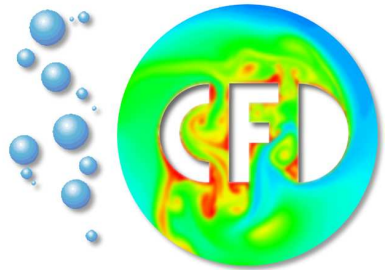
High Performance Computing for PDE Towards Petascale Computing

S. Turek, D. Göddeke
with support by: Chr. Becker, S. Buijssen, M. Grajewski, H. Wobker

Institut für Angewandte Mathematik, Univ. Dortmund
http://www.mathematik.uni-dortmund.de/LS3
http://www.featflow.de

January 30, 2007



Aim of this Talk

Overall Aim:

'High Performance Computing'

meets

'Hardware-Oriented Numerics for PDE'

Hardware-Oriented Numerics

What is:

Hardware-Oriented Numerics for PDE ?

It is more than "good Numerics" and "good Implementation" together with High Performance Computing techniques !

Critical quantity: 'Total Numerical Efficiency !'

Answer

What is the "Total Numerical Efficiency" for the computational simulation of PDE ?

'High (guaranteed) **accuracy** *for user-specific quantities with minimal #d.o.f. ($\sim N$) via fast and robust* **solvers** *– for a wide class of parameter variations – with 'optimal' ($\sim O(N)$)* **numerical complexity** *while exploiting a significant percentage of the available huge sequential/parallel* **GFLOP/s** *rates at the same time.'*

Mathematical Key Technologies

'A posteriori error control/adaptive meshing'

'Iterative (parallel) solution strategies'

'Operator-splitting for coupled problems'

But: *How to achieve a high "Total Numerical Efficiency" ?*

For iterative solvers + adaptive discretizations ?

Example

Example: Fast Multigrid Solvers

'Optimized' versions for scalar PDE problems
($\approx$ Poisson problems) on general meshes
should require 100 - 1000 FLOPs per unknown

Problem size 10^6: Much less than 1 sec on PC!

Problem size 10^{12}: Less than 1 sec on PFLOP/s computer!

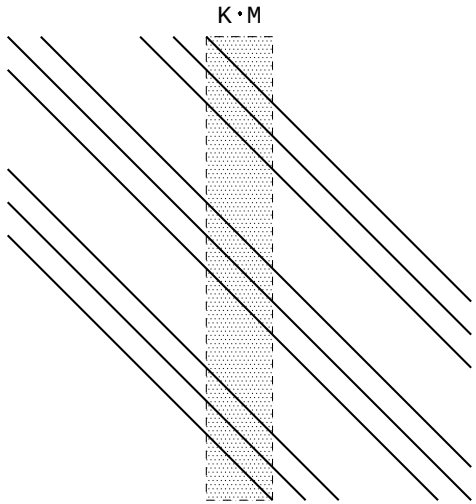
'Criterion' for Petascale Computing

Main Component: 'Sparse' MV Application

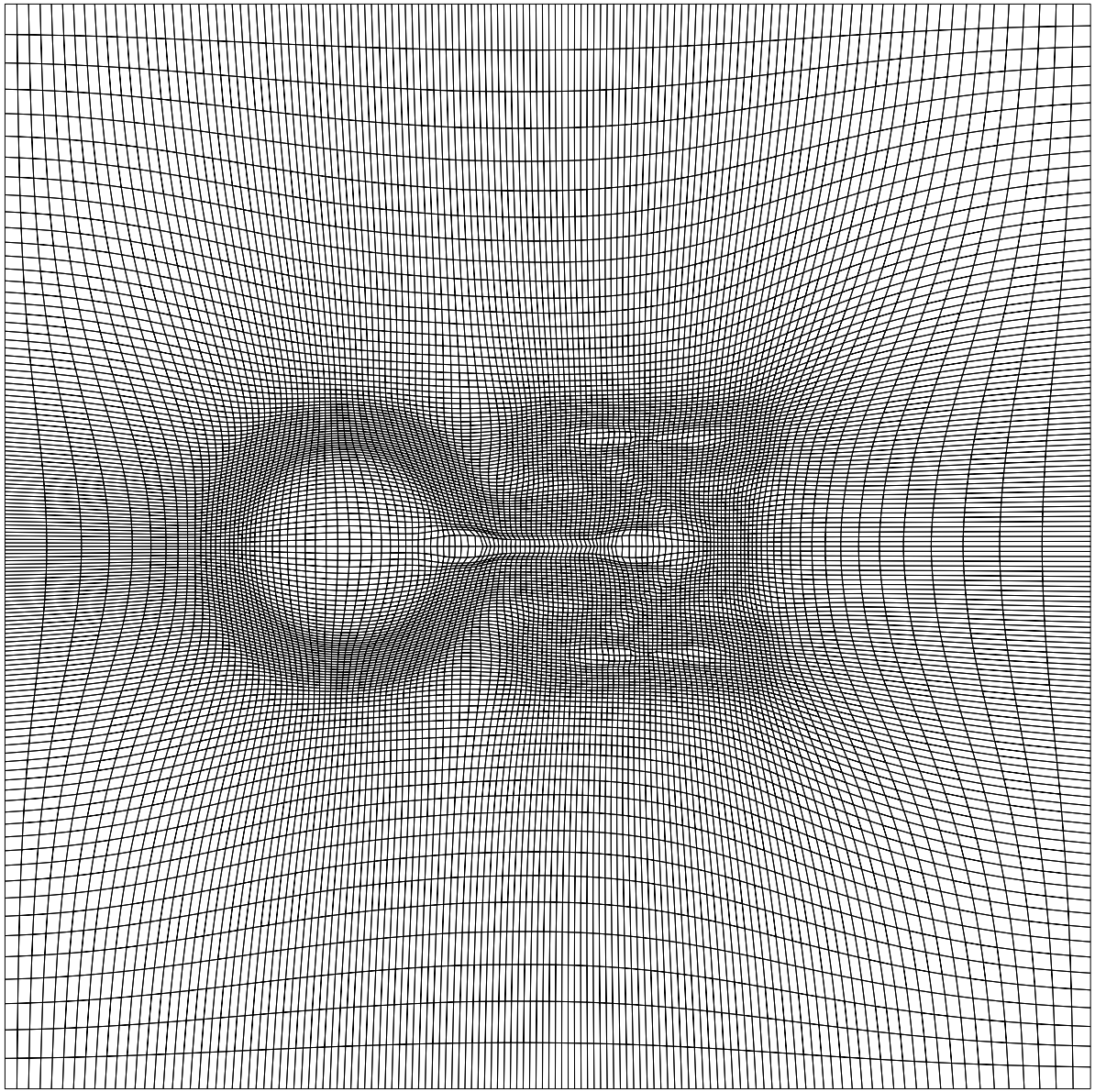
Sparse **Matrix-Vector techniques** ('indexed DAXPY')

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

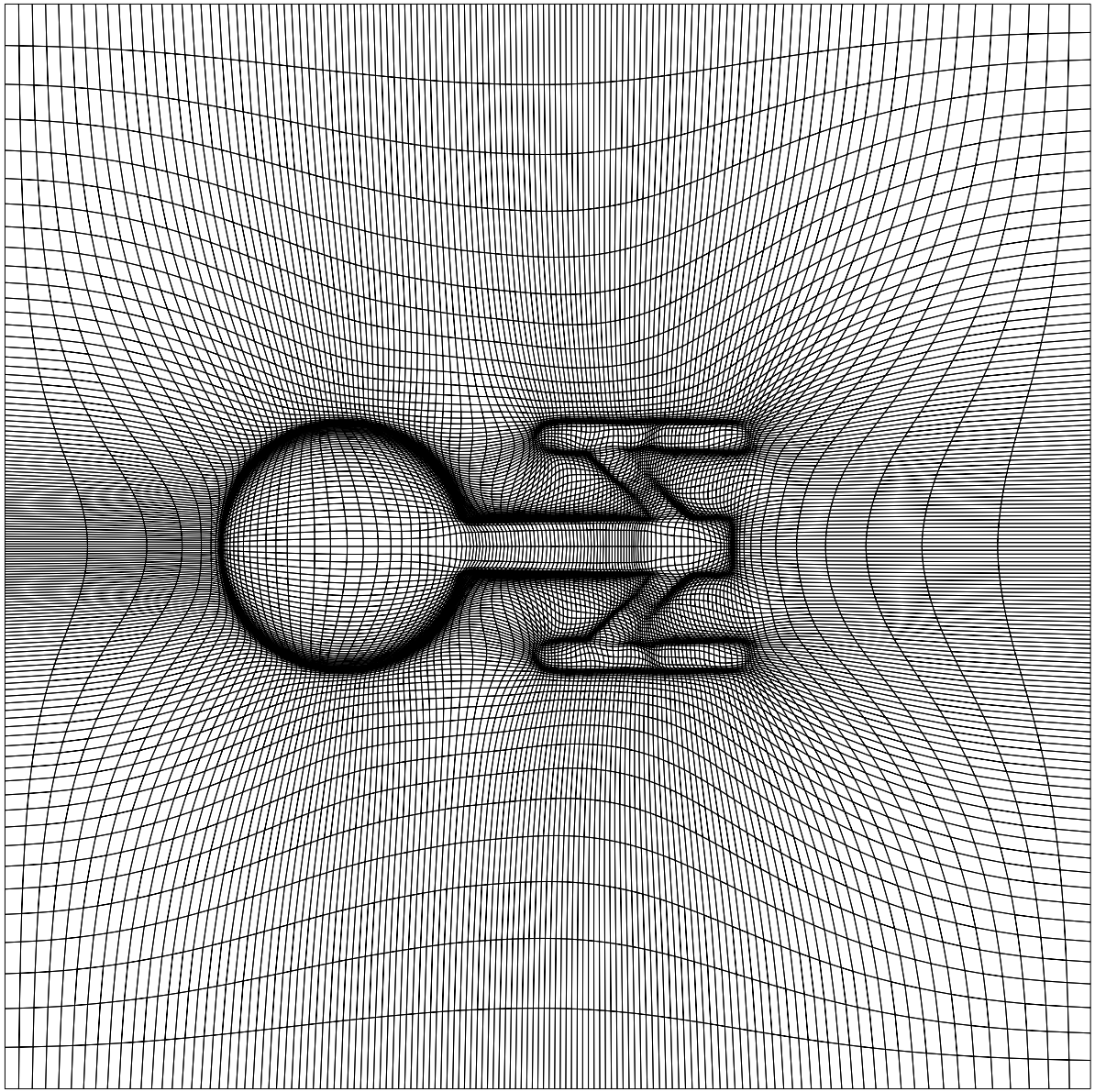
Sparse Banded **Matrix-Vector techniques**



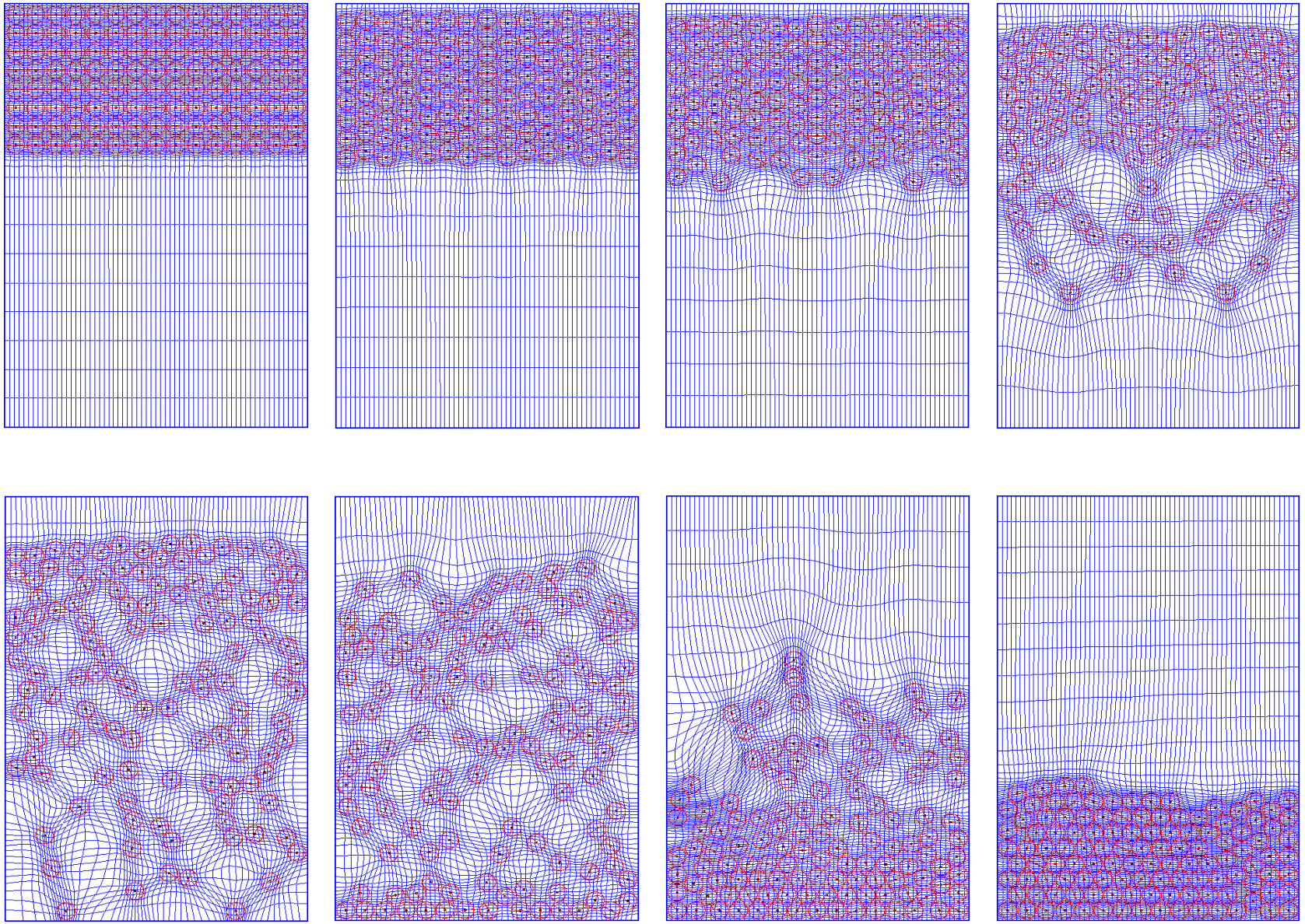
Generalized Tensorproduct Meshes



Generalized Tensorproduct Meshes



Generalized Tensorproduct Meshes



Single Processor Performance (I)

Sparse MV multiplication in (sequential) **FEATFLOW**:

Computer	#Unknowns	**CM**	**TL**	**STO**	**ILU-CM**	**ILU-TL**	**ILU-STO**
DEC 21264 (667 MHz) 'EV67'	8,320	147	136	**116**	90	76	72
	33,280	125	105	**100**	86	73	63
	133,120	81	71	**58**	81	52	55
	532,480	60	51	**21**	40	35	22
	2,129,920	58	47	**13**	38	30	14
	8,519,680	58	45	**10**	36	30	11

Single Processor Performance (II)

'Generalized Tensorproduct' meshes

2D case	NEQ	ROW (STO)	SBB-V	SBB-C	MGTRI-V	MGTRI-C
Sun V20z	65^2	2172 (633)	1806	3334	1541	2086
(2600 MHz)	257^2	574 (150)	627	2353	751	1423
'**Opteron**'	1025^2	**300 (64)**	**570**	**1774**	**538**	**943**
IBM POWER4	65^2	1521 (845)	2064	3612	906	1071
(1700 MHz)	257^2	943 (244)	896	2896	711	962
'**JUMP**'	1025^2	**343 (51)**	**456**	**1916**	**438**	**718**

SPARSE **MV techniques (**STO/ROW**)**

MFLOP/s rates vs. 'Peak Performance', problem size + numbering ???

Local Adaptivity !!!

SPARSE BANDED **MV techniques (**SBB**) + MGTRI**

'Supercomputing' (up to 4 GFLOP/s) vs. FEM for complex domains ???

Single Processor Performance (III)

Vectorization

2D case	NEQ	ROW (STO)	SBB-V	SBB-C	MGTRI-V	MGTRI-C
NEC SX-8	65^2	5070 (1521)	3611	3768	1112	1061
(2000 MHz)	257^2	5283 (1321)	6278	8363	1535	1543
'**Vector**'	1025^2	**5603 (1293)**	**7977**	**15970**	**1918**	**2053**

Necessary: Development of 'new' methods

$\Uparrow$

"Cyclic Reduction" preconditioner
"SPAI" preconditioner ($\sim$ pure MV multiplication)

Summary (I)

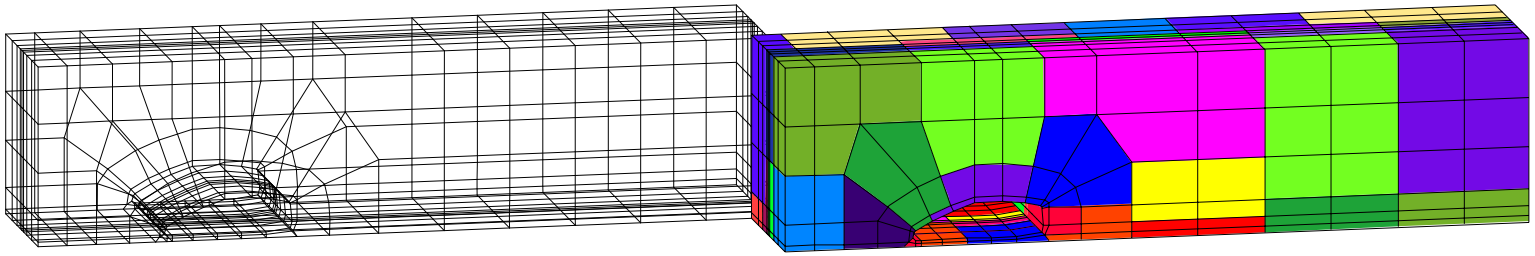
'It is non-trivial to reach Single Processor Peak Performance with modern (= high numerical efficiency) PDE tools !!!'

'Memory-intensive data/matrix/solver structures ?'

'Parallel Peak Performance with modern Numerics even harder...'

Parallel Performance (I)

'Complex (anisotropic) ASMO3D configuration'

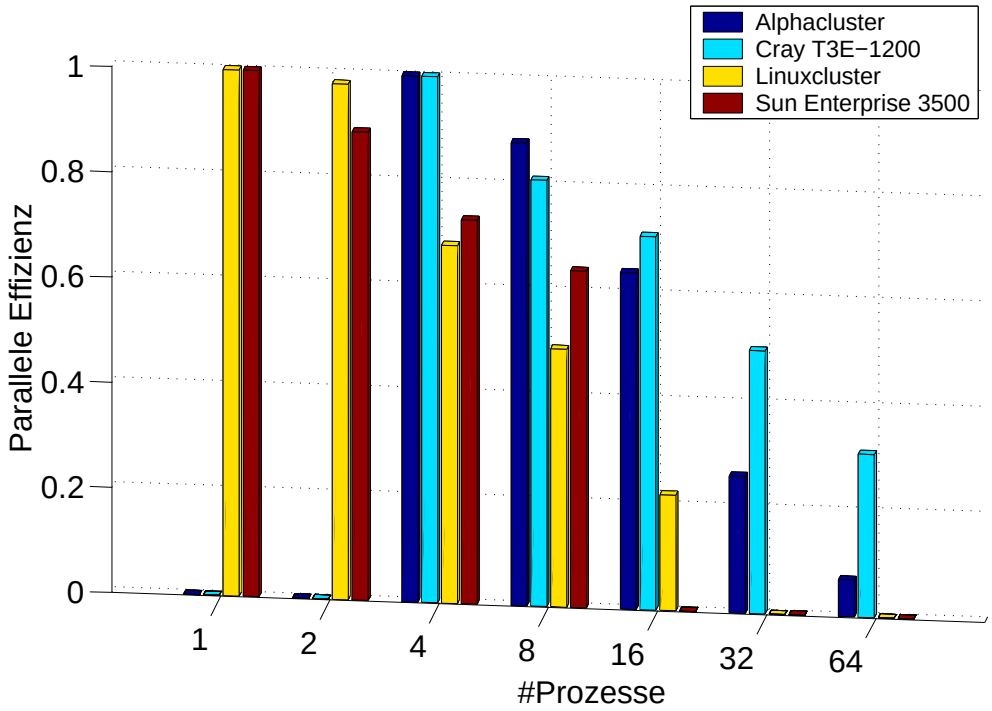


'(Moderate) mesh anisotropies (AR = 20)'

'Problems due to communication'

'Problems due to Pressure Poisson multigrid solver'

Parallel Performance (II)



	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 (II)

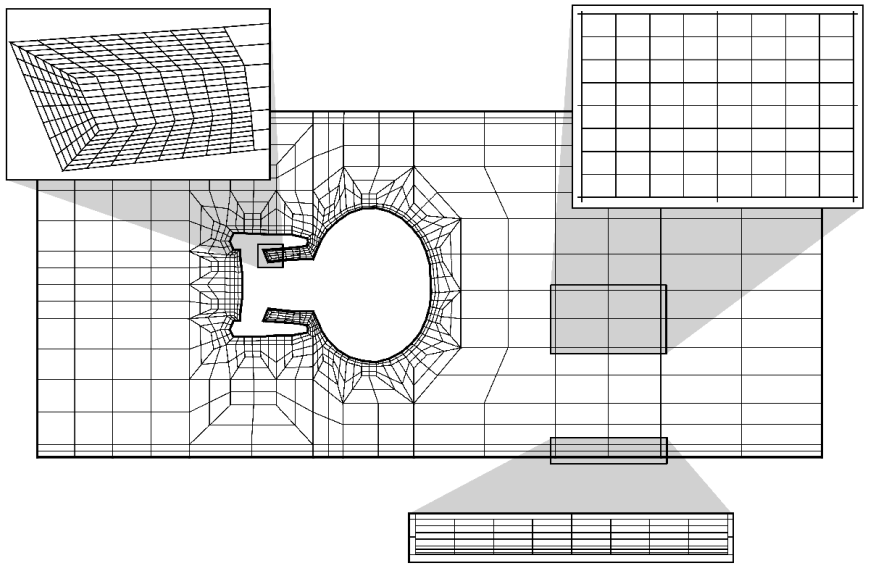
'Special requirements for numerical and algorithmic approaches in correspondance to modern hardware!'

'Hardware-Oriented Numerics for PDE'

FEAST Project

FEAST Solution Strategy

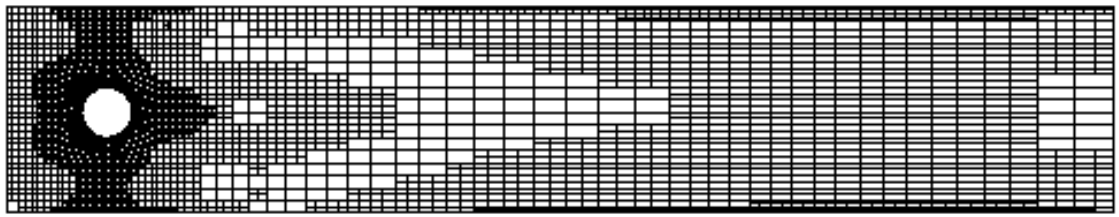
- ScaRC approach: Combine advantages of (parallel) domain decomposition and multigrid methods.
- Exploit structured subdomains for high efficiency.
- Hide anisotropies locally to increase robustness.
- Globally unstructured – locally structured.
- Recursive solution: Smooth outer global multigrid with local multigrid on the refined macros.
- Low communication overhead.



(Some) Numerical Techniques

I) *Patch-oriented h-p-r adaptivity*

'Many' local TP grids (SBB) *with arbitrary spacing*
'Few' unstructured grid parts (SPARSE)



II) *Generalized MG-DD solver:* ScaRC

Exploit locally 'regular' structures **(efficiency)**
Recursive 'clustering' of anisotropies **(robustness)**
'Strong local solvers improve global convergence !'

'Exploit locally regular structures !!!'

Open Numerical Problems

- **Adaptive remeshing ?**
  → degree of macro-refinement and/or deformation ?
  → *h*/*p*/*r*-**refinement ?** 'When to do what' decision ?

- **Load balancing ?**
  → due to 'total CPU time per accuracy per processor' ?
  → dynamical a posteriori process ?

- **(Recursive) Solver expert system ?**
  → numerical + computational a priori knowledge ?

- **'Optimality' of the mesh, resp., discretization ?**
  → *w.r.t. number of unknowns or total CPU time ?*

(Preliminary) Conclusions

- **Numerical efficiency ?**
  → *OK*
- **Parallel efficiency ?**
  → *(OK)*
- **Single processor efficiency ?**
  → *almost OK for CPU*
- **"Peak" efficiency ?**
  → *NO*
  → *Special GPU/FPGA-based FEM co-processors*

Aim: Numerics on Sony PlayStation 3

Cell multicore processor, 7 synergetic processing units @ 3.2 GHz, 218 GFLOP/s
memory clocked @ 3.2 GHz

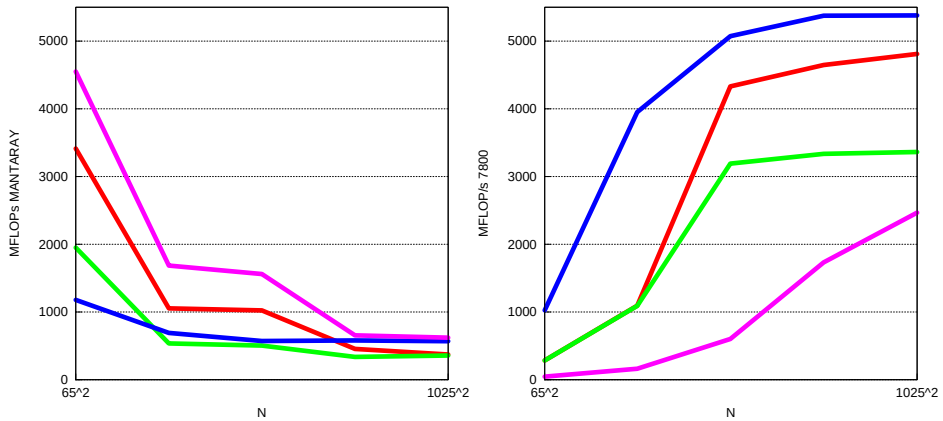
Graphics Processors: 128 parallel scalar processors @ 1.35GHz, 900 MHz GDDR3 memory (86.4 GB/s), $\approx$ 500 GFLOP/s

Motivation

- We want to solve large systems that arise from FEM discretisations fast on commodity clusters.
- CPUs are general-purpose and only achieve close-to-peak performance in-cache. CPUs devote most of the area to memory (hierachies) and not to processing elements (PEs).
- Emerging parallel specialised chips are PE-dominated and provide potentially lots of FLOPS and huge memory bandwidth.
- Goal: Investigate how such designs can be used as numerical co-processors in scientific computing.
- **Focus exemplary on Graphics Processors (soon: CELL Processor)**

Benchmarks: FEM building blocks

Typical performance of FEM building blocks SAXPY_C, SAXPY_V (variable coefficients), MV_V (9-point-stencil, Q_1 elements), DOT on Opteron 244 (SBBLAS) and GeForce 7800 GTX, $N = 65^2 \ldots 1025^2$:



- Basic linear algebra operations on banded matrices are typically memory-bound, we see $\approx 95\%$ peak memory bandwidth
- Comparable to in-cache performance on CPU for large vectors and matrices, but for large problem sizes!
- Open question: How to make them compute-bound?

GPU Limitations

Challenge: Reformulate algorithms to the data-stream based programming paradigm!

- PCIe bus between host system and GPU delivers up to 2 GB/s only.
- GPUs only provide quasi-IEEE 32-bit floating point storage and arithmetics. No double precision! Tests for Poisson equation in 2D

$$-\Delta \mathbf{u} = \mathbf{f} \text{ in some domain } \Omega \subset \mathbb{R}^2 \text{ with Dirichlet BCs}$$

Discretised with bilinear conforming Finite Elements.

		single precision			double precision	
Level	Cycles	Error	Reduction	Cycles	Error	Reduction
2	1	2.391E-3		1	2.391E-3	
3	2	5.950E-4	4.02	2	5.950E-4	4.02
4	2	1.493E-4	3.98	2	1.493E-4	3.99
5	2	3.750E-5	3.98	2	3.728E-5	4.00
6	2	1.021E-5	3.67	2	9.304E-6	4.01
7	2	6.691E-6	1.53	2	2.323E-6	4.01
8	2	2.012E-5	0.33	2	5.801E-7	4.00
9	2	7.904E-5	0.25	2	1.449E-7	4.00
10	2	3.593E-4	0.22	2	3.626E-8	4.00

Mixed Precision Iterative Refinement

- Single precision computation insufficient for required result accuracy, but: High precision only necessary at few, crucial stages!
- **Mixed precision iterative refinement** approach to solve $A\mathbf{x} = \mathbf{b}$:

Compute	$\mathbf{d}$	$=$	$\mathbf{b} - A\mathbf{x}$	in high precision.
Solve	$A\mathbf{c}$	$=$	$\mathbf{d}$	approximately in low precision.
Update	$\mathbf{x}$	$=$	$\mathbf{x} + \mathbf{c}$	in high precision and iterate.

- Use arbitrary iterative inner solvers until *"few"* digits are gained locally.
- Fits naturally on target hardware: Few, high precision updates on the CPU and expensive low precision iterative solution on the GPU.
- Exhaustive experimental and theoretical foundation: very robust wrt. solvers, degrees of anisotropy in the discretisation and matrix condition.

Results

- Poisson on unit square, regular refinement, conforming bilinear Q_1 elements. Multigrid solver with Jacobi smoother
- CPU: Athlon X2 4400+
- GPU: GeForce 7800 GTX, mixed precision iterative refinement

N	CPU time	CPU error	GPU time	GPU error	speedup
127^2	0.28	1.666003670E-6	0.26	1.666003655E-6	1.08
257^2	0.69	4.181054493E-7	0.33	4.181054014E-7	2.09
513^2	1.95	1.047283071E-7	0.56	1.047281043E-7	3.48
1025^2	7.07	2.620418265E-8	1.69	2.620376988E-8	4.18

Accuracy: Same error as double precision FEAST solver compared to analytically known reference solution.

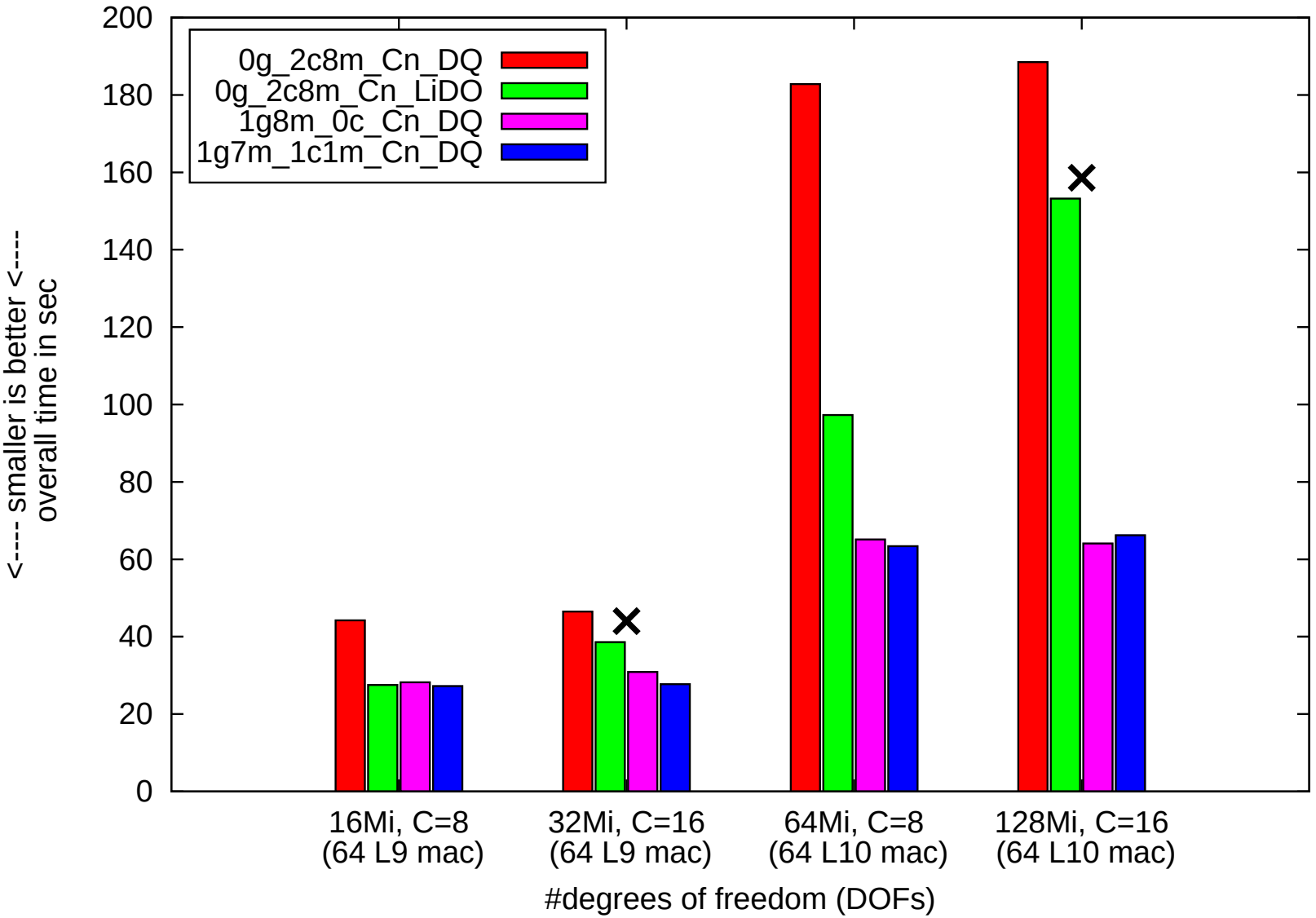
Integration into FEAST

- FEAST: Under development since 1999, 100K+ lines of code, tuned data structures, adaptions for clusters (MPI) and NEC vector machines.
- Consequence: Full rewrite to incorporate GPUs is out of question!
- Goal: Minimally invasive integration.
- Large-scale solution scheme: global MG smoothed by many local MGs
- GPU backend adds new smoother, while FEAST maintains all global data structures.
- Data flow example: Outer MG calls smoother, matrix and current defect are duplicated into GPU memory, smoothing is performed independently, correction term is read back to the CPU.

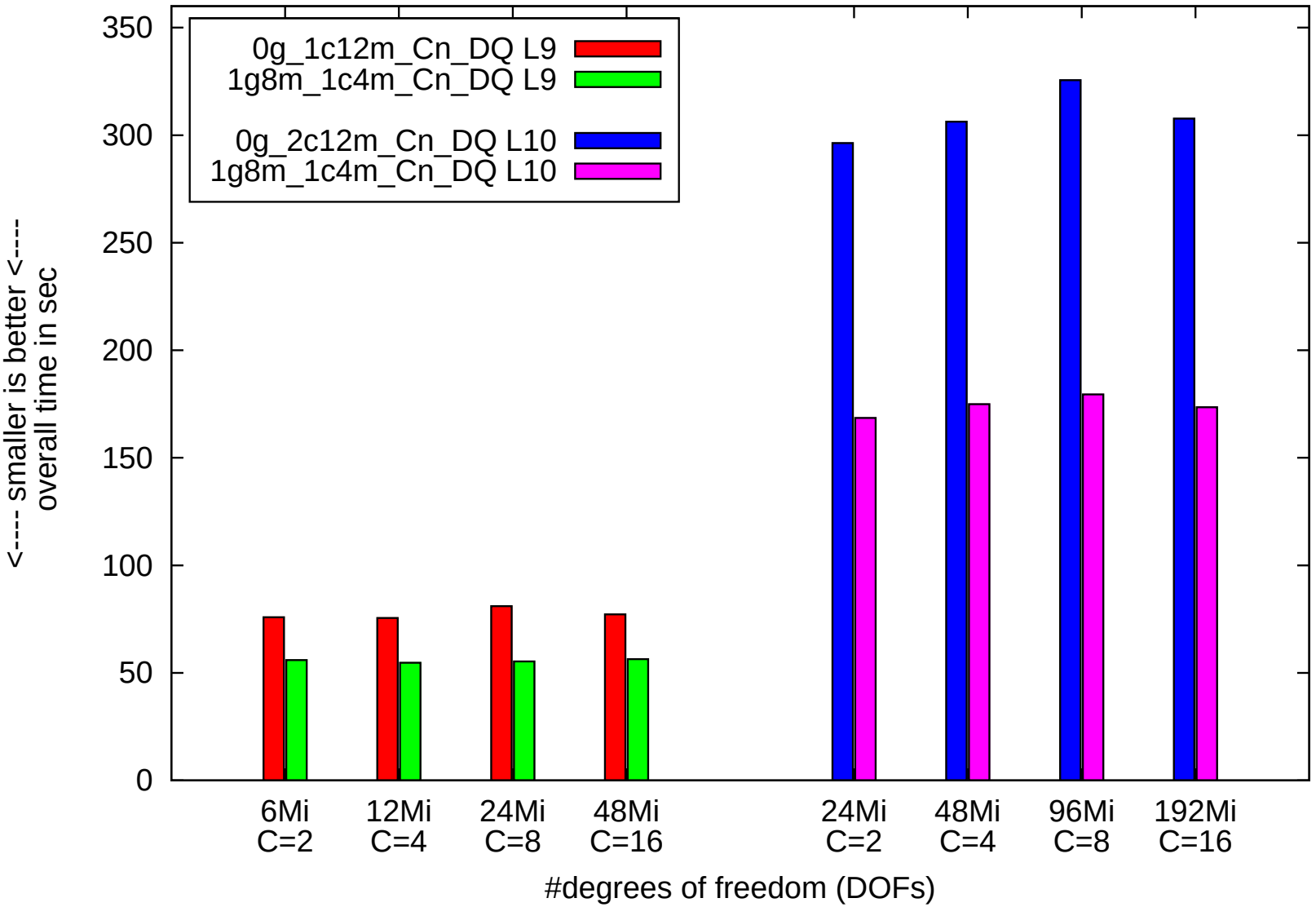
Preliminary GPU cluster results

- Joint work with colleagues from Stanford and Los Alamos.
- Cluster with 16 compute nodes and 1 master node.
- Dual Intel EM64T 3.4 GHz, NVIDIA Quadro FX4500 PCIe graphics card.
- Fully connected via Infiniband.
- Two test cases:
  - I Homogeneous domains: pure CPU vs. pure GPU vs. one CPU and one GPU per node
  - II Heterogeneous domains: CPU treats few unstructured subdomains and GPU treats many structured subdomains (each does what it's best at)

Test Case I


0g_2c8m_Cn_DQ
0g_2c8m_Cn_LiDO
1g8m_0c_Cn_DQ
1g7m_1c1m_Cn_DQ
<--- smaller is better <---
overall time in sec
0
20
40
60
80
100
120
140
160
180
200
16Mi, C=8
(64 L9 mac)
32Mi, C=16
(64 L9 mac)
64Mi, C=8
(64 L10 mac)
128Mi, C=16
(64 L10 mac)
#degrees of freedom (DOFs)

Test Case II


0g_1c12m_Cn_DQ L9
1g8m_1c4m_Cn_DQ L9
0g_2c12m_Cn_DQ L10
1g8m_1c4m_Cn_DQ L10
<---- smaller is better <----
overall time in sec
0
50
100
150
200
250
300
350
6Mi
C=2
12Mi
C=4
24Mi
C=8
48Mi
C=16
24Mi
C=2
48Mi
C=4
96Mi
C=8
192Mi
C=16
#degrees of freedom (DOFs)

Summary

- Interesting perspectives:
  - Inexpensive upgrade of commodity clusters.
  - Potential to accelerate production codes.
  - But: Maintaining two code lines on the solver and data structure level, not on the application level.
- Paradigm shift to data parallelism: Multicores, Cell BE etc., so start learning now: The first honest attempt at petascale computing, the IBM Roadrunner at LANL, will contain multi-GPUs, Cells, Opterons and will in general be a massively parallel hybrid machine.

Conclusions

There is a huge potential for the future...

But: Numerics has to consider recent and future hardware trends!

But: Developing 'HPC-PDE software' is more than the implementation of existing Numerics for PDE!

→ Understanding and definition of 'Total Numerical Efficiency'
→ Design, analysis and realization of hardware-oriented Numerics
→ Identification and realization of hardware-optimized basic components
→ CPU-GPU clusters as 'building blocks'

Do not forget Terascale tools for "daily life" !