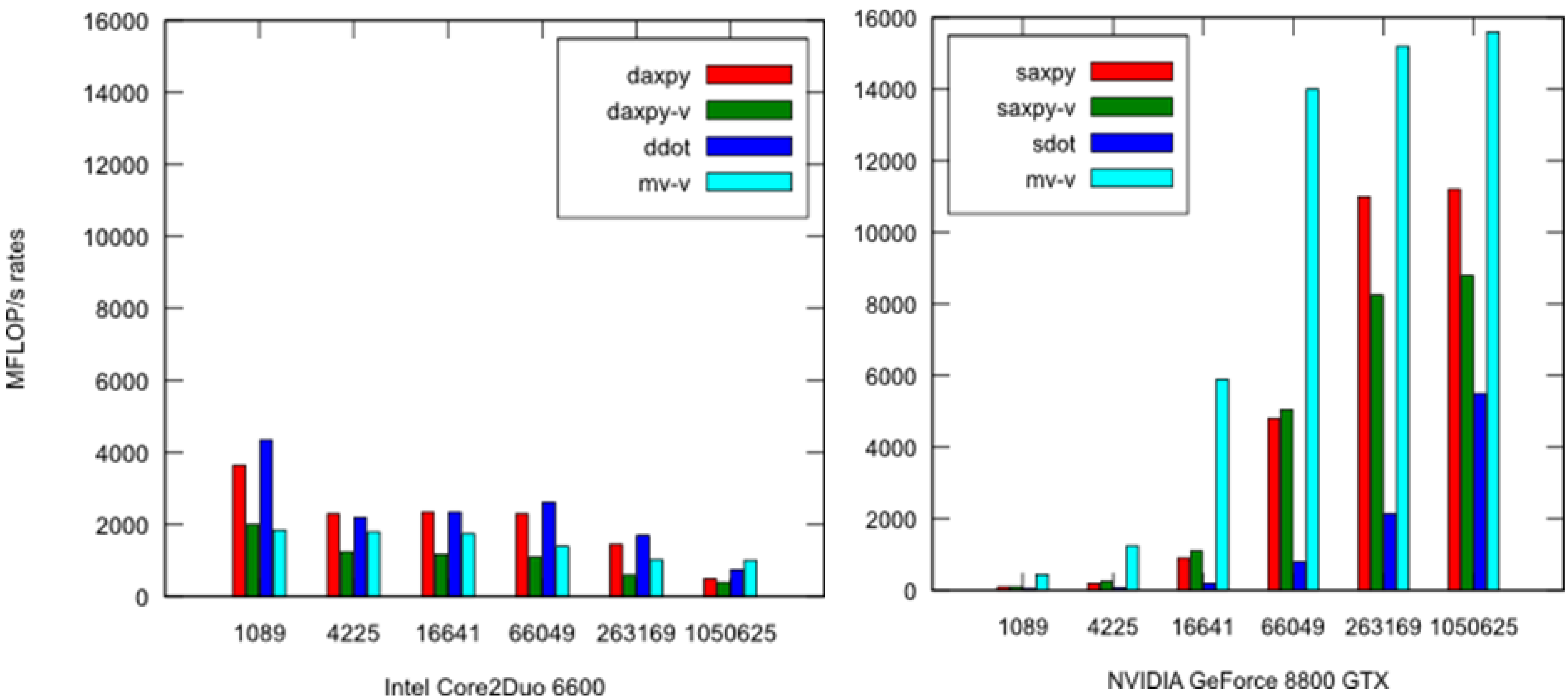
Bandwidth in a CPU/GPU Node

technische universität dortmund



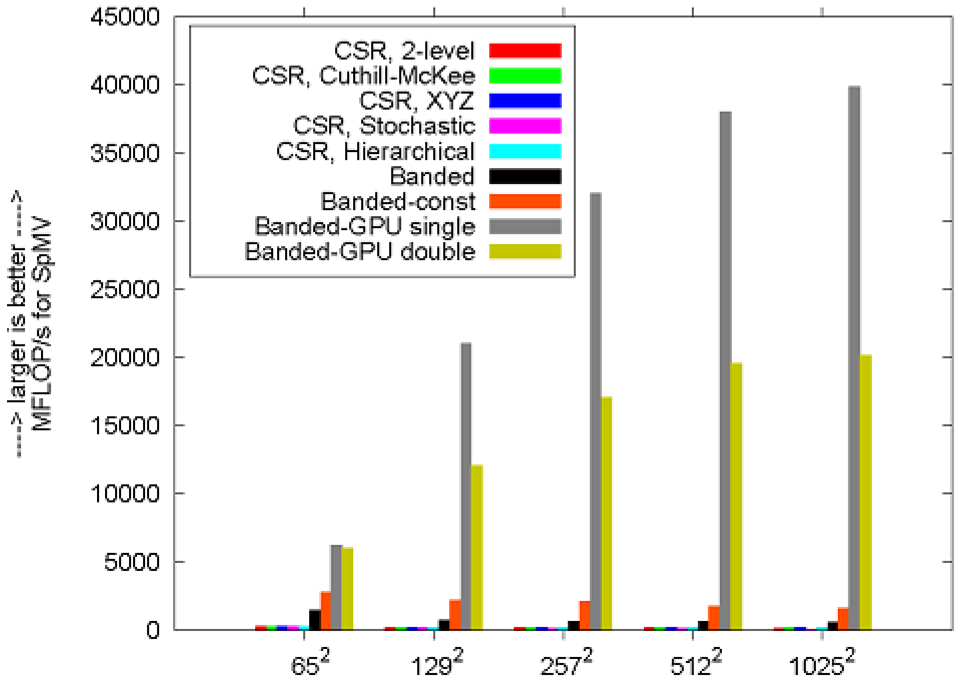
Benchmarks: FEM Building Blocks

- Typical performance of FEM building blocks SAXPY_C, SAXPY_V (variable coefficients), MV_V (9-point-stencil, Q1 elements), DOT



Example: SpMV on TP Grid

technische universität
dortmund

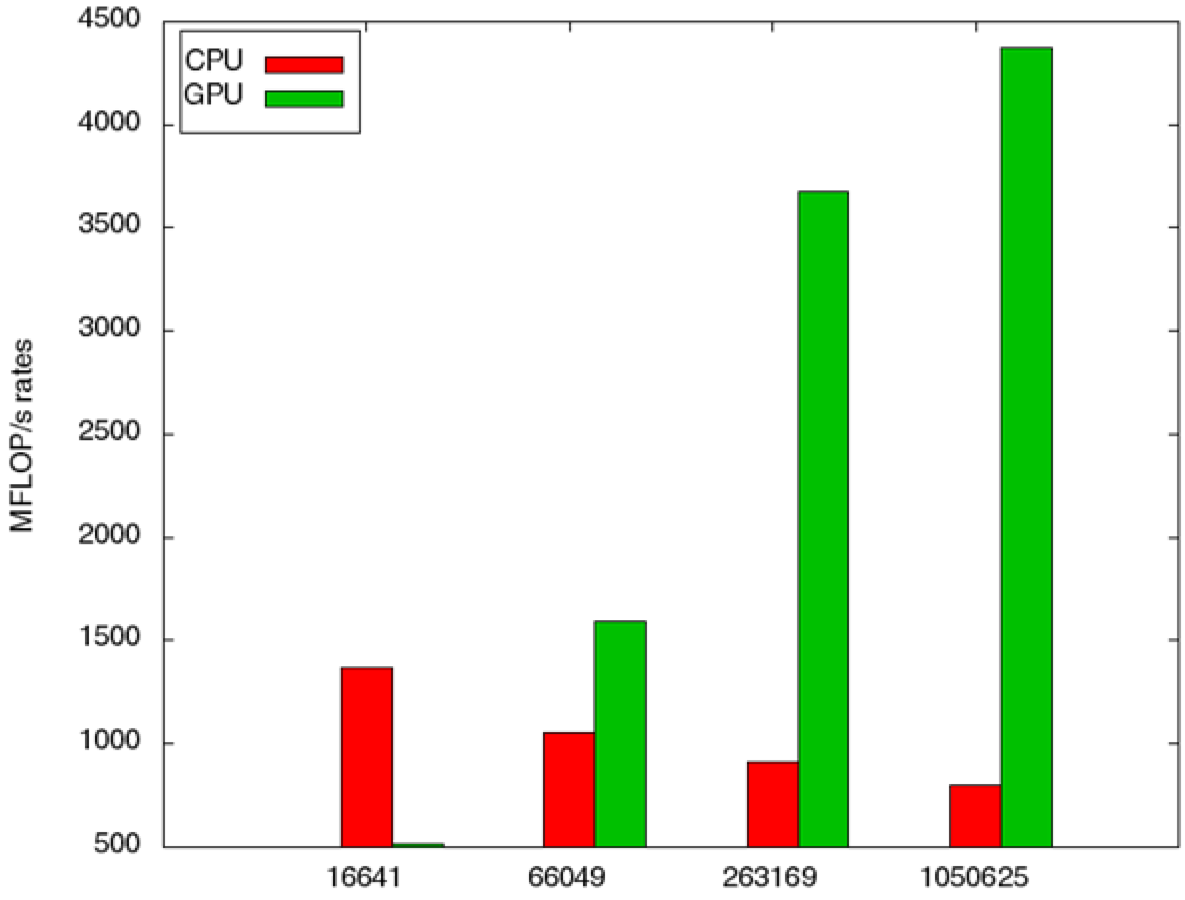


40 GFLOP/s, 140 GB/s with CUDA on GeForce GTX 280

'only' 13 GFLOP/s on 8800 GTX (90 GB/s peak)

Benchmarks: Complete Multigrid Solver

technische universität dortmund



Promising results,

attempt to integrate GPUs as FEM Co-Processors

Multigrid on TP Grid

technische universität dortmund

	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

Design Goals

Include GPUs into FEAST

- without
  - changes to application codes FEASTFLOW / FEASTSolid
  - fundamental re-design of FEAST
  - sacrificing either functionality or accuracy
- but with
  - noteworthy speedups
  - a reasonable amount of generality w.r.t. other co-processors
  - and additional benefits in terms of space/power/etc.

But: no --march=gpu/cell compiler switch

Integration Principles

technische universität dortmund

- **Isolate suitable parts**
  - Balance acceleration potential and acceleration effort
- **Diverge code paths as late as possible**
  - Local MG solver
  - Same interface for several co-processors
- **Important benefit of minimally invasive approach: No changes to application code**
  - Co-processor code can be developed and tuned on a single node
  - Entire MPI communication infrastructure remains unchanged

Minimally invasive integration

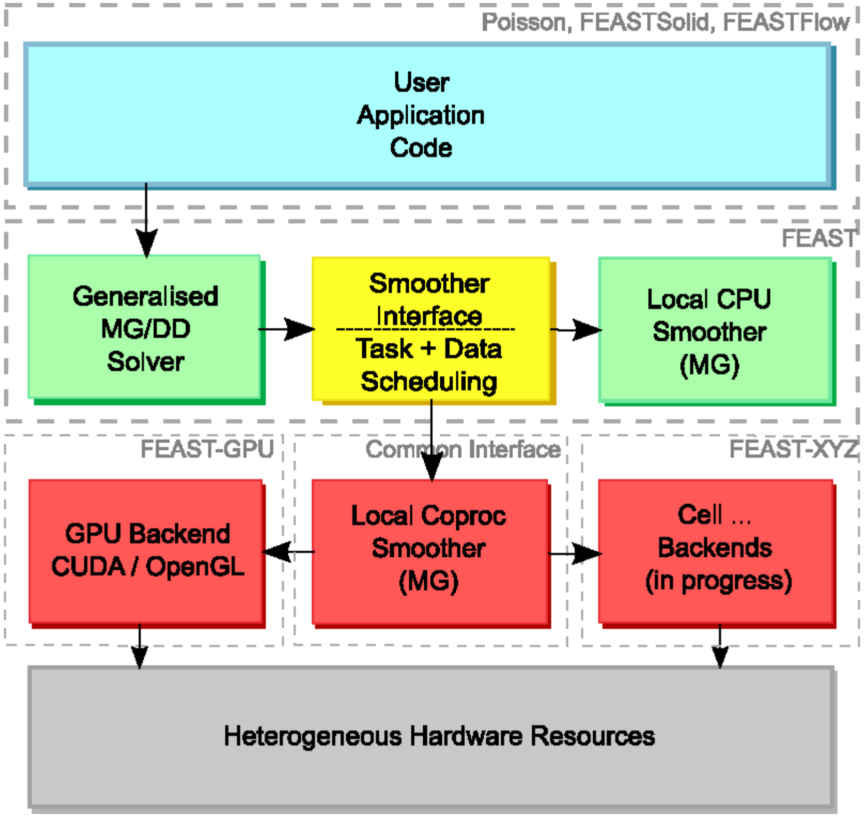
technische universität dortmund

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

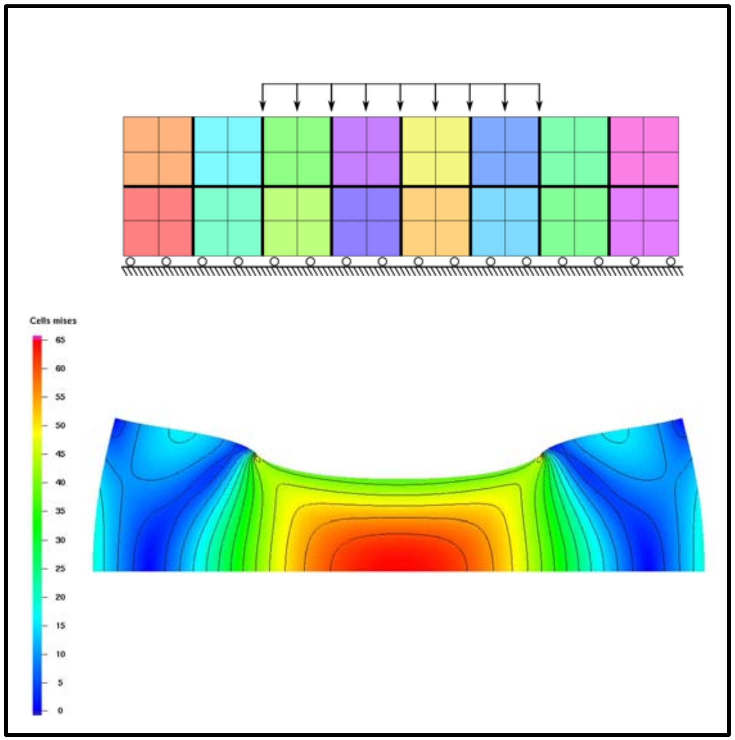
Applicable to many co-processors

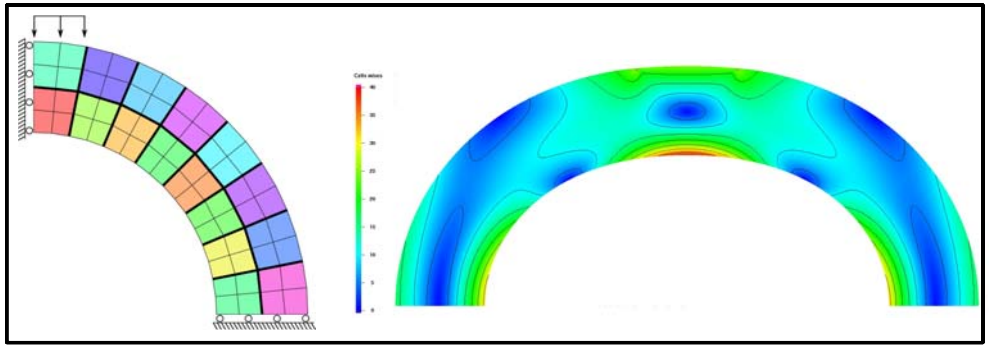


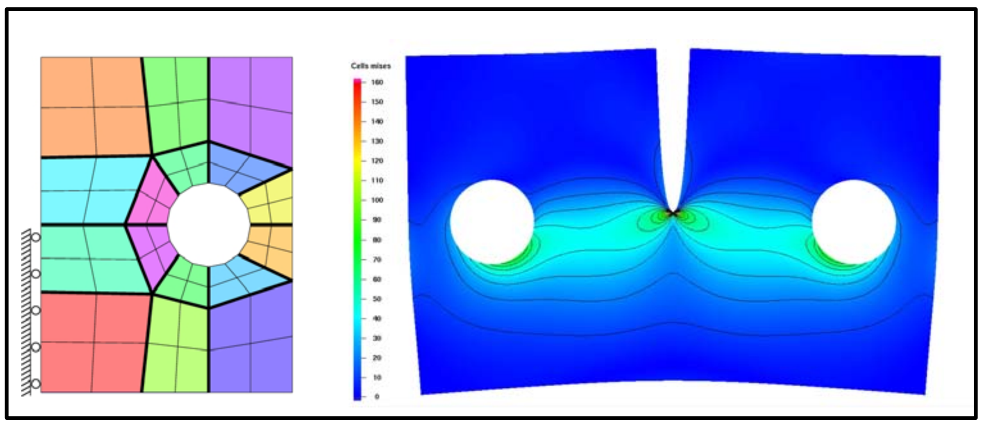
Show-Case: FEASTSolid

technische universität
dortmund

- Fundamental model problem:
  - solid body of elastic, compressible material (e.g. steel)
  - exposed to some external load







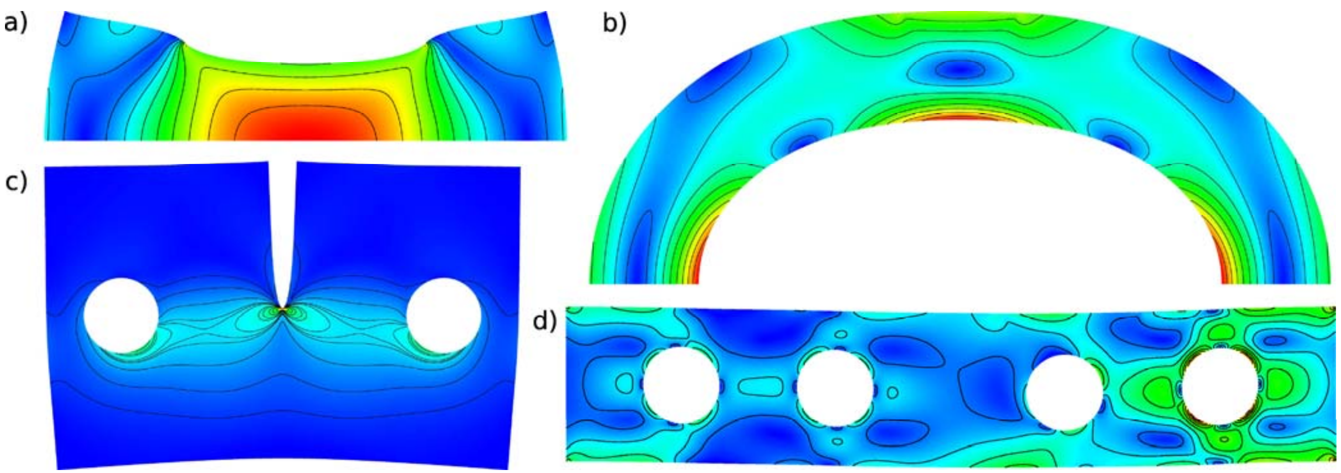
Linearised elasticity

technische universität dortmund

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

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

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



Mixed precision approach

technische universität dortmund

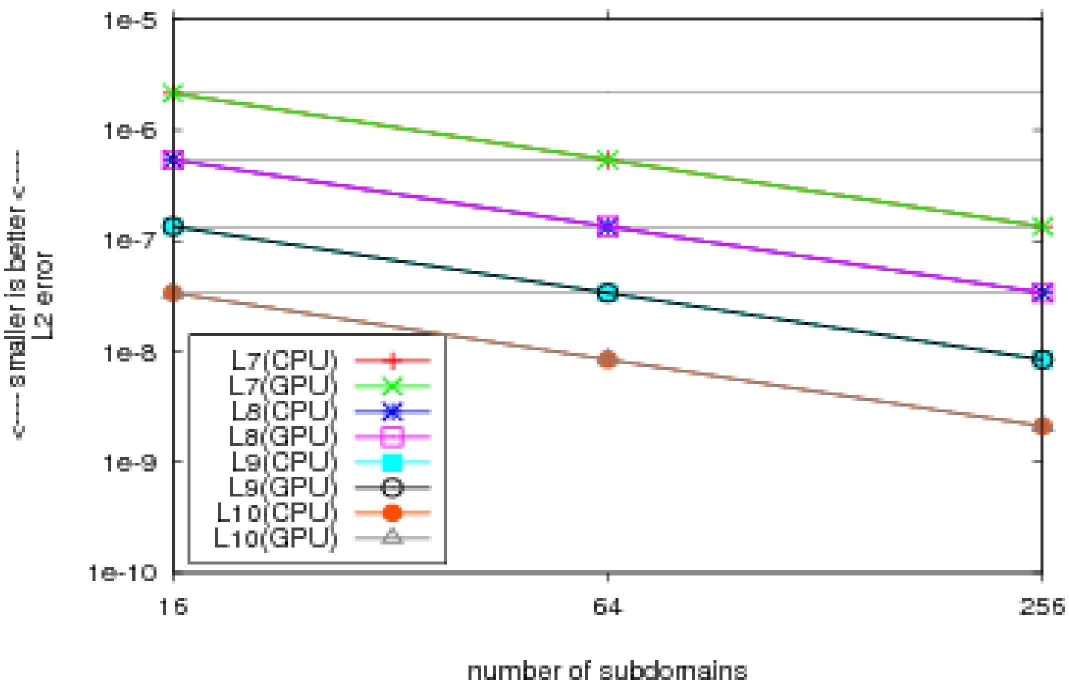
	single precision		double precision	
Level	Error	Reduction	Error	Reduction
2	2.391E-3		2.391E-3	
3	5.950E-4	4.02	5.950E-4	4.02
4	1.493E-4	3.98	1.493E-4	3.99
5	3.750E-5	3.98	3.728E-5	4.00
6	1.021E-5	3.67	9.304E-6	4.01
7	6.691E-6	1.53	2.323E-6	4.01
8	2.012E-5	0.33	5.801E-7	4.00
9	7.904E-5	0.25	1.449E-7	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
- ★ Mixed precision solver: double precision Richardson, preconditioned with single precision MG ('gain one digit')
- ★ Same results as entirely in double precision

Accuracy

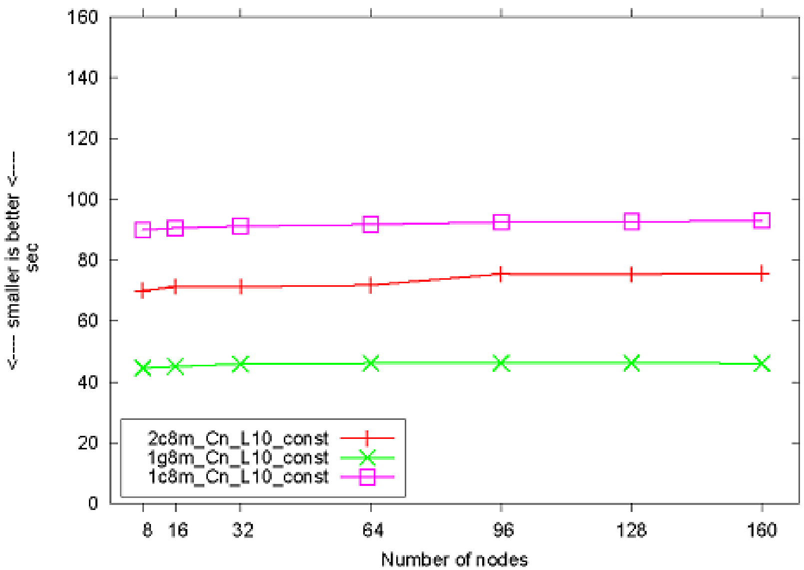
technische universität dortmund

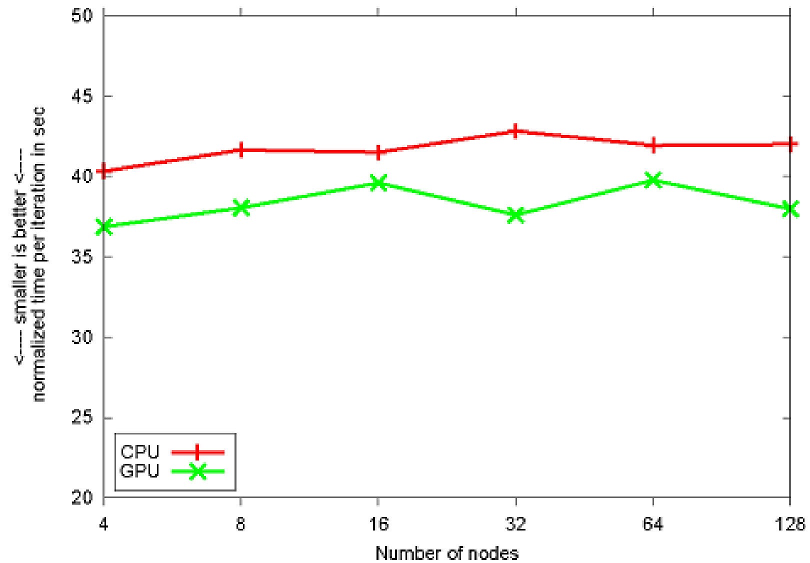
- L_2 error against reference solution



- Same results for CPU and GPU
  - expected error reduction independent of refinement and subdomain distribution

(Weak) Scalability

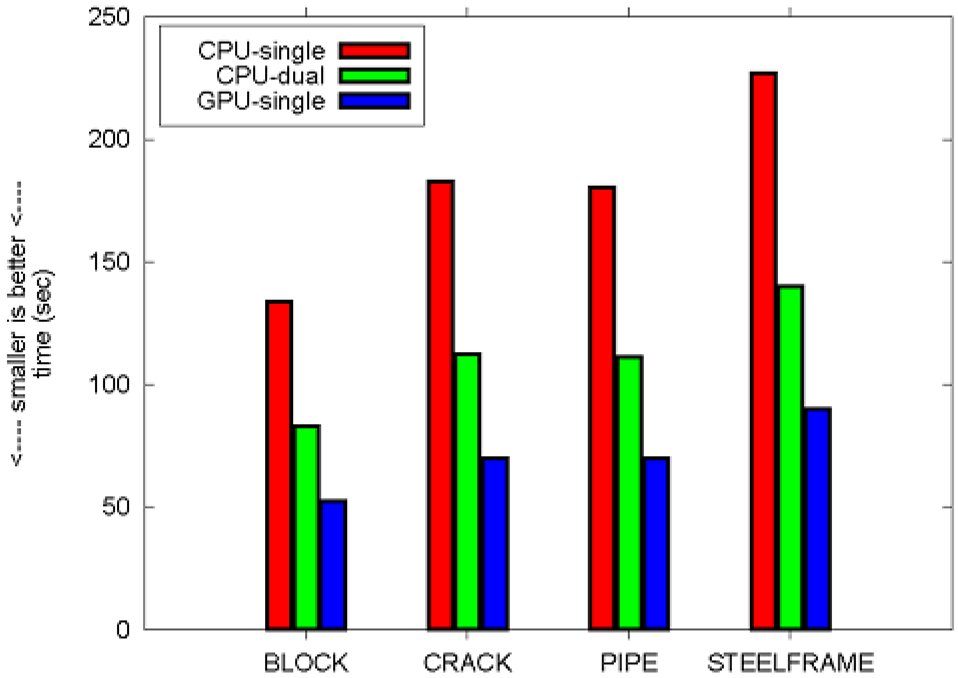




- ★ Outdated cluster, dual Xeon EM64T,
- ★ one NVIDIA Quadro FX 1400 per node (one generation behind the Xeons, 20 GB/s BW)
- ★ Poisson problem (left): up to 1.3 B DOF, 160 nodes
- ★ Elasticity (right): up to 1 B DOF, 128 nodes

Absolute Speedup

technische universität dortmund



- 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

- Speedups in 'time to solution' for one GPU:
  2.6x vs. Singlecore, 1.6x vs. Dualcore

- Amdahl's Law is lurking
  - Local speedup of 9x and 5.5x by the GPU
  - 2/3 of the solver accelerable => theoretical upper bound 3x

- Future work
  - Three-way parallelism in our system:
    - coarse-grained (MPI)
    - medium-grained (heterogeneous resources within the node)
    - fine-grained (compute cores in the GPU)
  - Better interplay of resources within the node
  - Adapt Hardware-oriented Numerics to increase accelerable part

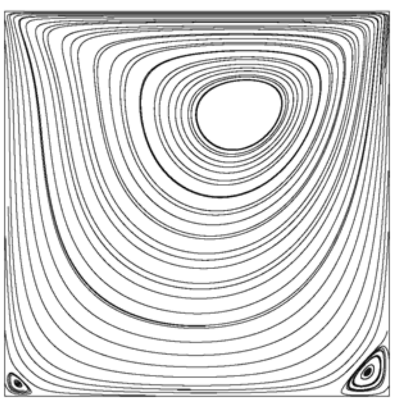
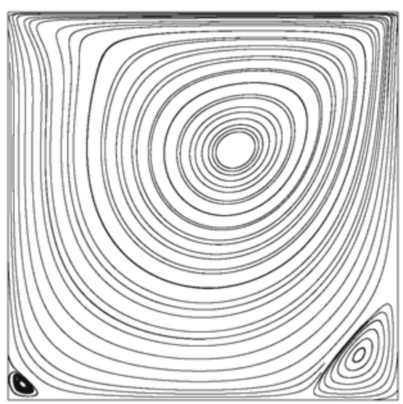
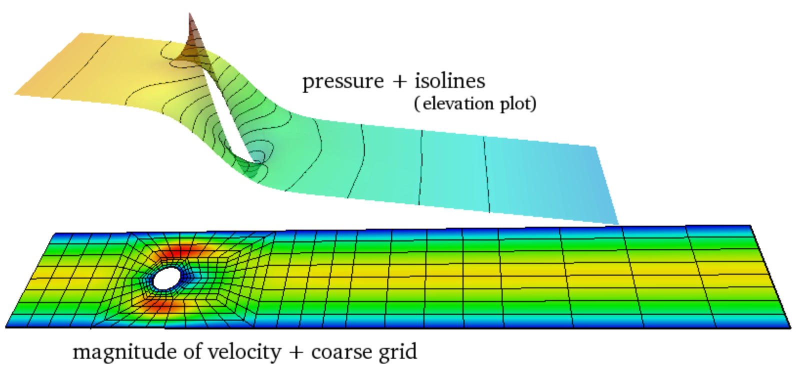
Stationary Navier-Stokes

technische universität dortmund

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

- ★ 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 u_1 with
   local MG
     for all Ω_i: solve for u_2 with
   local MG
   2) update RHS: $d_3 = -d_3 + B(c_1, c_2)$
   3) scale $c_3 = (M_p^{\mathrm{L}})d_3$

Navier-Stokes results

technische universität dortmund

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

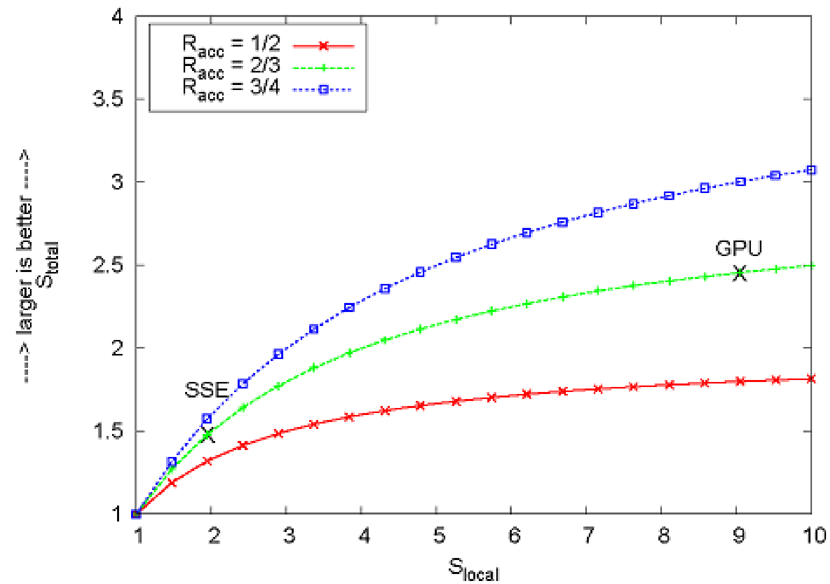
DC Re100		DC Re250		Channel flow	
plain	accel.	plain	accel.	plain	accel.
29:71	50:48	11:89	25:75	13:87	26:74

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



There is a Huge Potential for the Future …

But:

- **High Performance Computing** has to consider recent and future hardware trends, particularly for heterogeneous multicore architectures and massively parallel systems!

- The combination of **'Hardware-oriented Numerics'** and special **'Data Structures/Algorithms'** and **'Unconventional Hardware'** has to be used!

…or most of existing (academic/commercial) FEM software will be 'worthless' in a few years!

Acknowledgements

- FEAST Group
  (TU Dortmund)

- Robert Strzodka
  (Max Planck Center, Max Planck Institut Informatik)

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