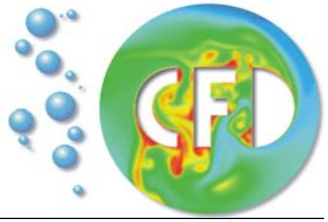
technische universität
dortmund

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

- 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 CFD and CSM.

Aim of this Talk

High Performance Computing

meets

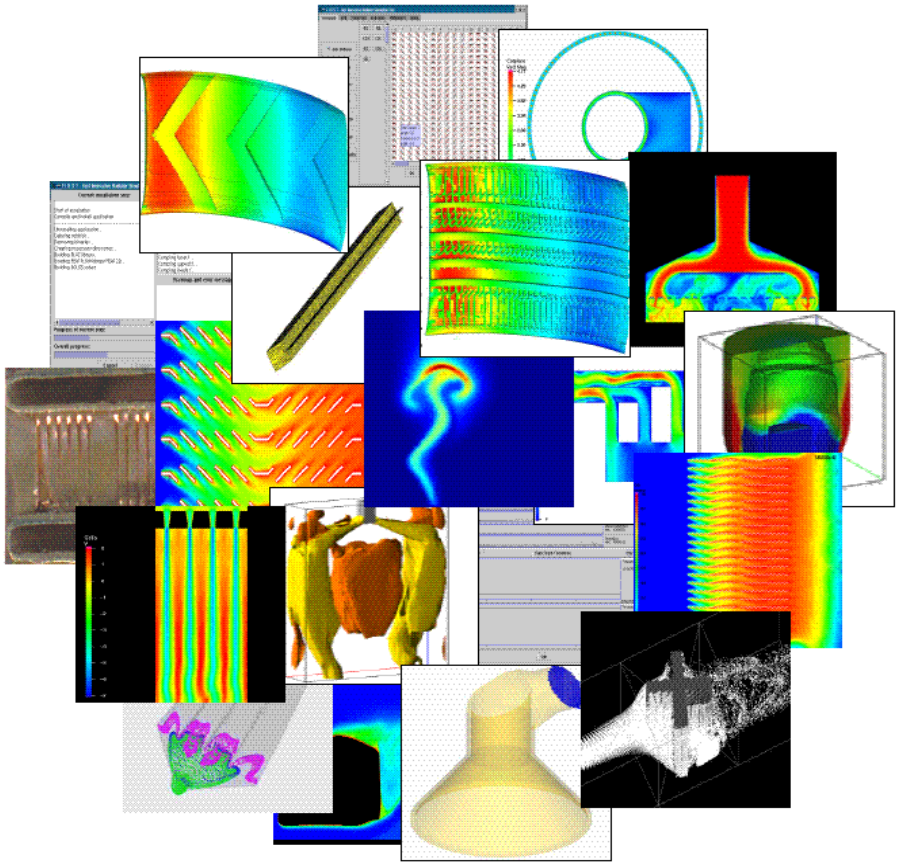
Hardware-oriented Numerics

on

Unconventional Hardware

for

Finite Element Methods



I) Hardware-Oriented Numerics

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

- '**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

- **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

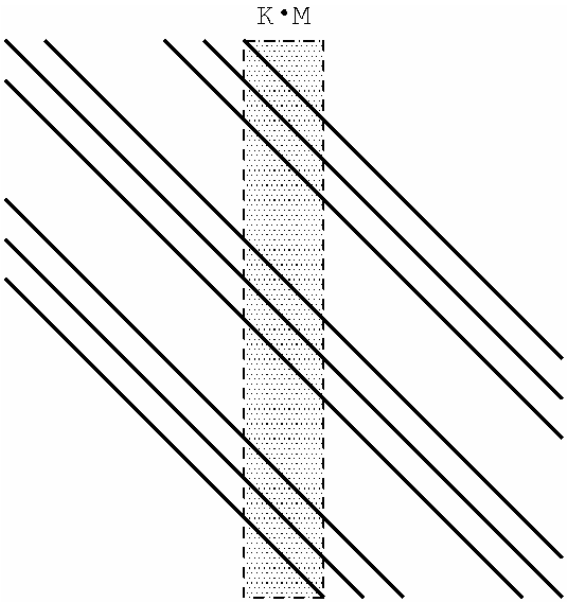
➔ **More realistic (and much harder) 'Criterion' for Petascale Computing in Technical Simulations**

Main Component: ‘Sparse’ MV Application

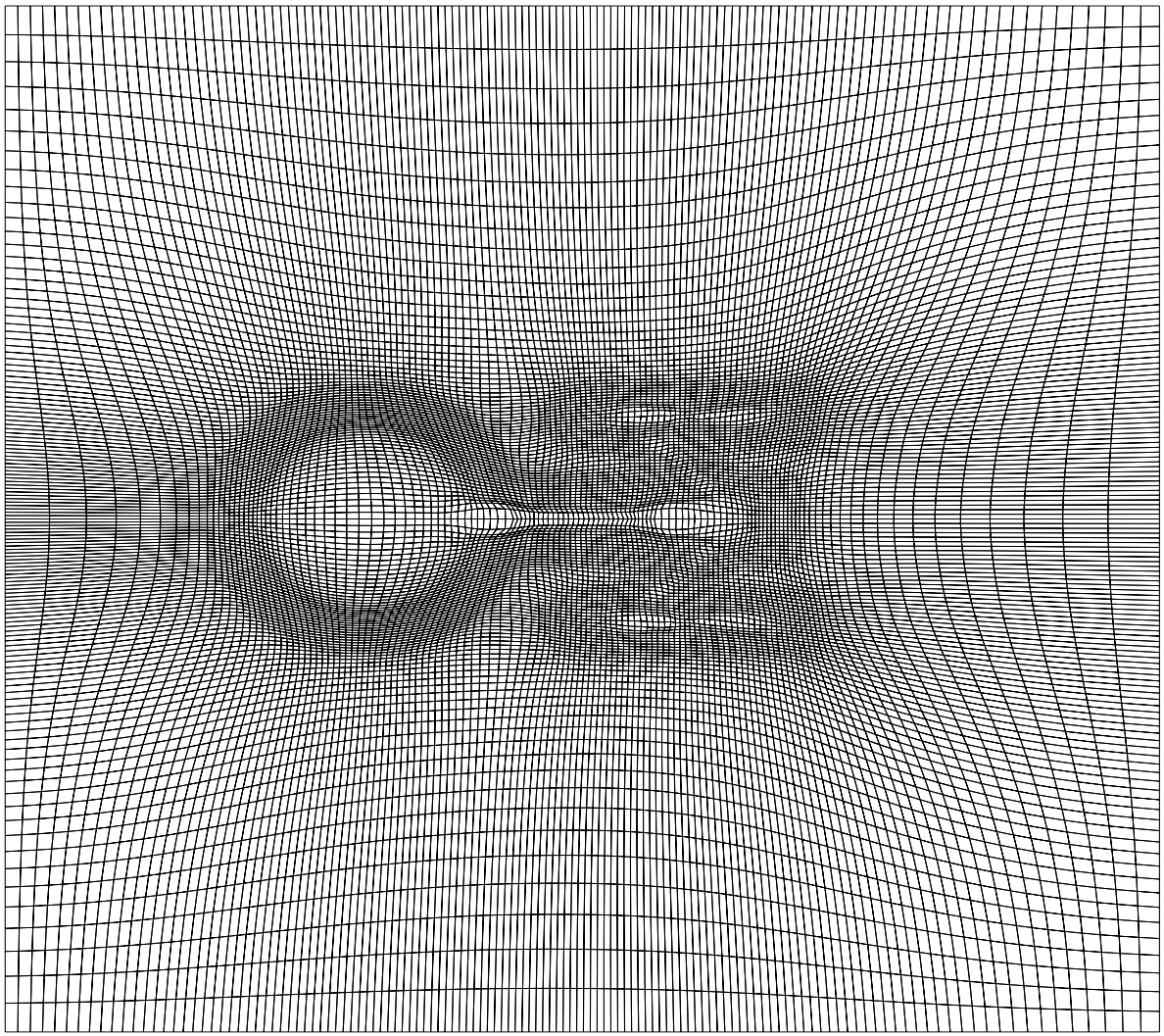
- Sparse **Matrix-Vector techniques** (‘indexed DAXPY’)

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

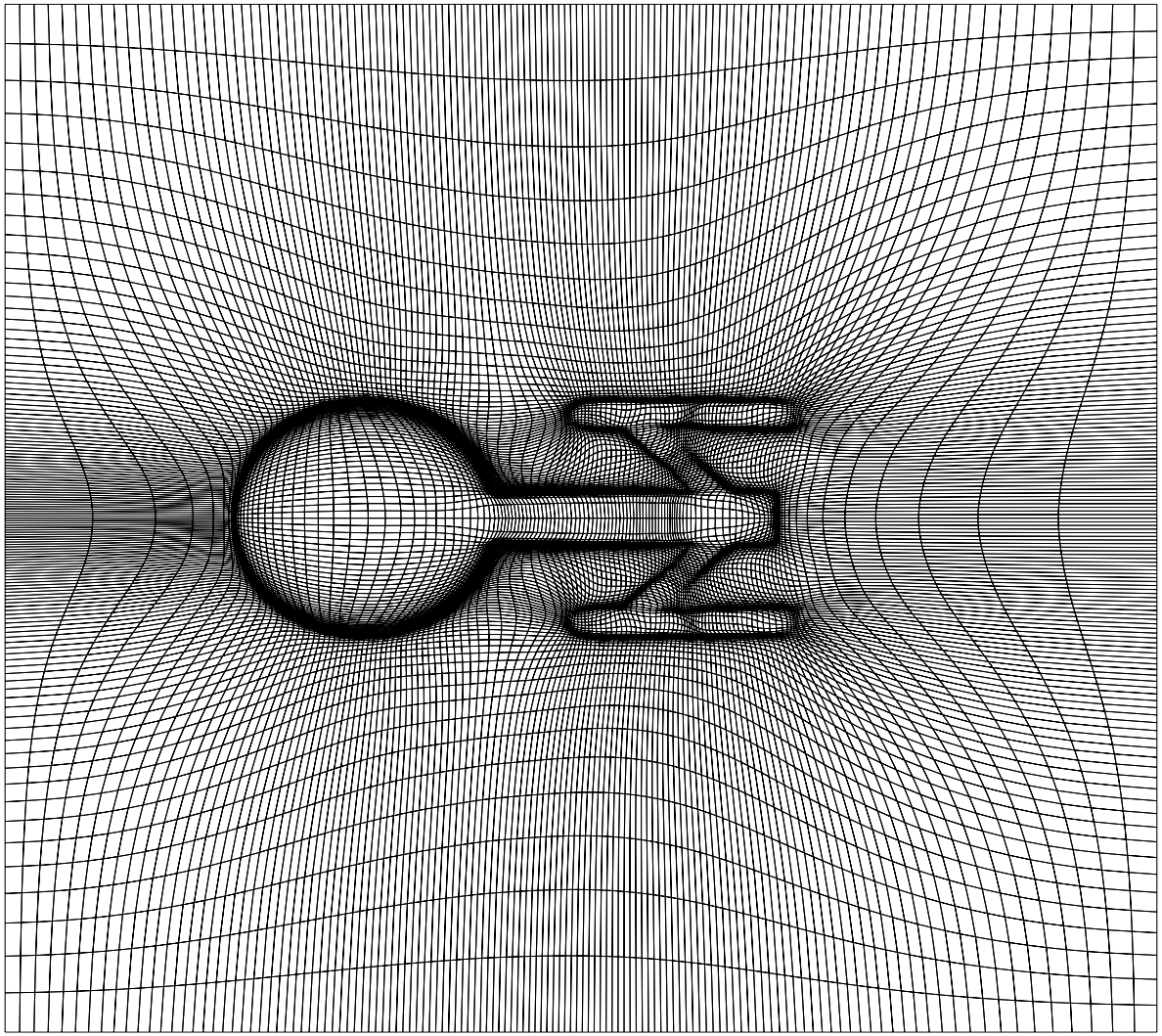
- Sparse Banded **MV techniques** on **generalized TP grids**



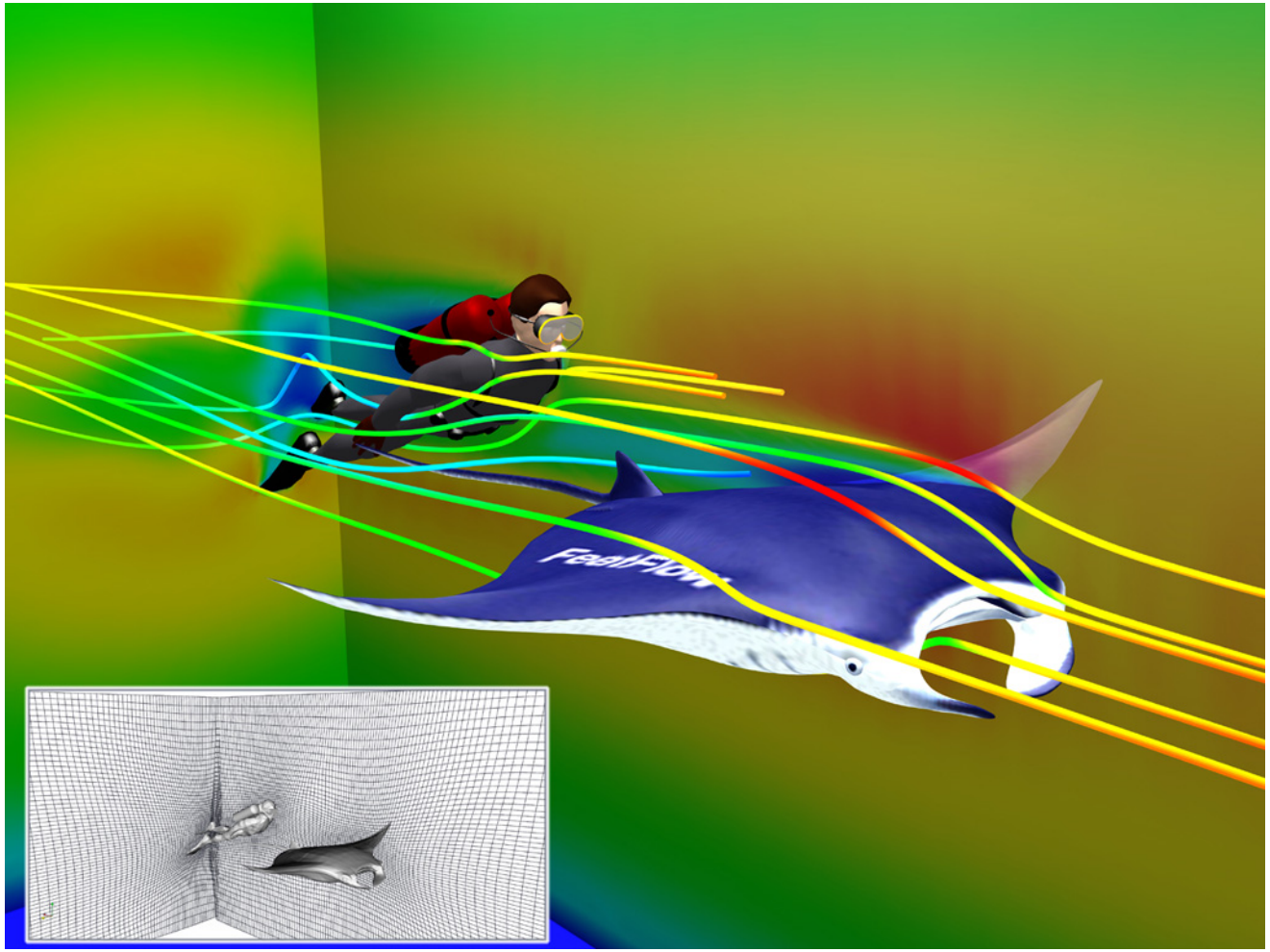
Generalized Tensorproduct Meshes



Generalized Tensorproduct Meshes



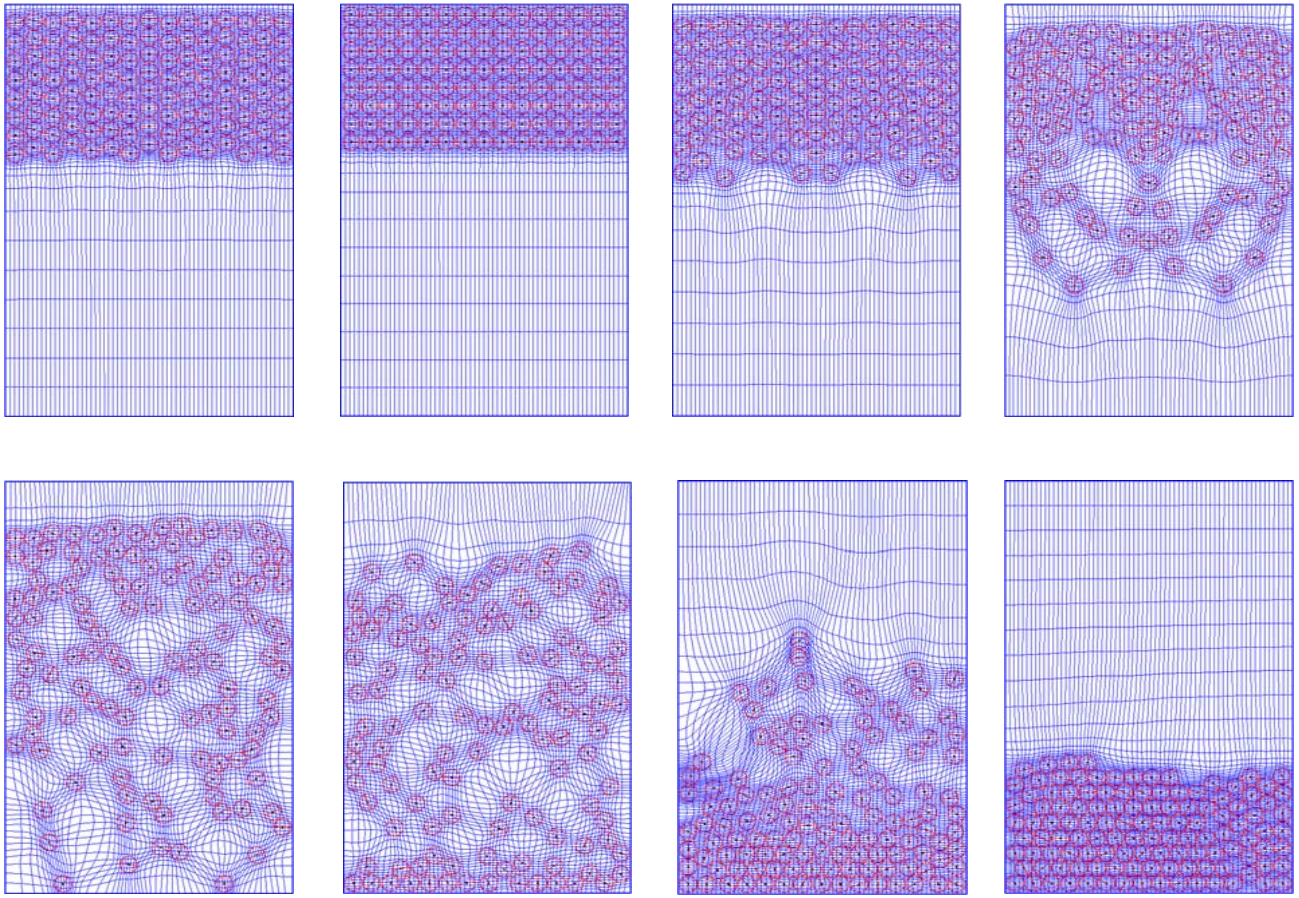
Generalized Tensorproduct Meshes

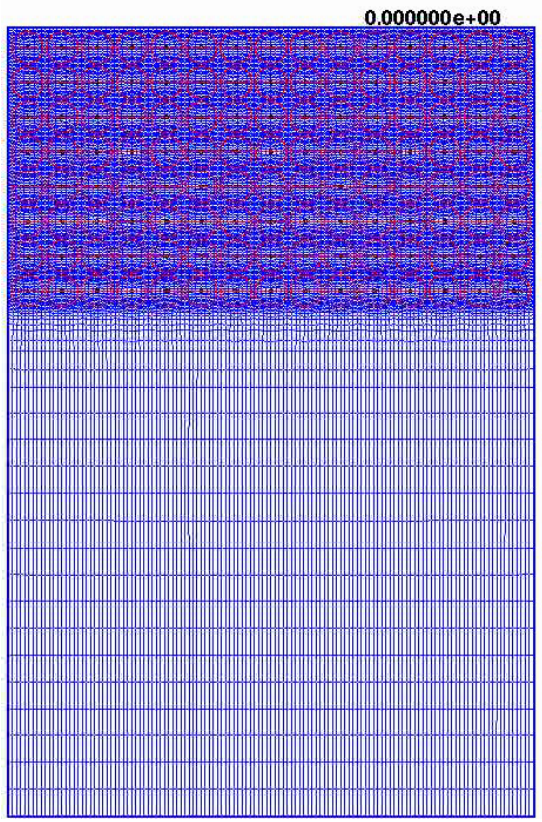


…with appropriate Fictitious Boundary techniques in FEATFLOW…..

Generalized Tensorproduct Meshes

….dynamic CFD problems…..





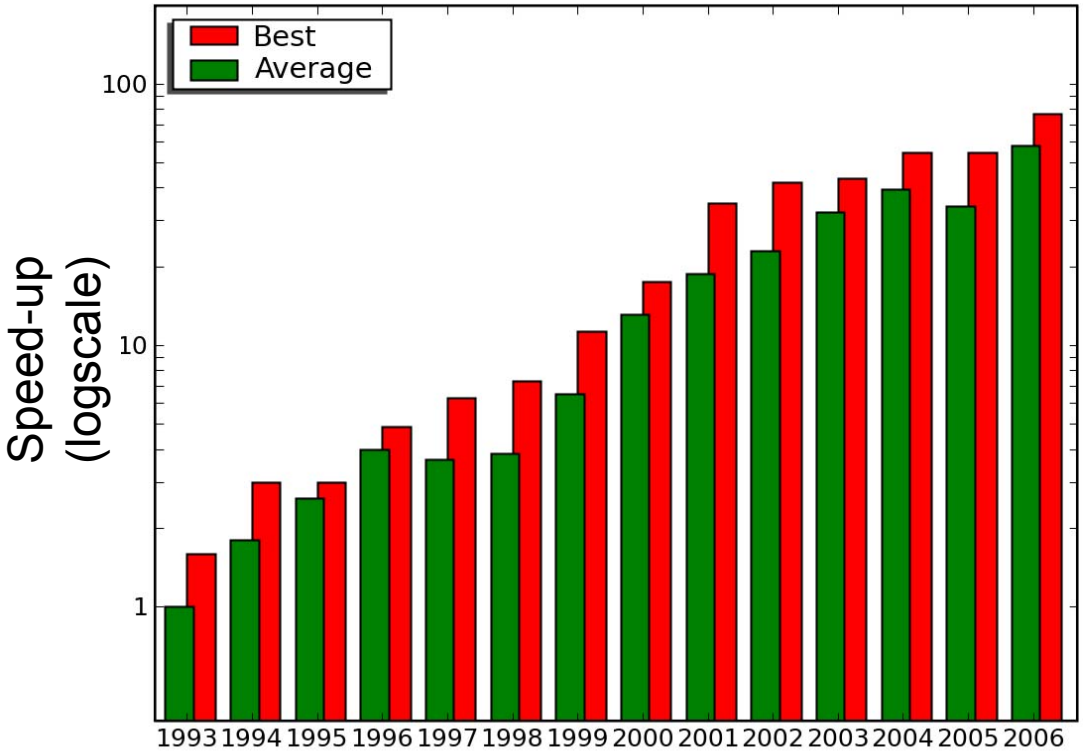
Observation I: Sparse MV Multiplication

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

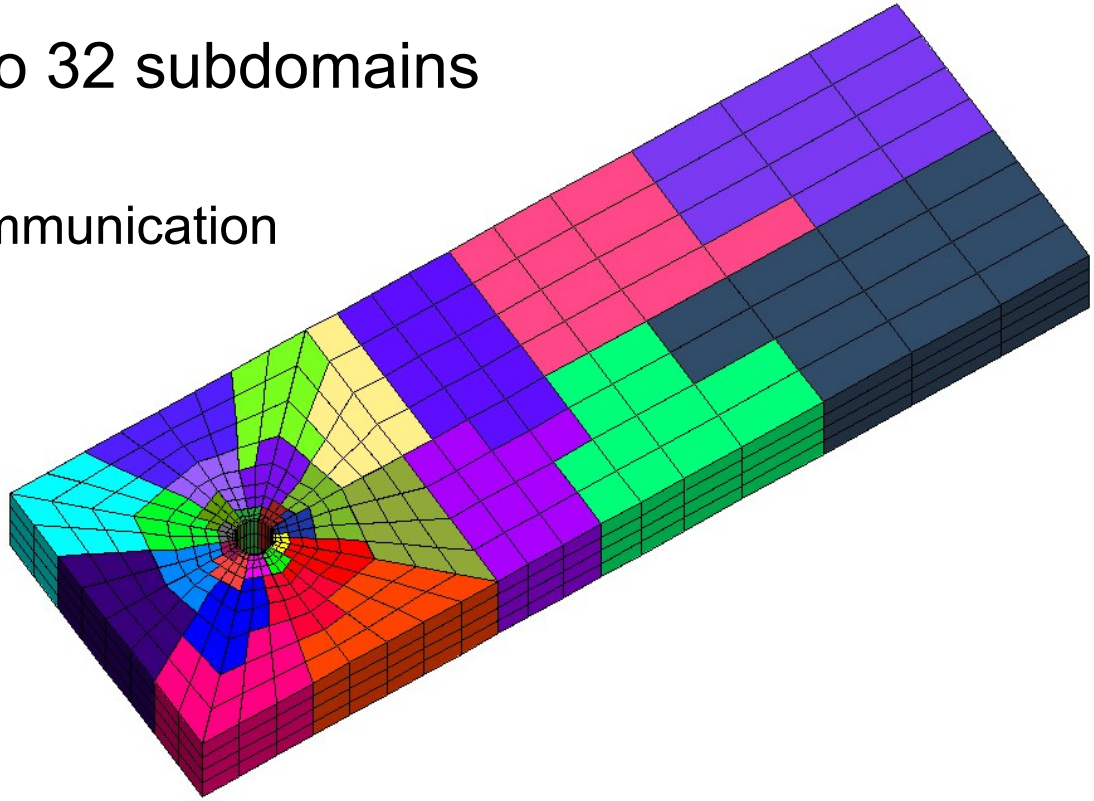


Speed-up of 100x for free in 10 years

Stagnation for standard simulation tools
on conventional hardware

Observation III: Parallel Performance

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



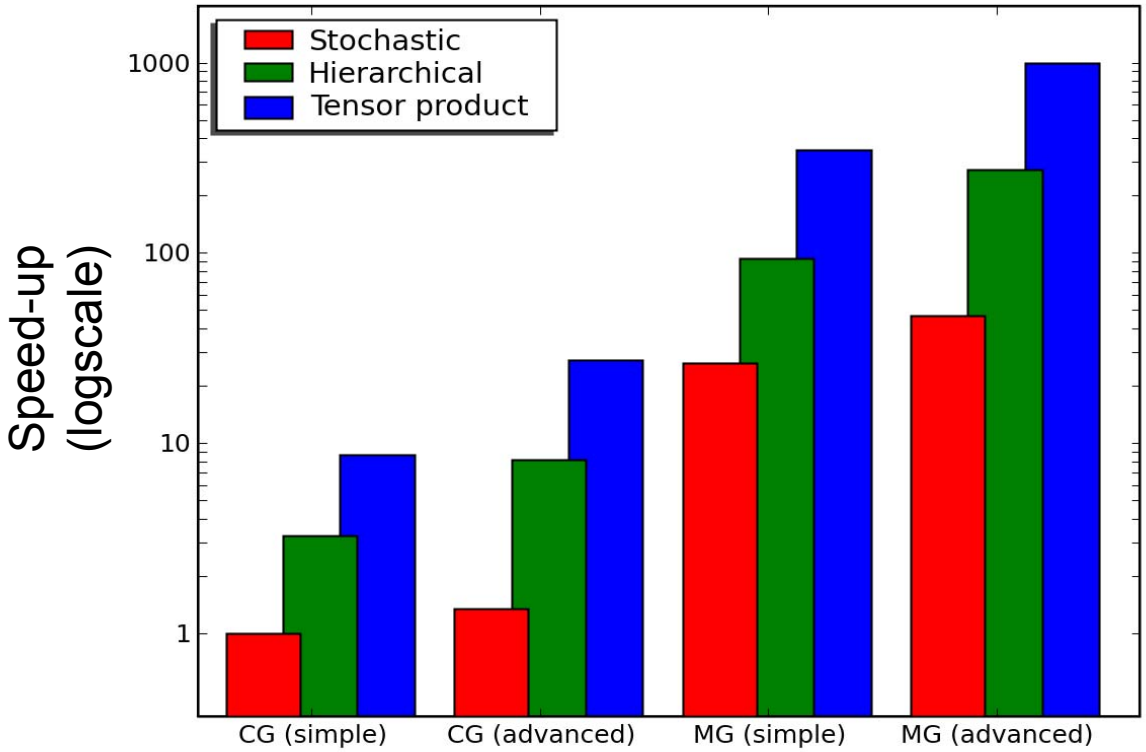
	1 P.	2 P.	4 P.	8 P.	16 P.	32 P.	64 P.
%Comm.	10%	24%	36%	45%	47%	55%	56%
# PPP-IT	2.2	3.0	3.9	4.9	5.2	5.7	6.2

Summary

- 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)

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

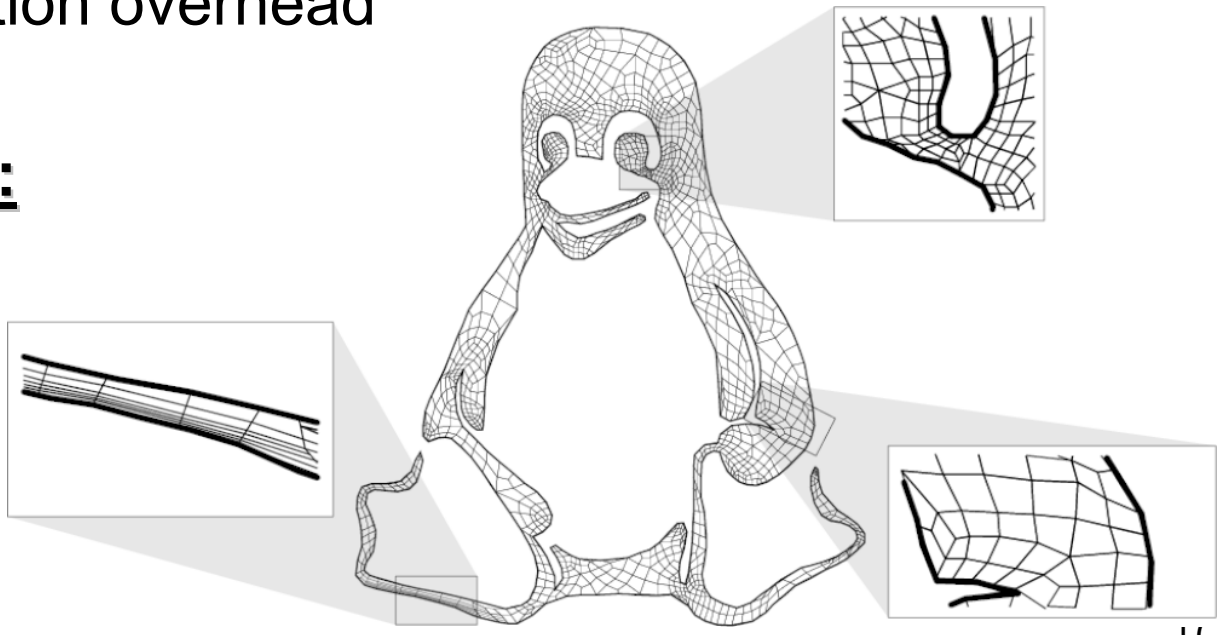


Dramatic improvement (factor 1000) due to **better Numerics** AND **better data structures/ algorithms**

FEAST – Realization of HwoN

- **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)



(Preliminary) State-of-the-Art

- **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

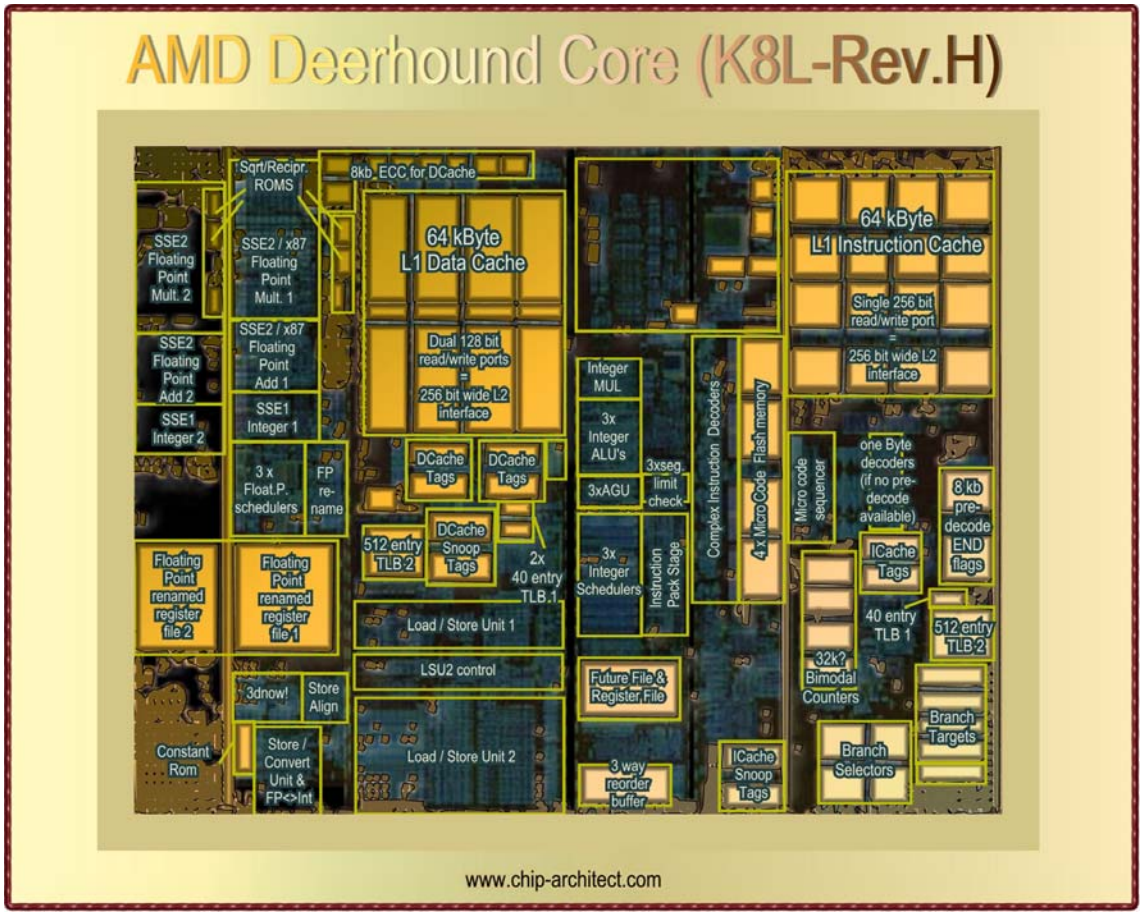
II) UnConventional HPC

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

- Graphics Processor (GPU):
  128 parallel scalar processors
  @ 1.35 GHz, 900 MHz GDDR3
  memory (86.4 GB/s)
  ≈ 350 GFLOP/s

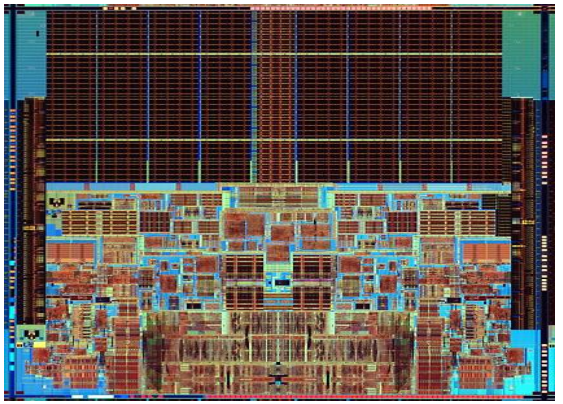
UnConventional High Performance Computing (UCHPC)

Why are GPUs and Cells so fast?



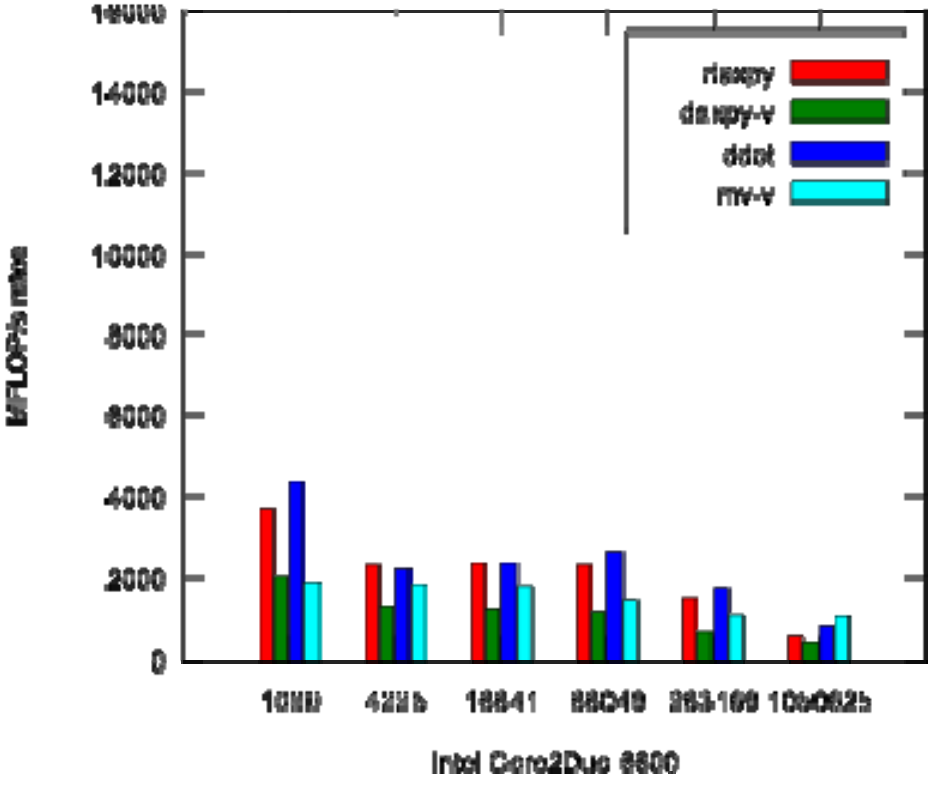
CPUs devote most of the transistors to caches and data movement for general purpose applications

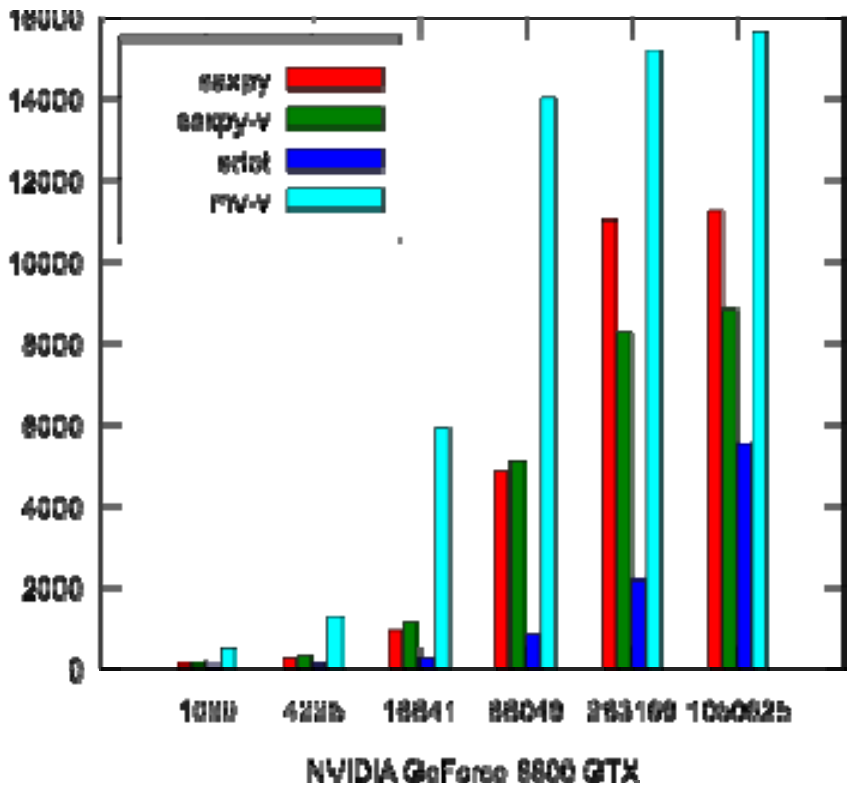
GPUs and **Cells** are more "transistor-efficient" w.r.t. floating point operations



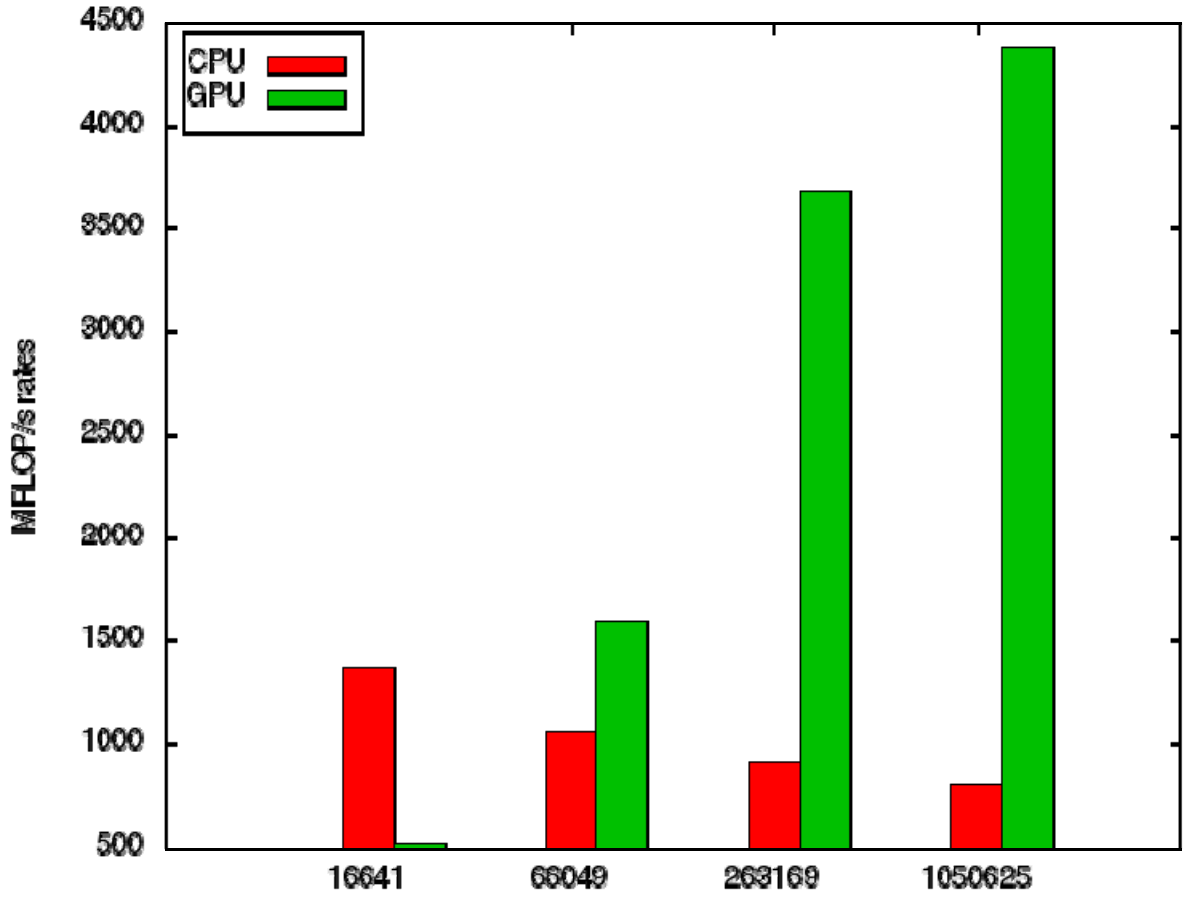
Benchmarks: FEM Building Blocks

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





Benchmarks: Complete Multigrid Solver



Promising results,
attempt to integrate GPUs as FEM Co-Processors

Design Goals

Include GPUs into FEAST

- without
  - changes to application codes FEASTFLOW / FEASTSolid
  - fundamental re-design of FEAST
  - sacrificing either functionality or accuracy

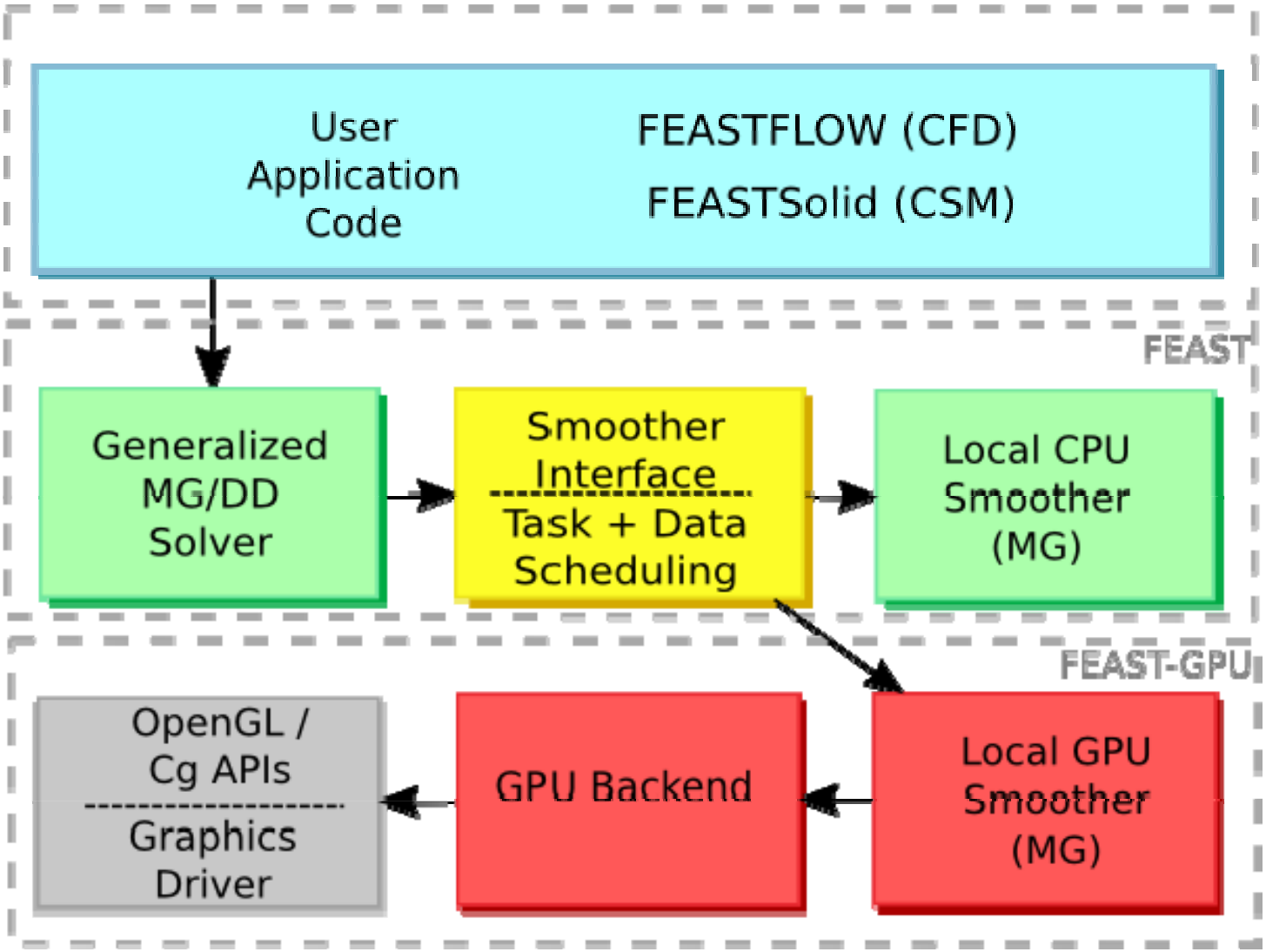
- but with
  - noteworthy speedups
  - a reasonable amount of generality w.r.t. other co-processors
  - and additional benefits in terms of space/power/etc.

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

Integration Principles

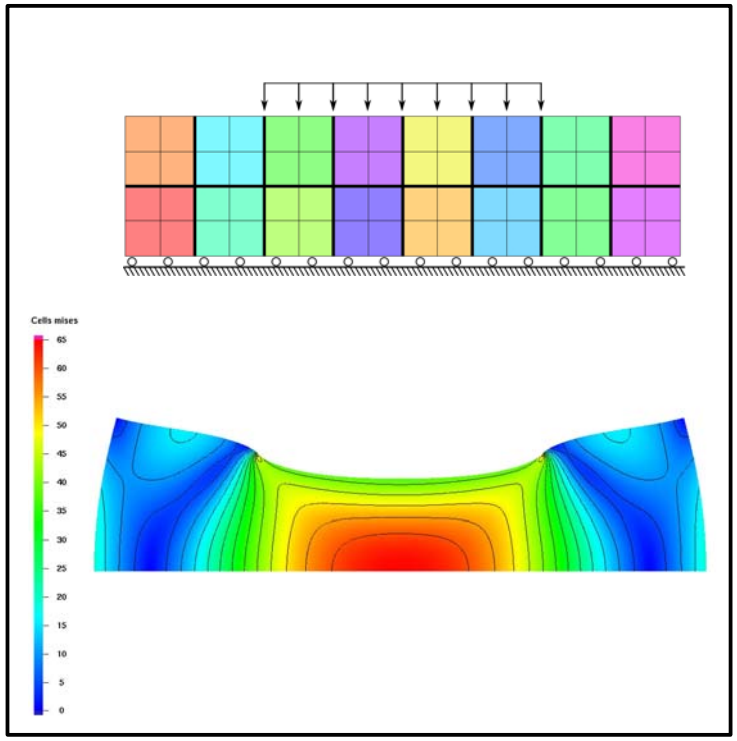
- **Isolate suitable parts**
  - Balance acceleration potential and acceleration effort
- **Diverge code paths as late as possible**
  - Local MG solver
  - Same interface for several co-processors
- **Important benefit of minimally invasive approach: No changes to application code**
  - Co-processor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

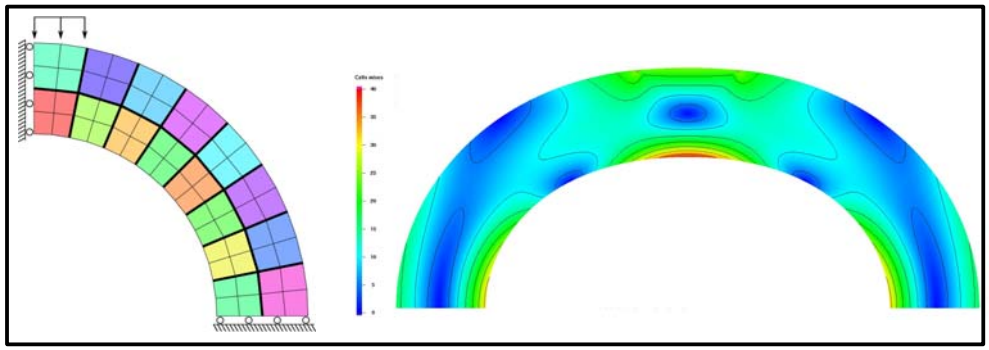
Integration Overview

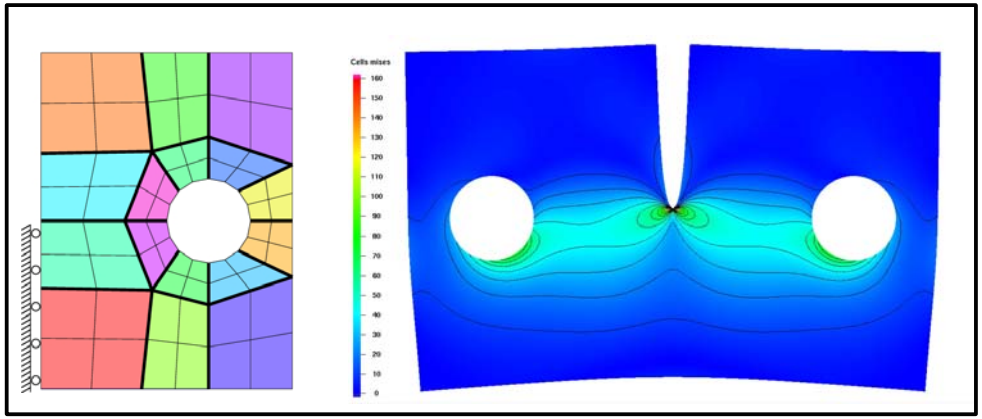


Show-Case: FEASTSolid

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

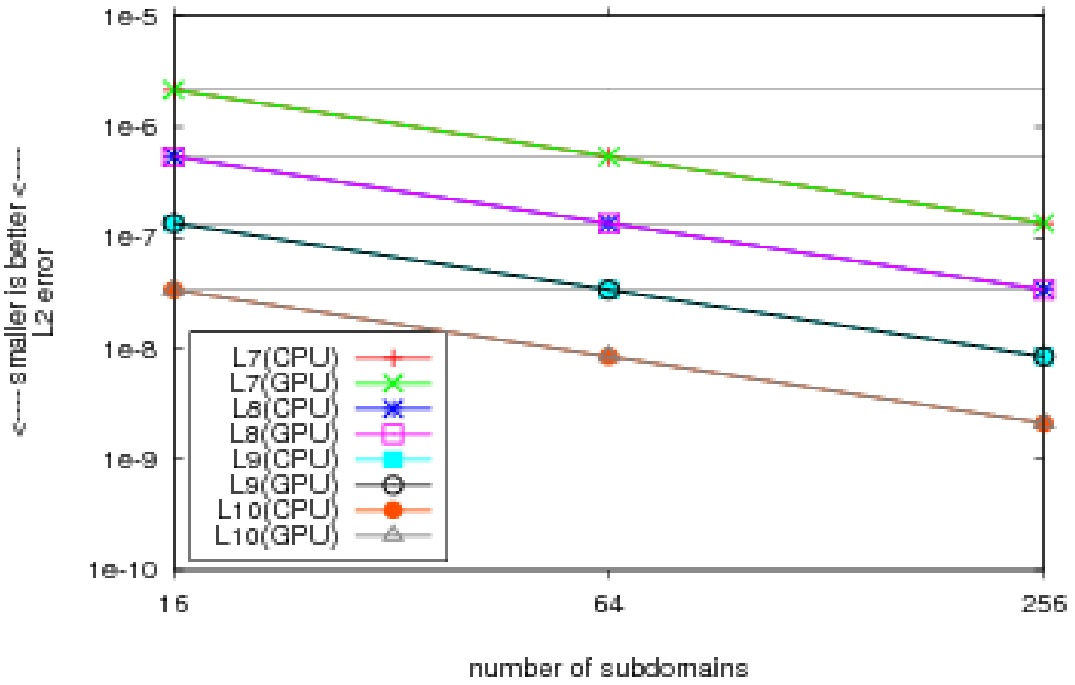






Accuracy

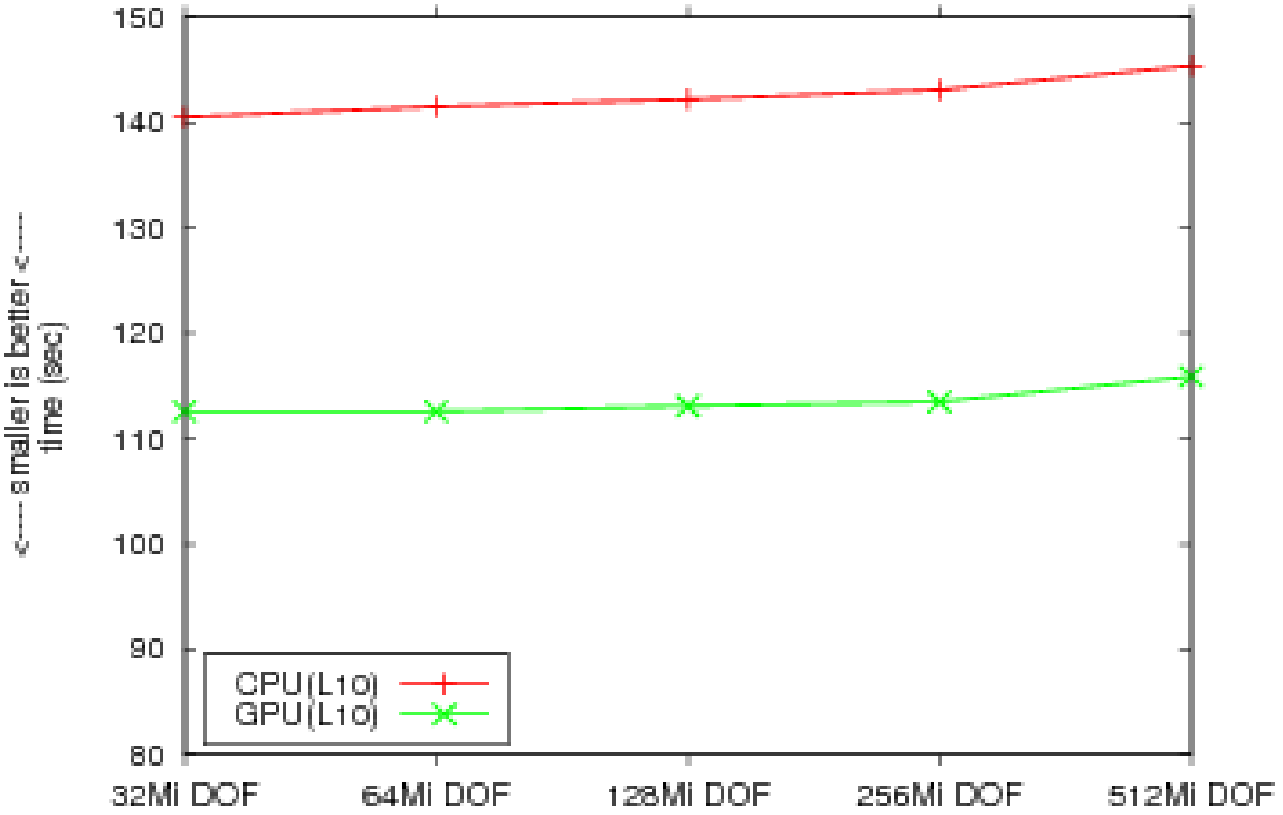
- L_2 error against reference solution



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

(Weak) Scalability

64 nodes,
2x Xeon EM64T
3.4 GHz,
1x outdated
Quadro 1400 GPU,
Infiniband

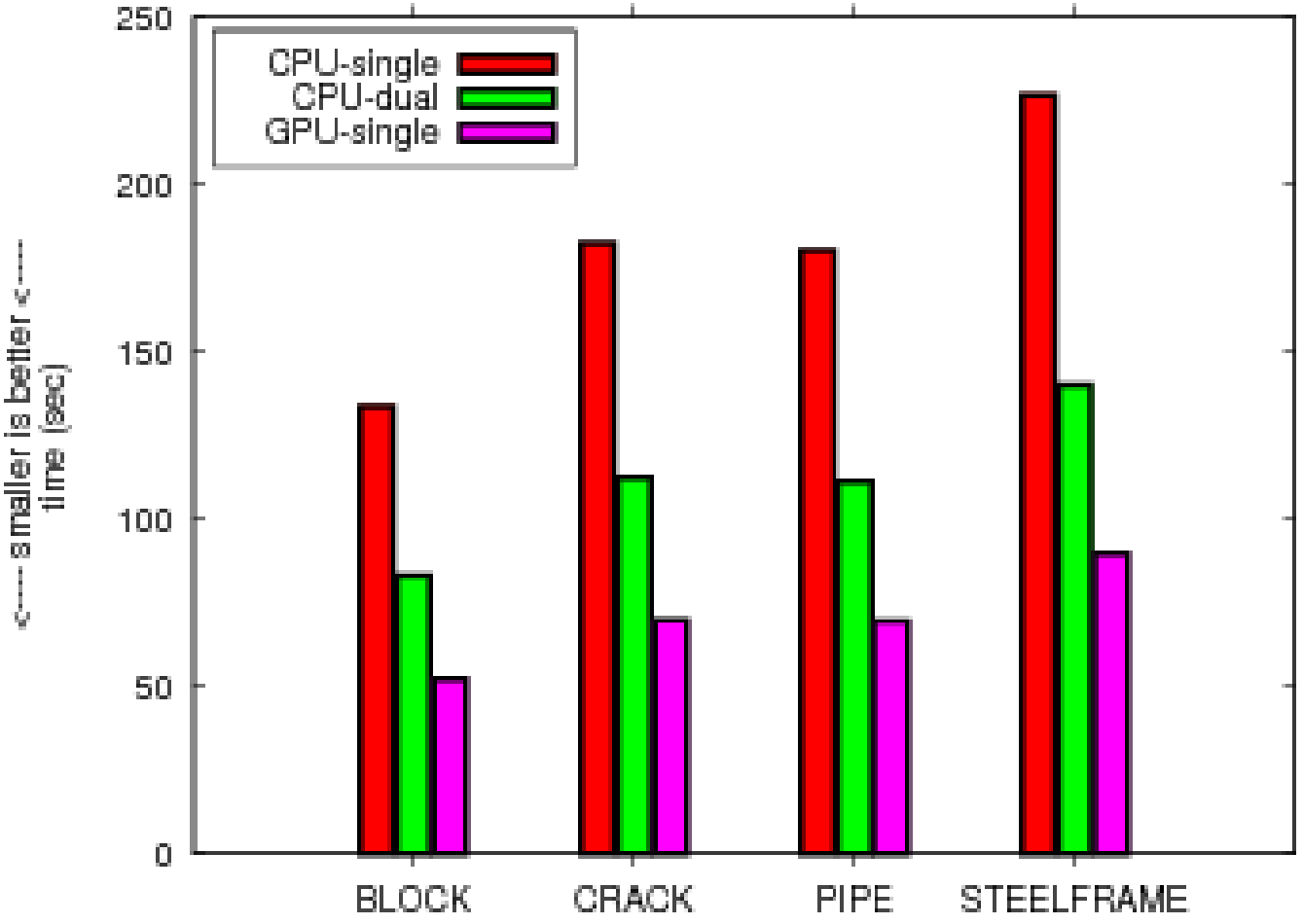


More results

- Poisson problem for 1.3 billion unknowns in less than 50 seconds on 160 outdated GPUs (Quadro 1400)

Speedup

16 nodes,
2x Santa Rosa dualcore CPUs,
1x Quadro 5600 GPU,
Infiniband



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

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)