Some Concepts of the software package

FEAST

(Finite Element Analysis & Solution Tools)

Ch. Becker, S. Kilian, S. Turek

Institut für Angewandte Mathematik,
Universität Heidelberg

email: {cbecker,susanne,ture}@gaia.iwr.uni-heidelberg.de

WWW: http://www.iwr.uni-heidelberg.de/featflow

Motivation

General observations:

- peak performance increases with every processor generation (in 10 years single PC faster than a Cray today)

1997 National Technology Roadmap for Semiconductors							
Year of 1st shipment	1997	1999	2001	2003	2006	2009	2012
Local clock (Mhz)	750	1250	1500	2100	3500	6000	**10K**
Chip size (mm^2)	300	340	385	430	520	620	750
Feature size (nm)	250	180	150	130	100	70	50
Transistors/chip	11M	21M	40M	76M	200M	520M	**1.4B**

- memory size also improves, but not the access time
- huge resources are needed for real world calculations

$\Longrightarrow$ gap between processing time and memory access increases!

Questions:

- which degree of performance can be achieved by recent software packages?
- are special strategies necessary to exploit higher performance?

Performance Tests I

Performance rates ($\frac{20N}{T}$) of a real code (FEATFLOW) with different numbering schemes for matrix/vector multiplication:

Computer	N	**TL**	**SW**	**ZW**	**CM**	**ST**
	3484	25	22	22	22	**21**
SUN U450	13688	20	23	22	22	**19**
(250 Mhz)	54256	15	17	16	17	**13**
	216032	14	16	16	16	**6**
	862144	15	15	15	16	**4**

Results:

- identical computational cost
- but very different runtimes
- far away from peak performance
- problem size dependent
- complete multigrid even worse

Performance Tests II

Multigrid/Krylov schemes, especially matrix/vector operations are CPU expensive

Test of various matrix/vector multiplication techniques:

- **sparse,SMV**: storing the nonzero entries with position
- **band,BMV**: storing diagonal by diagonal
- **blocked band,BB.**: blockwise diagonal application, BLAS 2+

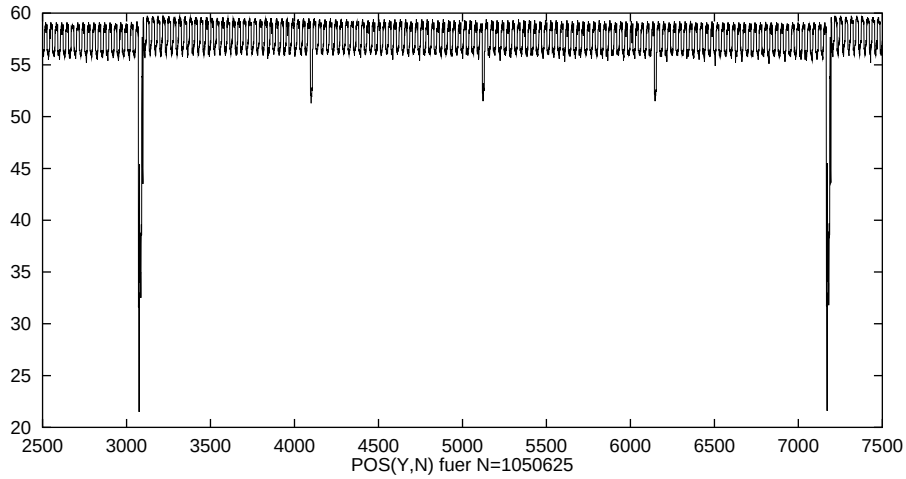
Test of Gaussian Elimination (Linpack): "Peak Test" (**GE**)

		$\frac{20N}{T}$				
Computer	N	SMV	BMV	BBMVA	BBMVC	GE
IBM RS6000	4K	54 (**114**)	154	154	240	365
(166 Mhz)	64K	49 (**24**)	71	140	234	
'SP2'	256K	50 (**7**)	71	143	240	
DEC Alpha	4K	34 (**17**)	46	81	223	309
(433 Mhz)	64K	22 (**14**)	42	63	189	
'CRAY T3E'	256K	22 (**7**)	42	67	196	
SUN U450	4K	35 (**83**)	102	100	127	246
(250 Mhz)	64K	15 (**16**)	21	35	119	
	256K	14 (**4**)	17	36	99	
INTEL PII	4K	20 (**32**)	28	24	60	34
(233 Mhz)	64K	11 (**12**)	14	20	47	
'Home PC'	256K	– (**8**)	–	–	–	

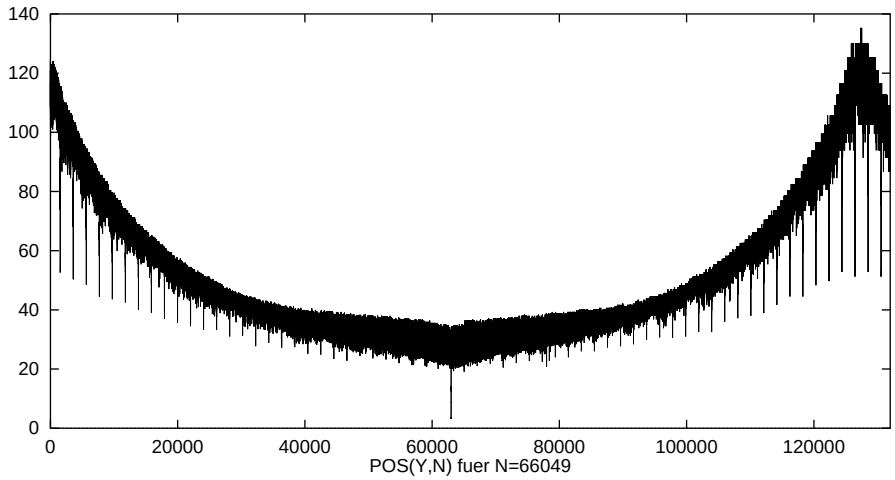
Performance Tests III

Cache test for vector/vector addition (variation of the distance of the two vectors)

Cray T3E:



Sun U450:



- performance highly dependent of memory position
- small “windows“ of higher performance
- how to control the memory position ???

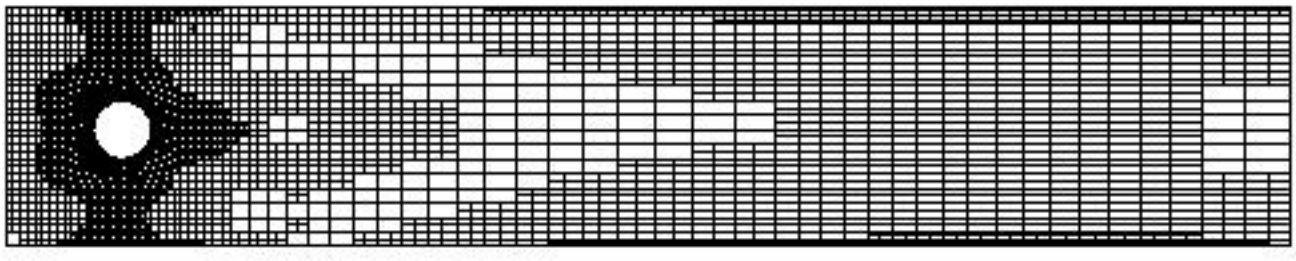
Conclusions

- iterative schemes with classical implementation often far beyond possible peak performance
- Gaussian Elimination highly efficient, but useable?
- standard sparse MV operations cannot be handled with high performance
- gap between PC and "number cruncher" not so great for recently used software
- processor advance not so easy useable
- banded block FD techniques clearly superior (w.r.t. efficiency and problem size), how realizable in FEM codes?

$\Longrightarrow$ **FEAST**

Hierarchical Divide and Conquer

Optimal grid for drag–coefficent via a posteriori error estimation:

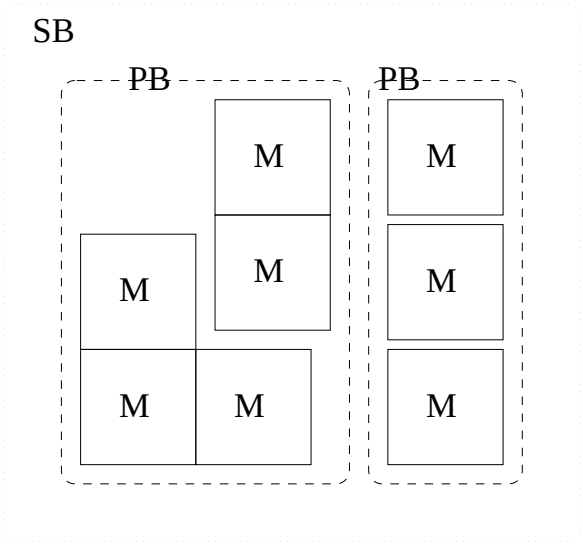


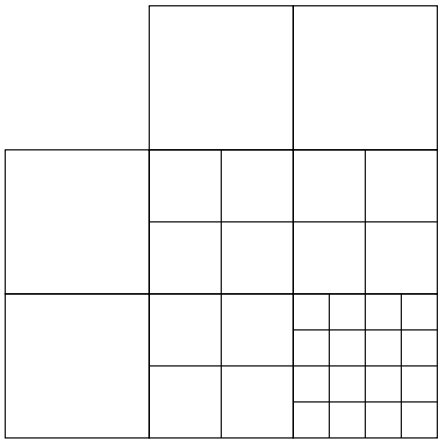
(Roland Becker, Inst.f.Ang.Math., Universität Heidelberg)

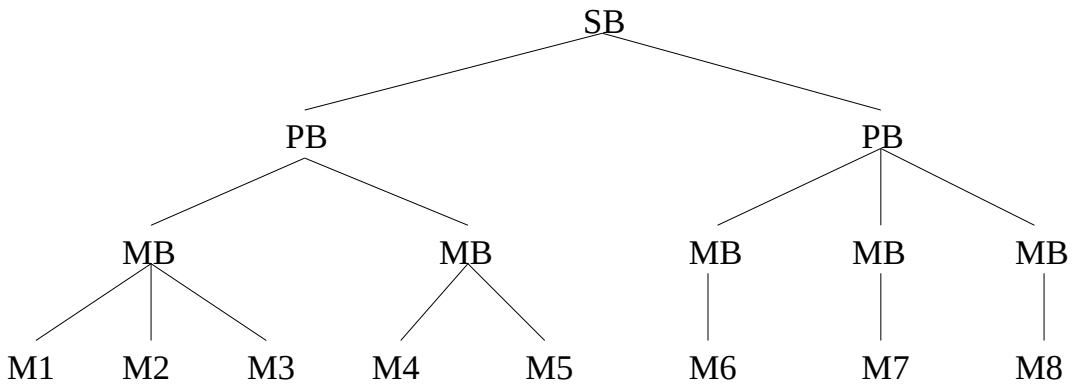
- adaptive techniques only locally near boundaries
- regular substructures (90 %) in interior
- typical example for CFD

- recursive **Divide and Conquer** strategy
- find locally structured parts (high performance)
- find locally anisotropic parts (hide locally, robustness)

Hierachical Structures







Structure parts: atomic units, matrix units, parallel units, subdomains, domain

- regular structures
- high performance linear algebra possible
- parallel calculation easily possible

Question: Corresponding solver?

Solver Structure

Method: **ScaRC** (**Sca**lable **R**ecursive **C**lustering)

- generalized multigrid/ domain decomposition scheme
- **recursive** combination of **global** defect correction with **local** iterative schemes/direct solvers
- information exchange per "global" multigrid
- matrix–vector application only **locally** on "matrix blocks"
- adaptive use of variant smoothers according to "hardness" of the domain (degree of anisotropy)

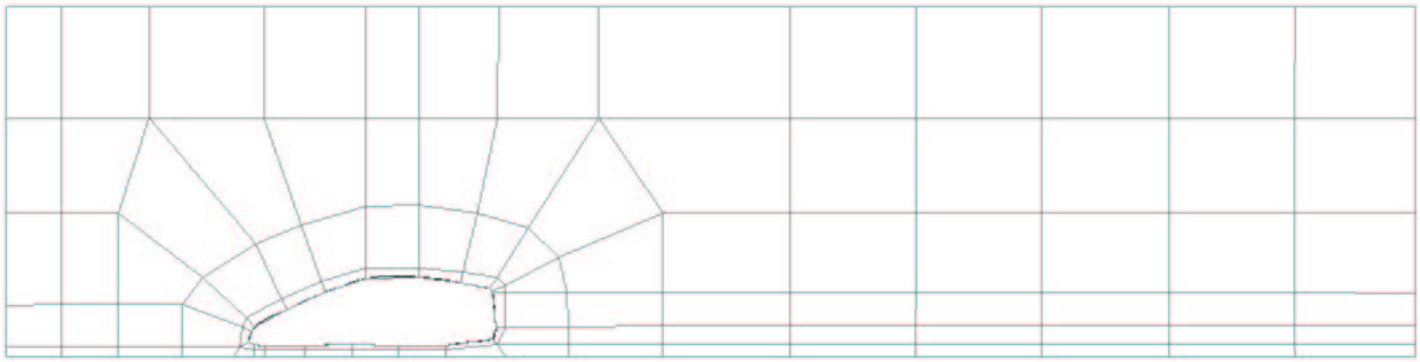
Recent results:

- excellent convergence rates of the solver
  - "independent" of grid size and number of subdomains
  - no overlap (implementation !!!)
- application of our new block banded techniques

  $\Rightarrow$ high performance possible
- robust handling of anisotropic regions and complex details
- automatic parallelization included !!!

ScaRC example

Example topology:



(r_1, r_2, r_3)	N_l	l	*Jacobi*			*SOR*			*ScaRC*		
			ite	ρ	t_{total}	*ite*	ρ	t_{total}	*ite*	ρ	t_{total}
(0.50,1.00,1.00)	8	1	**100**	0.901	7.85	**62**	0.799	4.66	**12**	0.297	1.14
		2	**90**	0.858	7.24	**33**	0.653	3.23	**6**	0.093	0.71
		4	**47**	0.745	5.94	**17**	0.439	1.92	**3**	0.010	0.46
	16	1	**100**	0.911	12.25	**84**	0.848	8.69	**12**	0.297	1.95
		2	**100**	0.886	12.75	**45**	0.734	6.48	**6**	0.096	1.39
		4	**67**	0.813	13.71	**24**	0.556	5.68	**4**	0.017	1.75
	32	1	**100**	0.914	18.46	**100**	0.872	18.13	**12**	0.296	5.83
		2	**100**	0.894	26.36	**55**	0.776	13.60	**7**	0.116	5.71
		4	**83**	0.846	32.82	**29**	0.619	12.54	**4**	0.021	5.88
(0.50,0.50,0.50)	8	1	**100**	0.894	9.39	**60**	0.794	5.24	**11**	0.271	1.15
		2	**82**	0.844	8.62	**32**	0.647	3.45	**6**	0.092	0.66
		4	**45**	0.734	5.71	**17**	0.436	2.44	**3**	0.009	0.57
	16	1	**0**	0.000	0.00	**78**	0.838	9.69	**10**	0.236	1.58
		2	**0**	0.000	0.00	**42**	0.719	6.11	**6**	0.090	1.40
		4	**0**	0.000	0.00	**23**	0.541	4.67	**4**	0.017	1.59
	32	1	**0**	0.000	0.00	**87**	0.853	14.88	**9**	0.198	3.98
		2	**0**	0.000	0.00	**48**	0.748	12.92	**6**	0.080	4.24
		4	**0**	0.000	0.00	**26**	0.587	10.77	**4**	0.019	5.74

FEAST

A flexible software package with special emphasis on:

- (almost) peak performance on recent and future hardware
- low storage requirements
- typical multigrid behaviour (efficiency/robustness)
- automatic expert system for solver selection
- parallelization directly included
- open for different adaptivity concepts
- pre– and postprocessing based on general java based framework **DeViSoR** and **AVS/Express** (free!!!)
- application tools for many “real life“ problems (FEASTFLOW for CFD)

Outlook

- first version of FEAST end 1998 expected

Further information on:

- FEAST home page

  `http://www.iwr.uni-heidelberg/featflow`

- The "Virtual Album of Flow Motion"

  `http://www.iwr.uni-heidelberg/featflow/album/book.html`