.

Trends in processor technology

and their impact on

Numerics for PDE's

S. Turek

Institut für Angewandte Mathematik, Universität Heidelberg
Im Neuenheimer Feld 294, 69120 Heidelberg, Germany

`http://gaia.iwr.uni-heidelberg.de/~ture`
`http://www.iwr.uni-heidelberg.de/~featflow`

October 1999: **Universität Dortmund**

Two main topics:

'A posteriori (error) control of FEM/FV discretizations with adaptive meshing strategies'

'(Iterative) Solution strategies for huge systems of equations'

- Is **error control** with **adaptive meshing** necessary?
  $\longrightarrow$ **YES !!!**
- Does **error control** lead to more efficient **simulation** tools?
  $\longrightarrow$ **??** $\Rightarrow$ 'real life' applications (**CFD** ???)
- Do **adaptive meshes** lead to more efficient **simulations**?
  $\longrightarrow$ **?** $\Rightarrow$ RAM ???
  $\longrightarrow$ **???** $\Rightarrow$ **CPU ???**

$\Uparrow$

- Robust and efficient (complete) solvers !
- **Modern processor technology !!!**

Main components in iterative schemes:

Matrix-Vector applications (MV)

- Krylov-space methods, Multigrid, etc.
- **Defect calculations**, smoothing, step-length control, etc.
- Sometimes consuming (at least) **60 - 90%** of CPU time

$\Downarrow$

MV techniques (Storage/Application):

1) *Sparse MV*

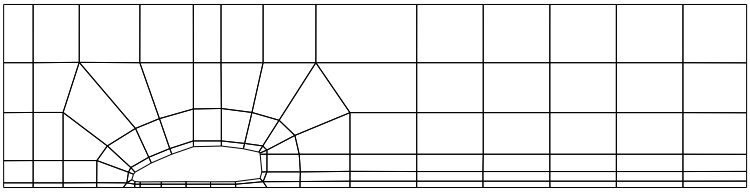
$\rightarrow$ Standard technique in most **FEM** or **FV** codes
$\rightarrow$ Storage of '**non-zero**' matrix elements only (*CSR, CSC, ...*)
$\rightarrow$ Access via **index vectors, linked lists, pointers**, etc

2) *(Sparse) Banded MV*

$\rightarrow$ Typical for **FD** codes on '**tensorproduct meshes**'
$\rightarrow$ Storage of '**non-zero**' elements in **bands/diagonals**
$\rightarrow$ MV multiplication '**bandwise**' (and '**window-oriented**')
$\rightarrow$ **FEAST !!!**

Results on general unstructured meshes

- SPARSE **MV** multiplication in **FEATFLOW** (F77 !!!)



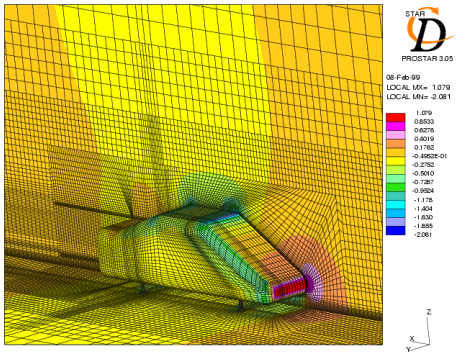
Computer	#Unknowns	**CM**	**TL**	**STO**
	13,688	22	20	**19**
SUN E450	54,256	17	15	**13**
($\sim$ 250 MFLOP/s)	216,032	16	14	**6**
	862,144	16	15	**4**

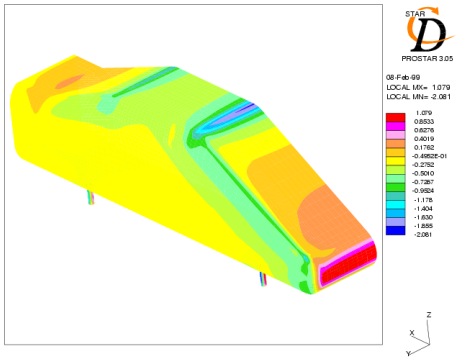
- **STAR-CD**, **500,000 hexaeders** (by DaimlerChrysler)
- SGI Origin2000 (**6 processors**), **6.5 h** (CPU)

$\Uparrow$

'Optimal' multigrid $\sim$ **0.1 sec/subproblem ???**

Quantity	**Experiment**	**Simulation**	**Difference**
Drag ('c_w')	0.165	0.317	**92 %**
Lift ('c_a')	-0.083	-0.127	**53 %**





First conclusions for *sparse MV*:

Different numbering strategies can lead to:

- **identical** numerical results and work (arithm. operations, memory accesses)!
- **huge differences** in elapsed CPU time!

Sparse MV techniques are '**slow**' and depend on:

- **problem size**!
- 'amount' and 'kind' (?) of **data access**!

Sparse MV techniques are **basis** for:

- Most available **commercial** codes!
- Most recent **research** software projects!

Results on locally structured meshes

3D case	N	STO	LINE	SBB-V	SBB-C	**MG-V**	**MG-C**
DEC 21264	17^3	150	164	446	765	342	500
(500 MHz)	33^3	54	64	240	768	233	474
'**DS20**'	65^3	**24**	**72**	**249**	**713**	**196**	**447**
IBM RS6000/597	17^3	81	86	179	480	171	368
(160 MHz)	33^3	16	81	170	393	152	300
'**SP2**'	65^3	**8**	**81**	**178**	**393**	**150**	**276**
INTEL PII	17^3	28	29	56	183	48	136
(400 MHz)	33^3	24	29	53	139	47	116
'**ALDI**'	65^3	**19**	**29**	**54**	**125**	**45**	**101**

Sparse **MV techniques (STO,LINE)**

- MFLOP/s rates far away from '**Peak Performance**'
- Depending on **problem size + numbering**
- '**Old**' (IBM PWR2) partially **faster** then '**new**' (IBM P2SC)
- PC partially **faster** than processors in 'supercomputers' !!!

Feast **MV techniques (SBB)**

- 'Supercomputing' power gets visible (up to **800 MFLOP/s**)
- **Warning:** Hard work !!! ($\rightarrow$ Sparse Banded Blas)

Further conclusions:

'Most adaptive codes should run on (PENTIUM) PC's'

'Processors are 'sensible' Parallel-Vector supercomputers w.r.t. **caching-in + pipelining***'*

Evaluation of Soft/Hardware components ???

Adaptivity concepts ???

Realization of complete FEM/FV approaches ???

⇓

FEAST project

Expected hardware development:

1997 National Technology Roadmap for Semiconductors						
Year	1997	1999	2003	2006	2009	2012
Transistors/chip	**11M**	21M	76M	200M	520M	**1.4B**

1. 1 Billion Transistors ($\times$ **100 !**)
2. 10 Ghz clock rate ($\times$ **20 !**)

1 PC 'faster' than **complete** *CRAY T3E today !*

Memory access ???
Data locality and internal parallelism ???
Vectorization ???

FEAST project

⇓

Numerics and **implementation**
techniques adapted to **hardware !!!**

⇓

Precise knowledge of **processor** characteristics

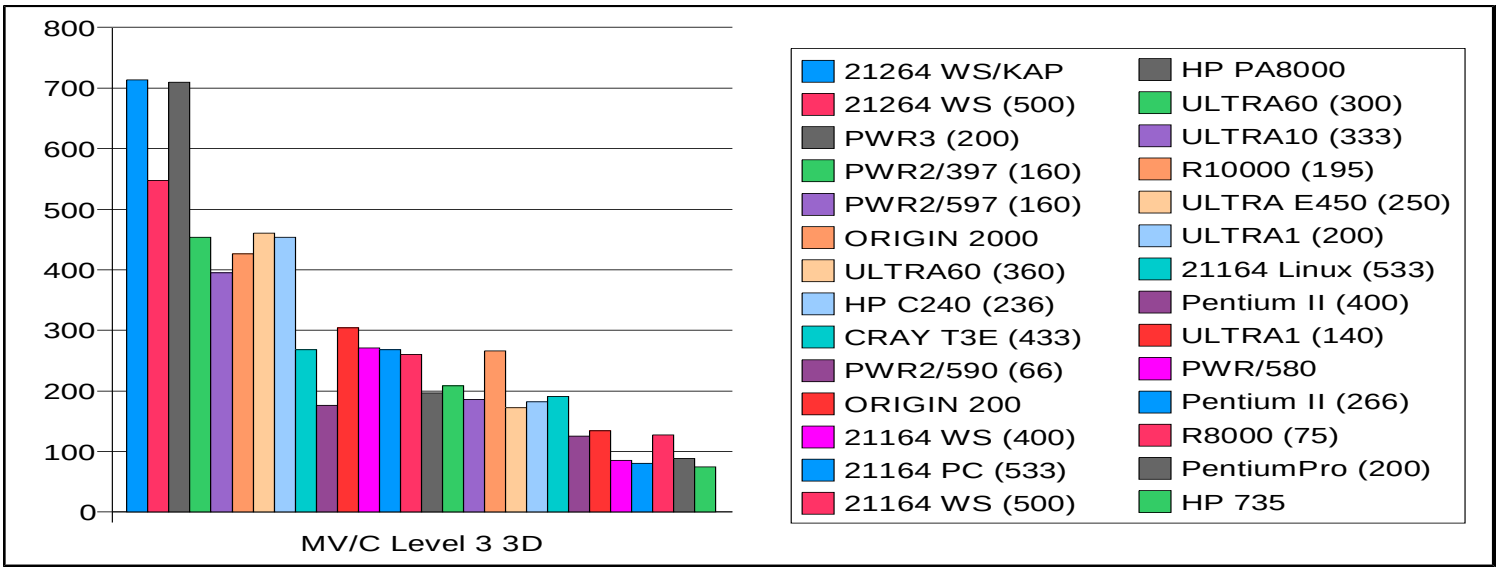
⇓

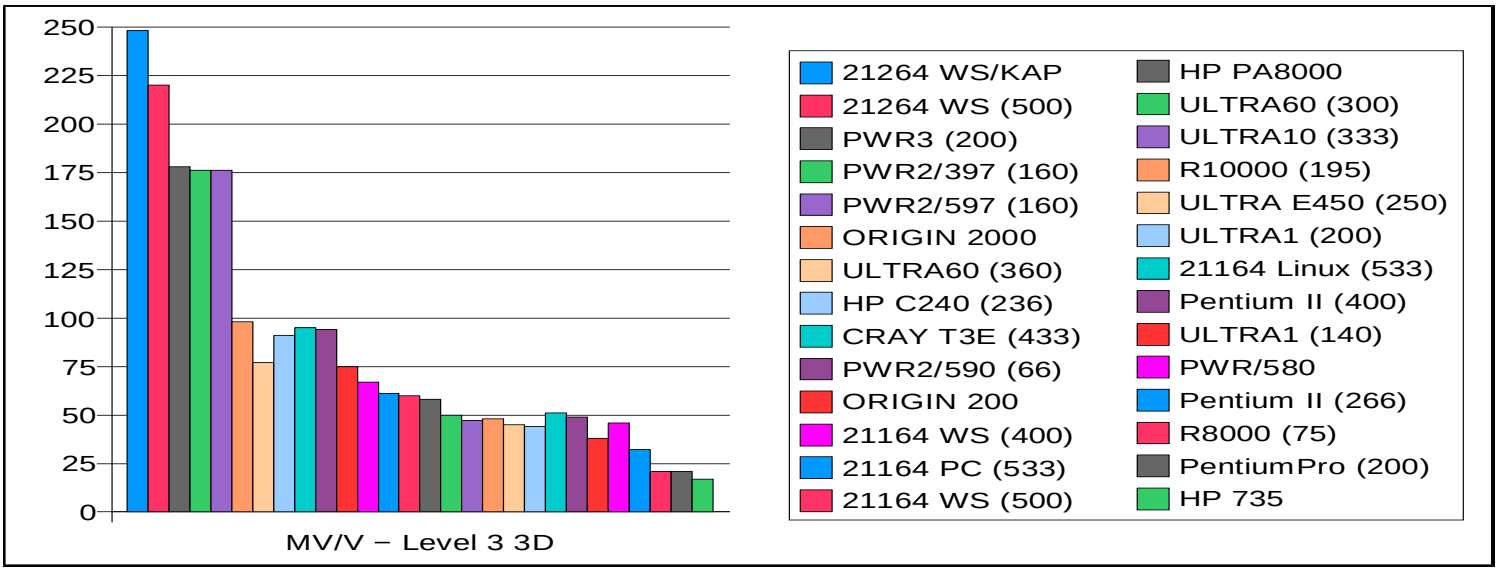
(Collection of) **FEAST INDICES**

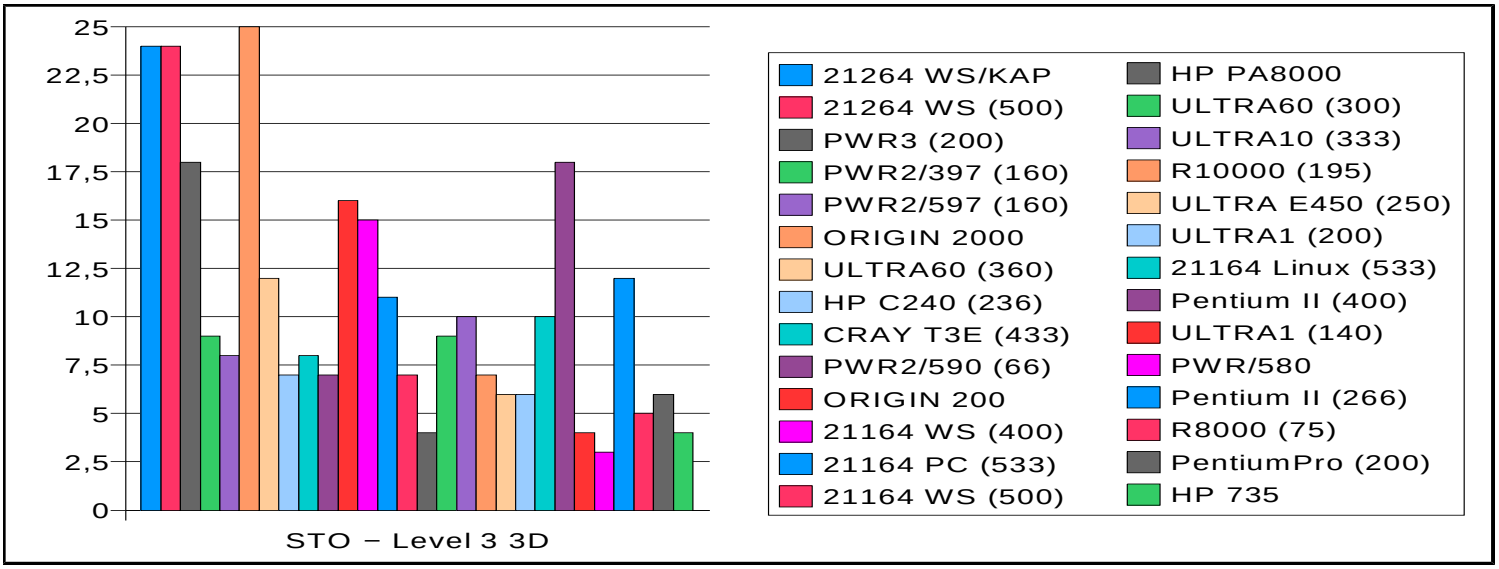
⇕

Results for the different **tasks** *in*
MG/Krylov-space *solvers for various*
FEM/FV *discretizations and* **mesh sizes**

MV-MULT with identical matrix !!!





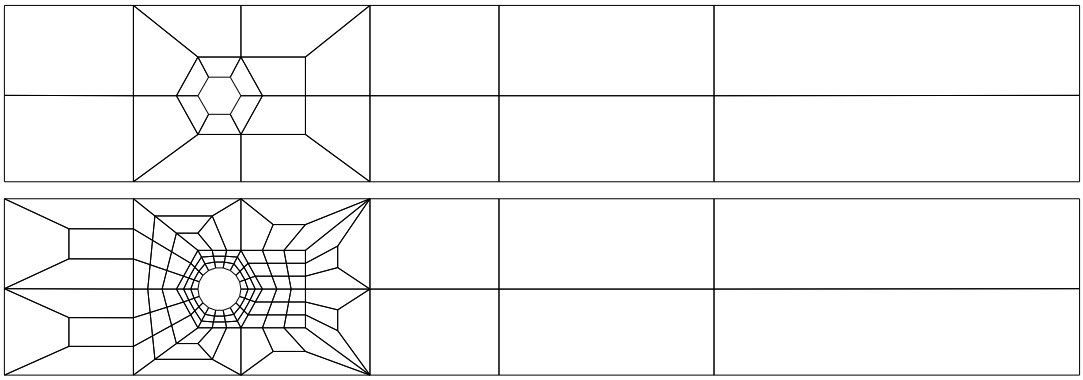


Example: Concepts for adaptive meshing

1) macro-oriented adaptivity

→ '**blind**' (1–irregular) macro-nodes

→ **conforming** completion



⇓

2) patchwise adaptivity

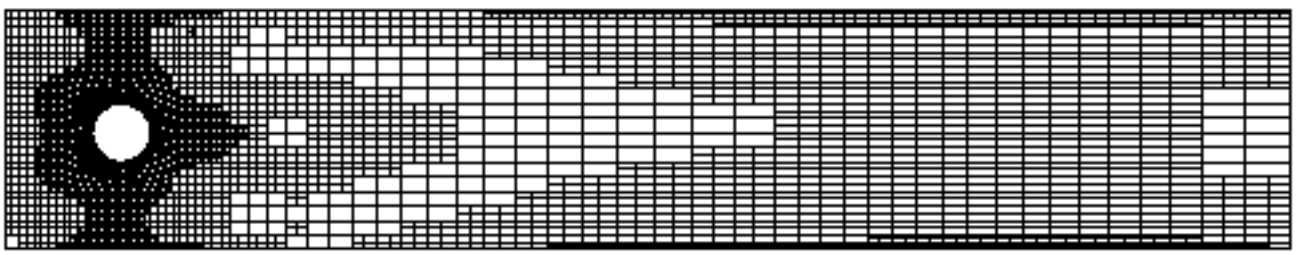
→ many (local) logically equivalent **tensorproduct** meshes

→ **moving** mesh points

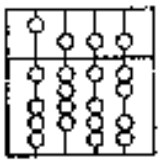
⇓

3) fully local adaptivity

→ (local) **unstructured** meshes



'Exploit the locally nice structures !'



FAKULTÄT FÜR INFORMATIK

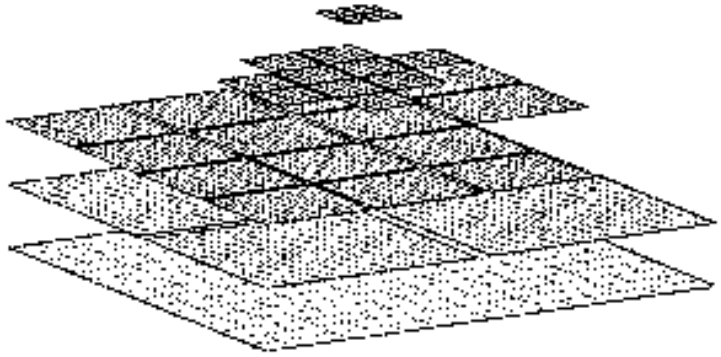
DER TECHNISCHEN
UNIVERSITÄT MÜNCHEN

Parallele adaptive Mehrgitterverfahren

Ein objektorientierter Ansatz auf semistrukturierten Gittern

Diplomarbeit

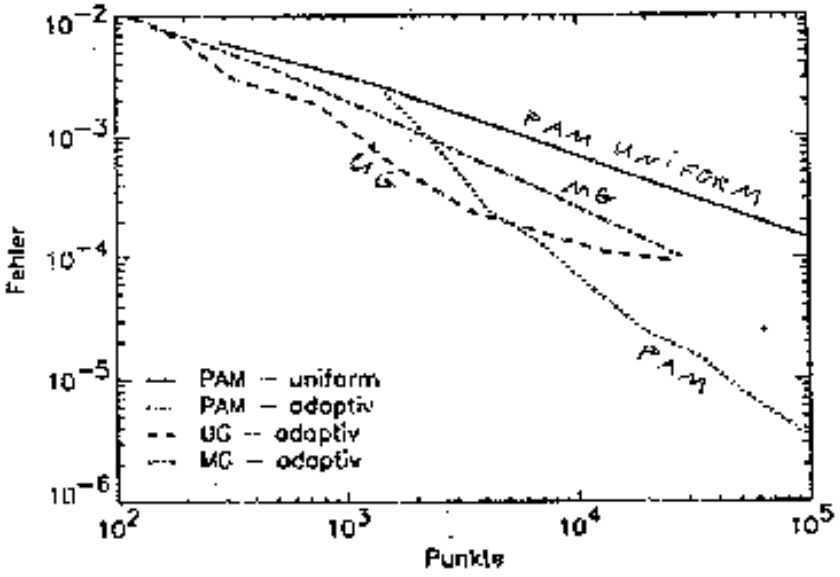
Heiko Lötzbeyer

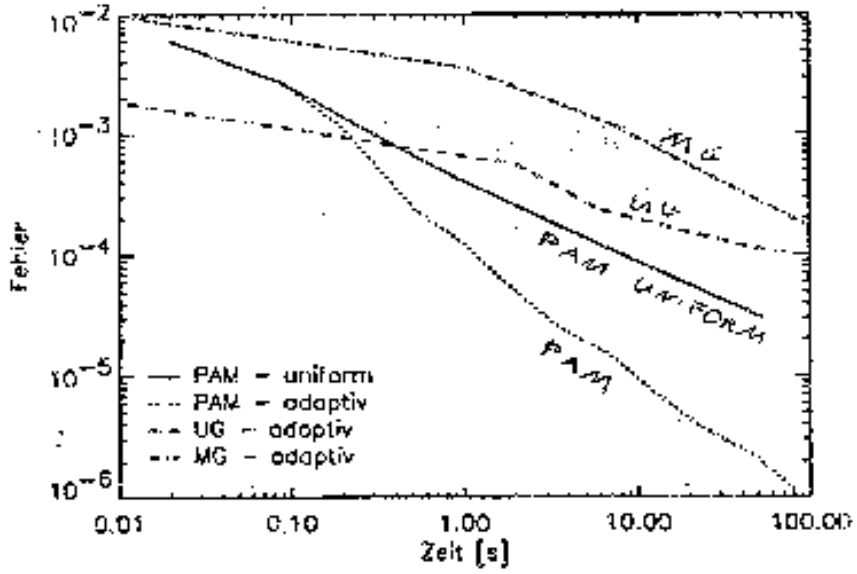


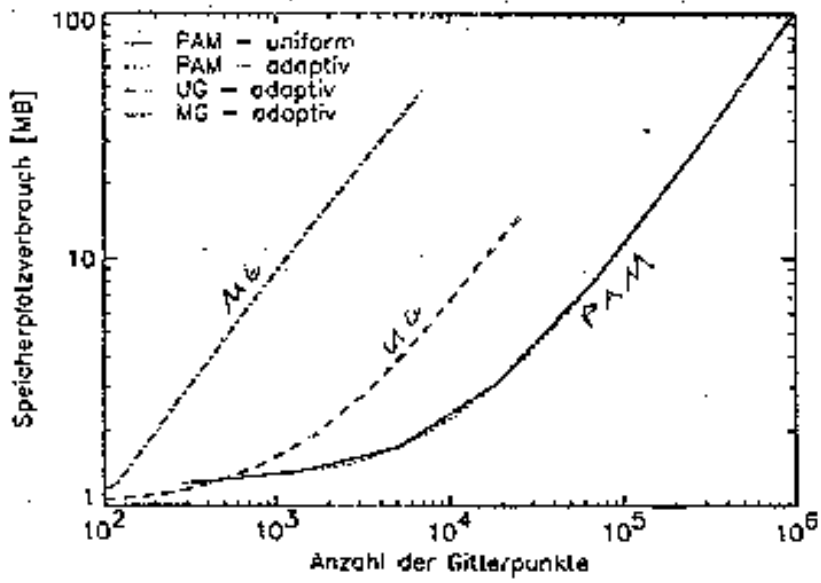
Aufgabensteller: Prof. Dr. Chr. Zenger
Betreuer: Prof. Dr. U. Rüde[1]

Abgabedatum: 15.12.1996

[1] Institut für Mathematik der Universität Augsburg


Fehler
Punkte
PAM – uniform
PAM – adaptiv
UG – adaptiv
MG – adaptiv
PAM UNIFORM
MG
UG
PAM


Fehler
Zeit [s]
PAM – uniform
PAM – adaptiv
UG – adaptiv
MG – adaptiv
MG
UG
PAM UNIFORM
PAM


Speicherplatzverbrauch [MB]
Anzahl der Gitterpunkte
PAM – uniform
PAM – adaptiv
UG – adaptiv
MG – adaptiv
MG
UG
PAM

Conclusions:

Everybody can/must (?) test the computational performance!

→ FORTRAN 90, C++, JAVA ???

Most adaptive FEM codes are NOT slower on PC's than on 'High Performance' workstations or even 'supercomputers'!

→ **However:** there is 'supercomputing power' available!!!

'Amount' and 'kind' of memory accesses determines significantly the computational efficiency!

→ modern processors are '**sensible**' supercomputers!!!

User-defined memory managment and optimized implementations 'w.r.t. hardware' are absolutely necessary!

→ massive performance losses due to Cache/Heap organization!!!

Modern Numerics has to consider recent and future hardware trends!

→ math. theory for rigorous **macro-oriented adaptivity** ???

→ improved **MG/DD** solvers (= ScaRC !!!)

→ **algorithmic design:** Balance of **FLOPs** and **data access**