FEAST: The Future

(Finite Element Analysis & Solution Tools)

Christian Becker

christian.becker@math.uni-dortmund.de

Institut für Angewandte Mathematik und Numerik, LS3,
Universität Dortmund,
Germany

Overview

- Performance
- Solver
- FEAST structure

Performance

Where to calculate the world?

Today three kinds of so called ’Supercomputer’

Where to calculate the world?

Today three kinds of so called ’Supercomputer’

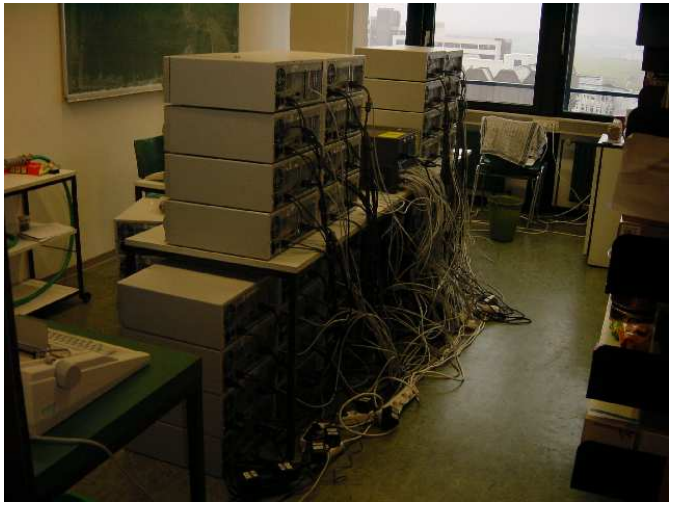
high cost ’classical’ parallel system:

Where to calculate the world?

Today three kinds of so called ’Supercomputer’

high cost ’classical’ parallel system:

low cost cluster system:

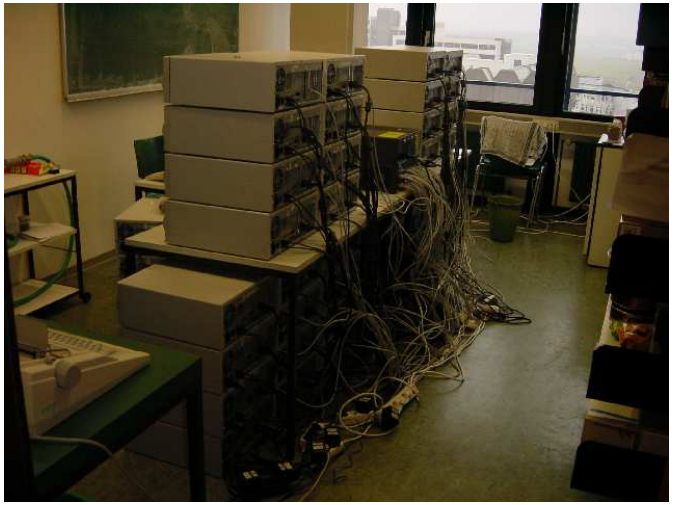


Where to calculate the world?

Today three kinds of so called ’Supercomputer’

high cost ’classical’ parallel system:

low cost cluster system:



vector system:

Problems

Problems to use these machines:

Problems

Problems to use these machines:

- complete new implementation

Problems

Problems to use these machines:

- complete new implementation
- explicit communication structure

Problems

Problems to use these machines:

- complete new implementation
- explicit communication structure
- synchronisation

Problems

Problems to use these machines:

- complete new implementation
- explicit communication structure
- synchronisation
- loadbalancing

Problems

Problems to use these machines:

- complete new implementation
- explicit communication structure
- synchronisation
- loadbalancing
- utilisation of vector capabilities and built in CPU features

Problems

Problems to use these machines:

- complete new implementation
- explicit communication structure
- synchronisation
- loadbalancing
- utilisation of vector capabilities and built in CPU features
- automatic parallelization but only efficient for #CPU < 8

Problems

Problems to use these machines:

- complete new implementation
- explicit communication structure
- synchronisation
- loadbalancing
- utilisation of vector capabilities and built in CPU features
- automatic parallelization but only efficient for #CPU < 8
- but no other way with respect to memory and CPU requirements for realistic numerical simulation

How to measure 'Performance'

Linpack test, Gaussian Elimination, benchmark for the TOP500 list

How to measure 'Performance'

Linpack test, Gaussian Elimination, benchmark for the TOP500 list

Rank	Name	Rmax $GFlop/s$	Site	#CPU
1	IBM ASCI White,SP Power3 375 MHz	7226	LLNL USA	8192
2	Compaq AlphaServer SC ES45/1 GHz	4059	Pittsburgh SC USA	3024
3	IBM SP Power3 375 MHz	3052	NERSC/LBNL USA	3328
10	IBM SP Power3 375 MHz	1293	DWD Germany	1280
17	Hitachi SR8000-F1/112	1035	Leibniz RZ Germany	112
21	Fujitsu VPP5000/100	886	ECMWF UK	100
30	Self-made CPlant/Ross Cluster	706	Sandia USA	1369
500	Cray Inc. T3E1200	94	EPA USA	116

Suitable Algorithm?

’Typical’: 3D Poisson problem

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^9$ FLOP
  100 sec on a 10 MFLOP/s computer

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^9$ FLOP
  100 sec on a 10 MFLOP/s computer
- $1000 \times 1000 \times 1000$ grid points $\longrightarrow$ problem size $N = 10^9$

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^9$ FLOP
  100 sec on a 10 MFLOP/s computer
- $1000 \times 1000 \times 1000$ grid points $\longrightarrow$ problem size $N = 10^9$
- Complexity of GE: $N^{7/3} \approx 10^{21}$ FLOP
  1,000,000 sec on a 1 PFLOP/s computer

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^9$ FLOP
  100 sec on a 10 MFLOP/s computer
- $1000 \times 1000 \times 1000$ grid points $\longrightarrow$ problem size $N = 10^9$
- Complexity of GE: $N^{7/3} \approx 10^{21}$ FLOP
  1,000,000 sec on a 1 PFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^{12}$ FLOP
  1,000 sec on a 1 GFLOP/s computer

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^9$ FLOP
  100 sec on a 10 MFLOP/s computer
- $1000 \times 1000 \times 1000$ grid points $\longrightarrow$ problem size $N = 10^9$
- Complexity of GE: $N^{7/3} \approx 10^{21}$ FLOP
  1,000,000 sec on a 1 PFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^{12}$ FLOP
  1,000 sec on a 1 GFLOP/s computer

$\Rightarrow$ **Questions:** Are only 10 MFLOP/s realistic in modern simulation packages?

Suitable Algorithm?

'Typical': 3D Poisson problem

- $100 \times 100 \times 100$ grid points $\longrightarrow$ problem size $N = 10^6$
- Complexity of GE: $N^{7/3} \approx 10^{14}$ FLOP
  100 sec on a 1 TFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^9$ FLOP
  100 sec on a 10 MFLOP/s computer
- $1000 \times 1000 \times 1000$ grid points $\longrightarrow$ problem size $N = 10^9$
- Complexity of GE: $N^{7/3} \approx 10^{21}$ FLOP
  1,000,000 sec on a 1 PFLOP/s computer
- Complexity of (opt.) multigrid: $1000N \approx 10^{12}$ FLOP
  1,000 sec on a 1 GFLOP/s computer

$\Rightarrow$ **Questions:** Are only 10 MFLOP/s realistic in modern simulation packages? What is "optimal" multigrid?

Performance of commercial codes

’Optimal’ multigrid $\sim$ 1 sec/subproblem/processor???

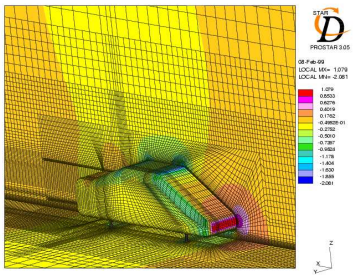
Performance of commercial codes

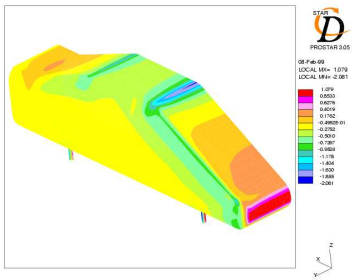
'Optimal' multigrid $\sim$ 1 sec/subproblem/processor???

But typical situation in high performance computing today:

Example: STAR-CD ($k-\epsilon$), 500 000 cells (by Daimer Chrysler), SGI Origin2000 (6 CPUs), 6.5 CPU h

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





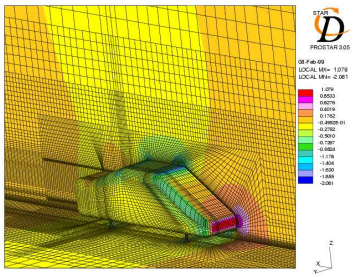
Performance of commercial codes

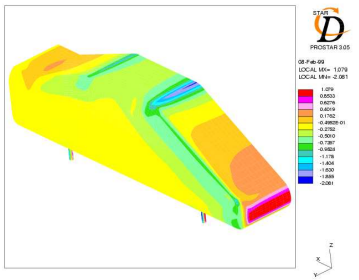
'Optimal' multigrid $\sim$ 1 sec/subproblem/processor???

But typical situation in high performance computing today:

Example: STAR-CD ($k-\epsilon$), 500 000 cells (by Daimer Chrysler), SGI Origin2000 (6 CPUs), 6.5 CPU h

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





- Are recent software able to benefit from the performance race?

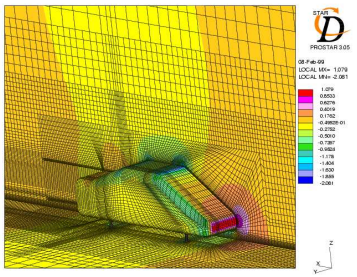
Performance of commercial codes

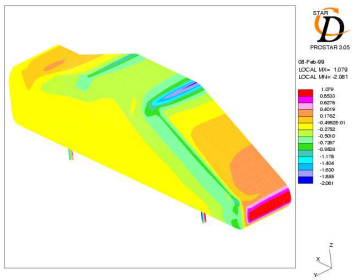
'Optimal' multigrid $\sim$ 1 sec/subproblem/processor???

But typical situation in high performance computing today:

Example: STAR-CD ($k-\epsilon$), 500 000 cells (by Daimer Chrysler), SGI Origin2000 (6 CPUs), 6.5 CPU h

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





- Are recent software able to benefit from the performance race?
- Are special strategies necessary to exploit the performance?

Influence of implementation

Linpack test, the same test used for TOP500 list

AMD Athlon XP 1500+ with 1333Mhz core frequency, measured MFlop/s

Influence of implementation

Linpack test, the same test used for TOP500 list

AMD Athlon XP 1500+ with 1333Mhz core frequency, measured MFlop/s

- Case 1: self compiled BLAS/LAPACK libraries

Influence of implementation

Linpack test, the same test used for TOP500 list

AMD Athlon XP 1500+ with 1333Mhz core frequency, measured MFlop/s

- Case 1: self compiled BLAS/LAPACK libraries
- Case 2: vendor delivered BLAS/LAPACK, partially assembler

Influence of implementation

Linpack test, the same test used for TOP500 list

AMD Athlon XP 1500+ with 1333Mhz core frequency, measured MFlop/s

- Case 1: self compiled BLAS/LAPACK libraries
- Case 2: vendor delivered BLAS/LAPACK, partially assembler
- Case 3: special tuned BLAS/LAPACK, with use of CPU extensions like SSE

Influence of implementation

Linpack test, the same test used for TOP500 list

AMD Athlon XP 1500+ with 1333Mhz core frequency, measured MFlop/s

- Case 1: self compiled BLAS/LAPACK libraries
- Case 2: vendor delivered BLAS/LAPACK, partially assembler
- Case 3: special tuned BLAS/LAPACK, with use of CPU extensions like SSE

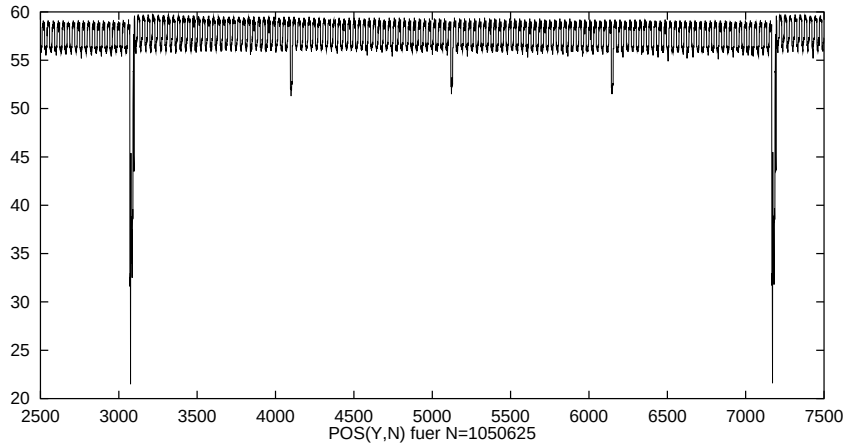
N	case 1	case 2	case 3
500	323	477	940
1000	275	481	1119
3000	116	152	1362

Cache influence

Cache test vector addition (variation of the distance of the 2 vectors)

Cache influence

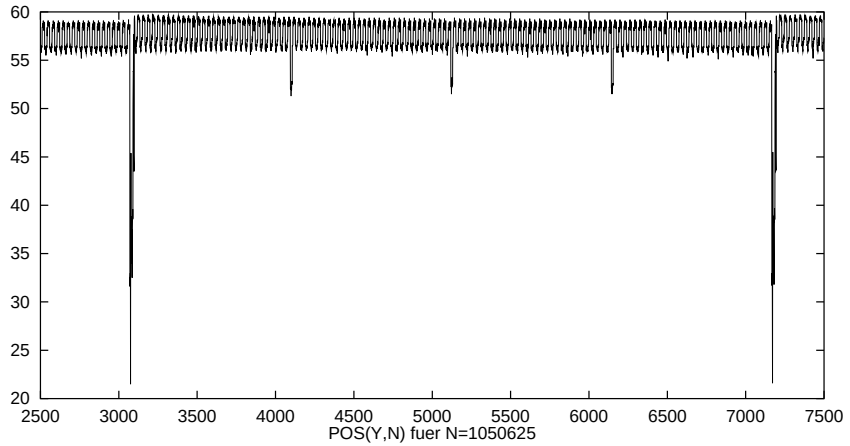
Cache test vector addition (variation of the distance of the 2 vectors)



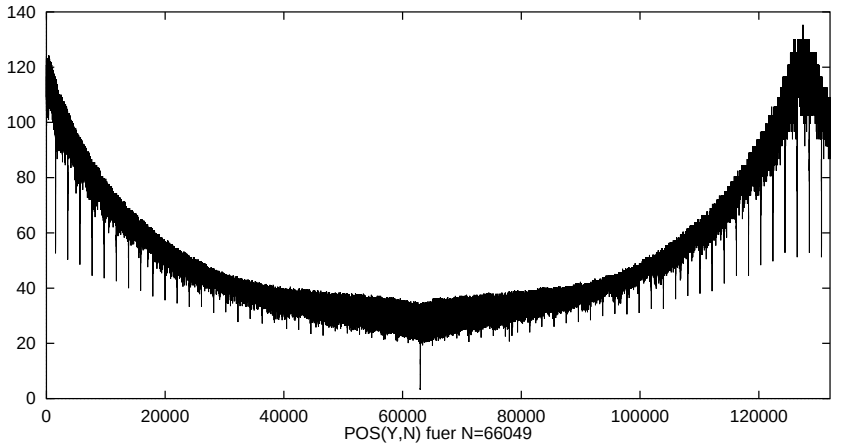
Cray T3E:

Cache influence

Cache test vector addition (variation of the distance of the 2 vectors)



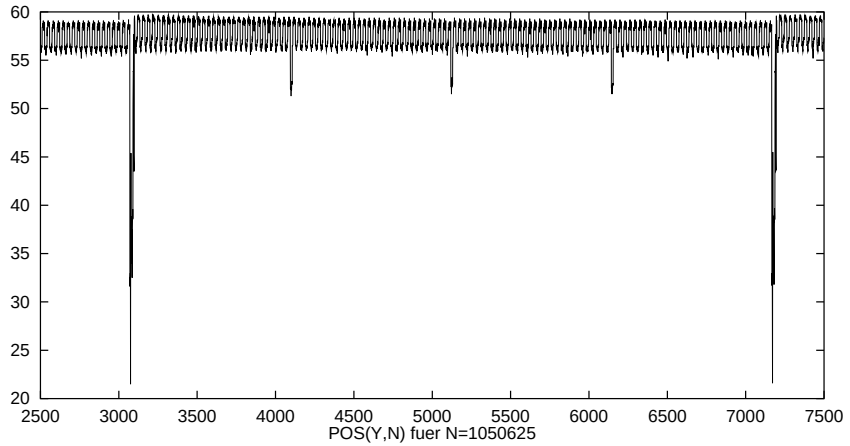
Cray T3E:



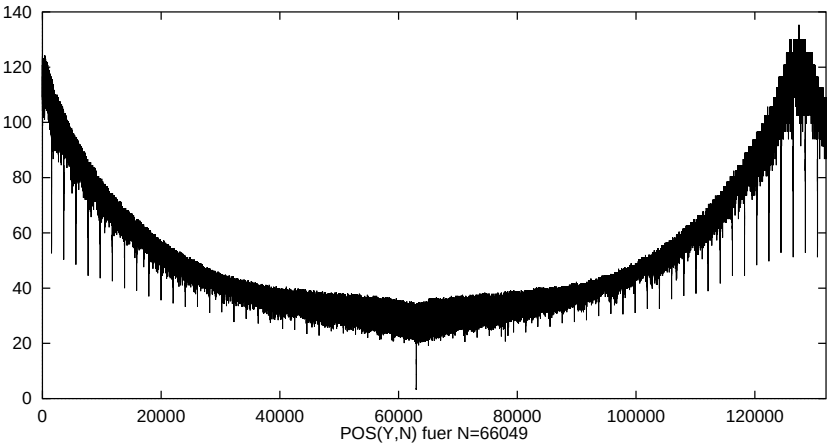
Sun U450

Cache influence

Cache test vector addition (variation of the distance of the 2 vectors)



Cray T3E:

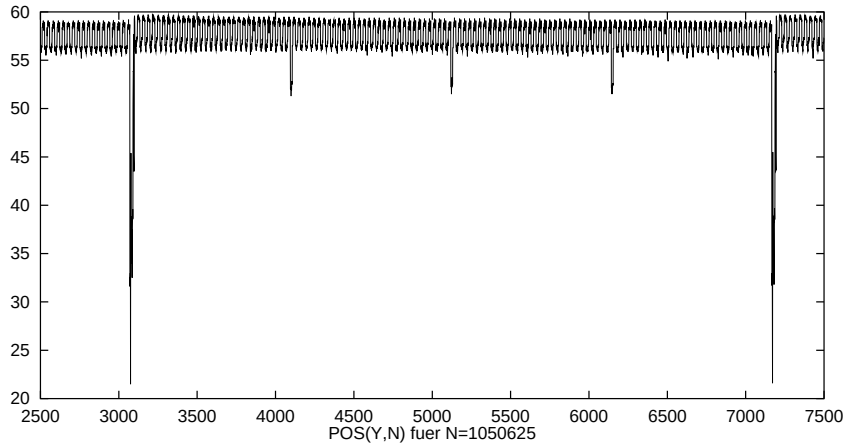


Sun U450

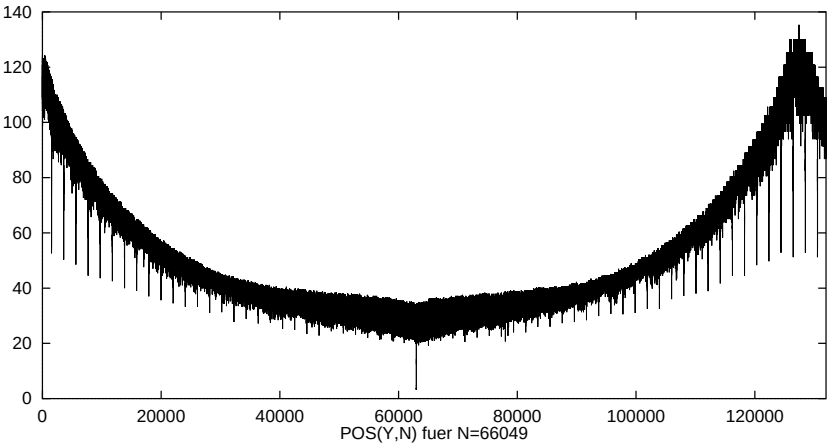
- performance highly dependent of memory position

Cache influence

Cache test vector addition (variation of the distance of the 2 vectors)



Cray T3E:

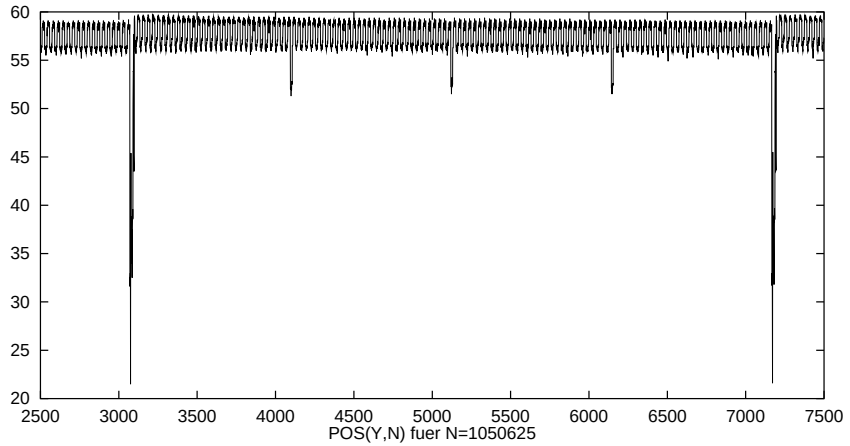


Sun U450

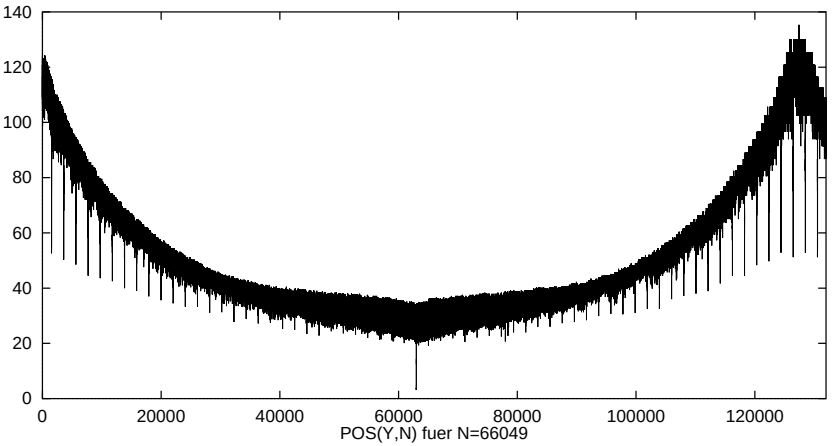
- performance highly dependent of memory position
- small ’windows’ of higher performance

Cache influence

Cache test vector addition (variation of the distance of the 2 vectors)



Cray T3E:



Sun U450

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

Sparse Matrix Vector Multiplication

Standard sparse matrix vector algorithm:

(DAXPY indexed)

```
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 Matrix Vector Multiplication

Standard sparse matrix vector algorithm:

(DAXPY indexed)

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

Performance rates of FEATFLOW with different numbering schemes (Cuthill–McKee, TwoLevel, Stochastic) for matrix vector multiplication:

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

Sparse Matrix Vector Multiplication

Standard sparse matrix vector algorithm:

(DAXPY indexed)

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

Performance rates of FEATFLOW with different numbering schemes (Cuthill–McKee, TwoLevel, Stochastic) for matrix vector multiplication:

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

- sparse techniques basis for most of the recent packages

Sparse Matrix Vector Multiplication

Standard sparse matrix vector algorithm:

(DAXPY indexed)

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

Performance rates of FEATFLOW with different numbering schemes (Cuthill–McKee, TwoLevel, Stochastic) for matrix vector multiplication:

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

- sparse techniques basis for most of the recent packages
- different numberings can lead to identical numerical results and work (w.r.t. arith.ops and data accesses) but to huge differences in CPU time

Sparse Matrix Vector Multiplication

Standard sparse matrix vector algorithm:

(DAXPY indexed)

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

Performance rates of FEATFLOW with different numbering schemes (Cuthill–McKee, TwoLevel, Stochastic) for matrix vector multiplication:

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

- sparse techniques basis for most of the recent packages
- different numberings can lead to identical numerical results and work (w.r.t. arith.ops and data accesses) but to huge differences in CPU time
- sparse techniques 'slow' and depend on problem size and kind of data access

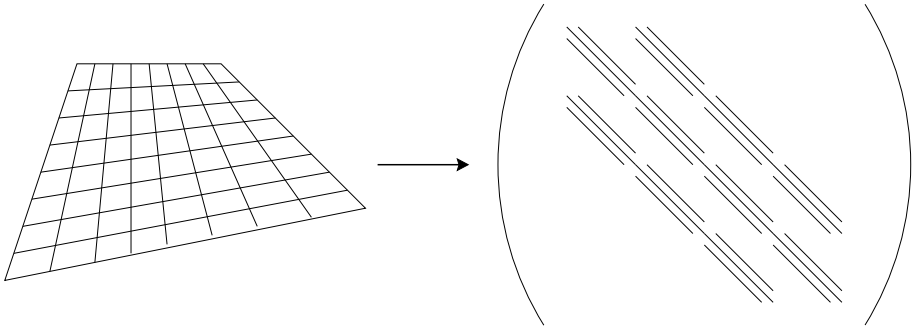
Alt.: Sparse Banded Techniques

Question: How to exploit more performance?

Alt.: Sparse Banded Techniques

Question: How to exploit more performance?

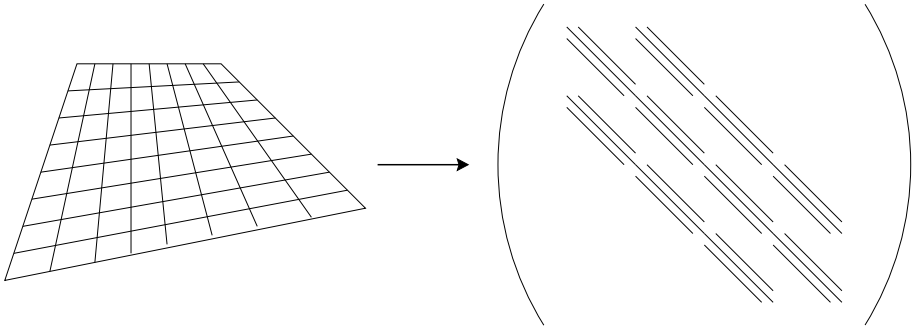
Line- or rowwise numbering:



Alt.: Sparse Banded Techniques

Question: How to exploit more performance?

Line- or rowwise numbering:



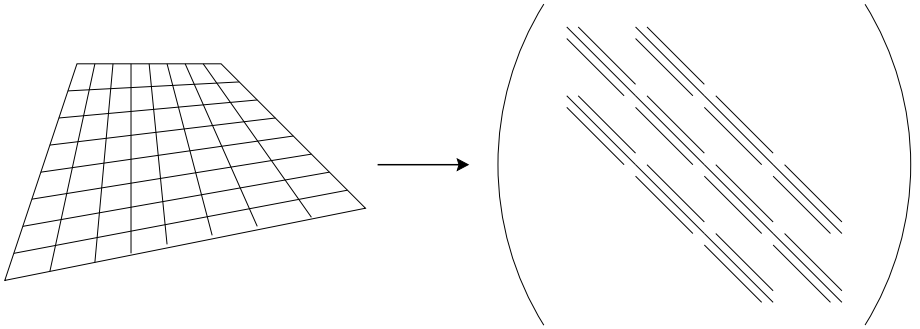
Sparse *banded* matrix vector multiplication:

- FD discr. leads to band structure on tensorproduct meshes

Alt.: Sparse Banded Techniques

Question: How to exploit more performance?

Line- or rowwise numbering:



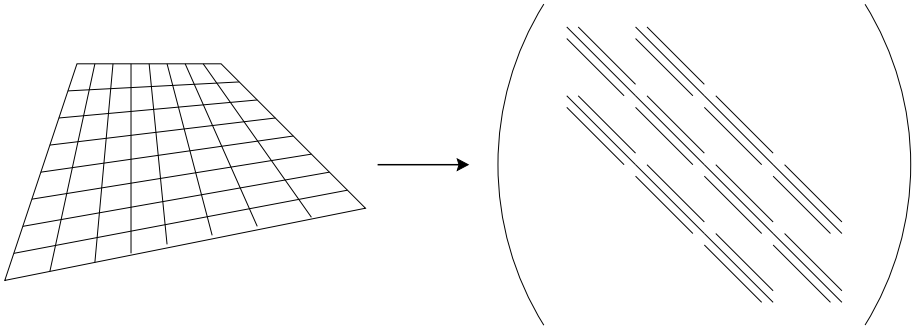
Sparse *banded* matrix vector multiplication:

- FD discr. leads to band structure on tensorproduct meshes
- storing of matrix elements in diagonals

Alt.: Sparse Banded Techniques

Question: How to exploit more performance?

Line- or rowwise numbering:



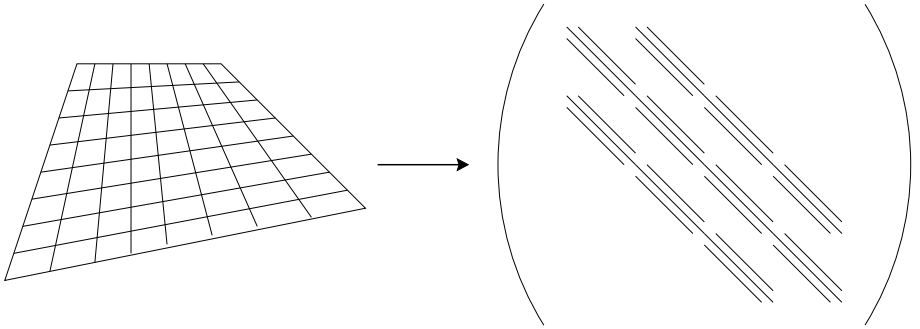
Sparse *banded* matrix vector multiplication:

- FD discr. leads to band structure on tensorproduct meshes
- storing of matrix elements in diagonals
- matrix vector multiplication 'bandwise'

Alt.: Sparse Banded Techniques

Question: How to exploit more performance?

Line- or rowwise numbering:

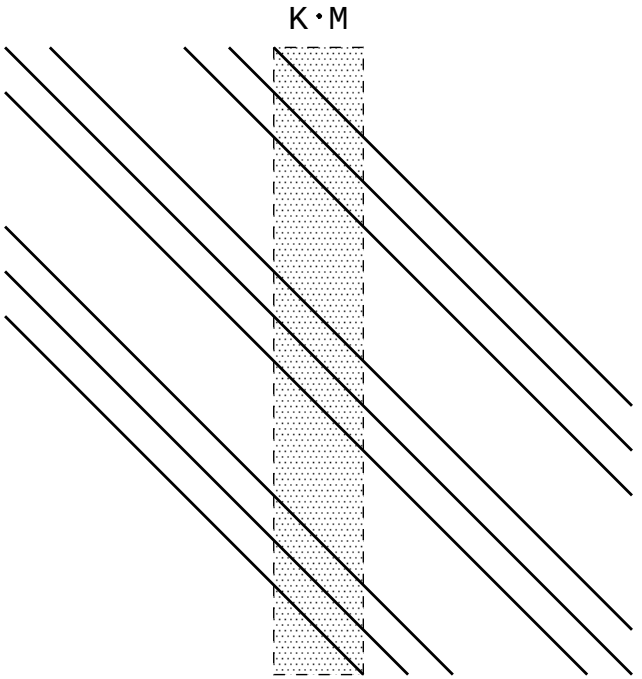


Sparse *banded* matrix vector multiplication:

- FD discr. leads to band structure on tensorproduct meshes
- storing of matrix elements in diagonals
- matrix vector multiplication 'bandwise'
- in equidistant case for certain operators diagonals are constant

Sparse Banded Techniques I

Bandwise windowed multiplication (variable, constant):




```
DO 1 IM=1,M/K
  DO 2 I=1,K*M
2 Y(I) =Y(I) +DD(I)*X(I)+DL(I)*X(I-1)+DU(I)*X(I+1)
  DO 3 I =1,K*M
3 Y(I-M)=Y(I-M)+LD(I)*X(I)+LL(I)*X(I-1)+LU(I)*X(I+1)
  DO 4 I=1,K*M
4 Y(I+M)=Y(I+M)+UD(I)*X(I)+UL(I)*X(I-1)+UU(I)*X(I+1)
1 CONTINUE
```

Sparse Banded Techniques II

2D case	N	DAXPY-I	SBB-V	SBB-C	**MG-V**	**MG-C**
DEC 21264	65^2	205 (178)	538	795	370	452
(667 MHz)	257^2	224 (110)	358	1010	314	487
'**ES40**'	1025^2	**78 (11)**	**158**	**813**	**185**	**401**
HITACHI	65^2	173 (82)	238	391	191	266
(375 MHz)	257^2	143 (29)	243	388	198	260
'**SR8000**'	1025^2	**144 (7)**	**226**	**390**	**200**	**267**
AMD K7	65^2	203 (195)	101	556	122	355
(850 MHz)	257^2	29 (27)	78	241	72	166
'**ATHLON**'	1025^2	**31 (10)**	**64**	**236**	**58**	**126**

Sparse Banded Techniques II

2D case	N	DAXPY-I	SBB-V	SBB-C	**MG-V**	**MG-C**
DEC 21264	65^2	205 (178)	538	795	370	452
(667 MHz)	257^2	224 (110)	358	1010	314	487
'**ES40**'	1025^2	**78 (11)**	**158**	**813**	**185**	**401**
HITACHI	65^2	173 (82)	238	391	191	266
(375 MHz)	257^2	143 (29)	243	388	198	260
'**SR8000**'	1025^2	**144 (7)**	**226**	**390**	**200**	**267**
AMD K7	65^2	203 (195)	101	556	122	355
(850 MHz)	257^2	29 (27)	78	241	72	166
'**ATHLON**'	1025^2	**31 (10)**	**64**	**236**	**58**	**126**

Question: How to use these techniques on complex domains?

Sparse Banded Techniques II

2D case	N	DAXPY-I	SBB-V	SBB-C	**MG-V**	**MG-C**
DEC 21264	65^2	205 (178)	538	795	370	452
(667 MHz)	257^2	224 (110)	358	1010	314	487
'**ES40**'	1025^2	**78 (11)**	**158**	**813**	**185**	**401**
HITACHI	65^2	173 (82)	238	391	191	266
(375 MHz)	257^2	143 (29)	243	388	198	260
'**SR8000**'	1025^2	**144 (7)**	**226**	**390**	**200**	**267**
AMD K7	65^2	203 (195)	101	556	122	355
(850 MHz)	257^2	29 (27)	78	241	72	166
'**ATHLON**'	1025^2	**31 (10)**	**64**	**236**	**58**	**126**

Question: How to use these techniques on complex domains?

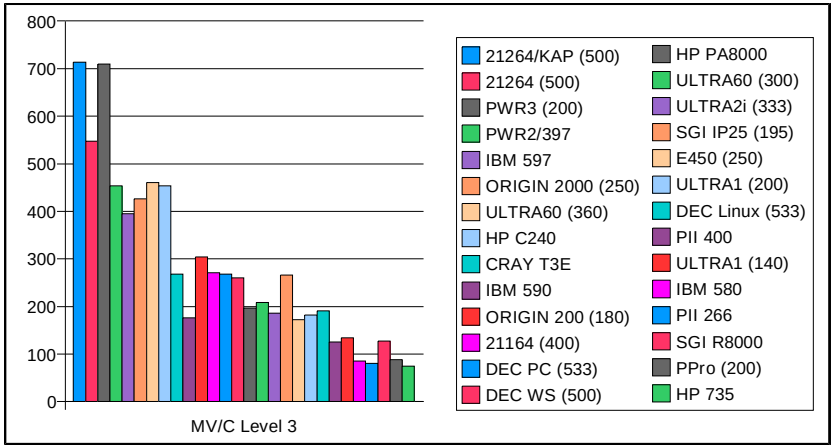
$\Rightarrow$ **ScaRC**

FEAST Indices I

Framework of different numerical linear algebra tests for different problem sizes, using the Sparse Banded BLAS routines

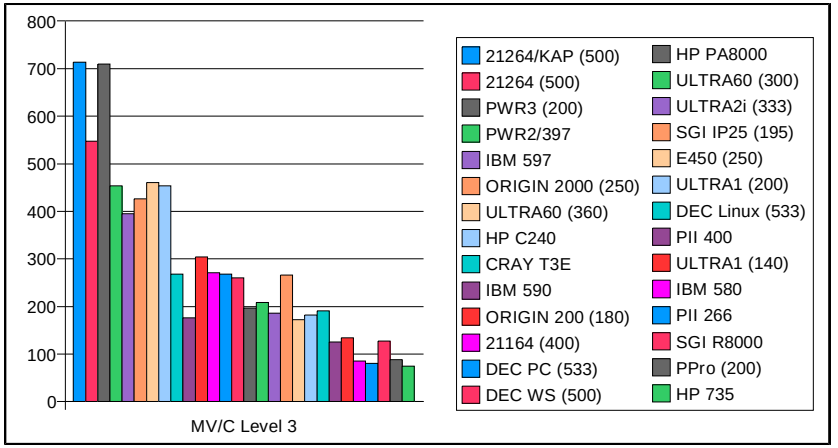
FEAST Indices I

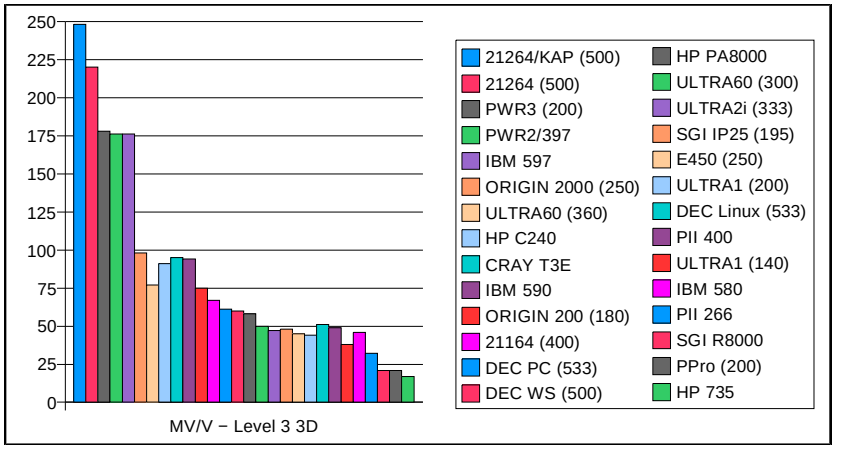
Framework of different numerical linear algebra tests for different problem sizes, using the Sparse Banded BLAS routines



FEAST Indices I

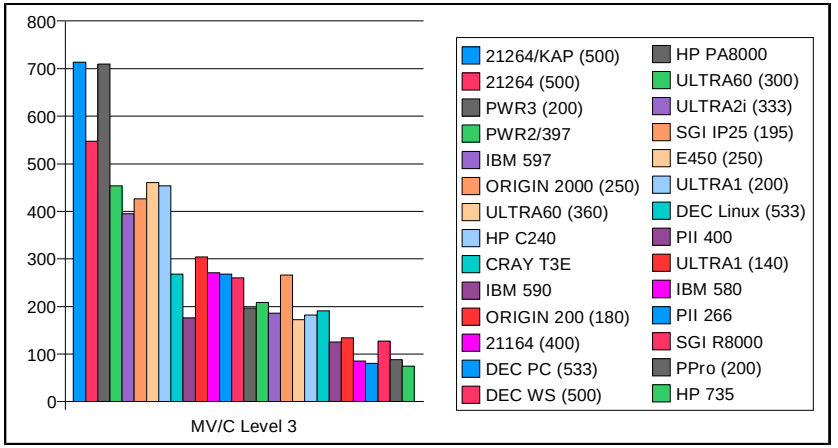
Framework of different numerical linear algebra tests for different problem sizes, using the Sparse Banded BLAS routines

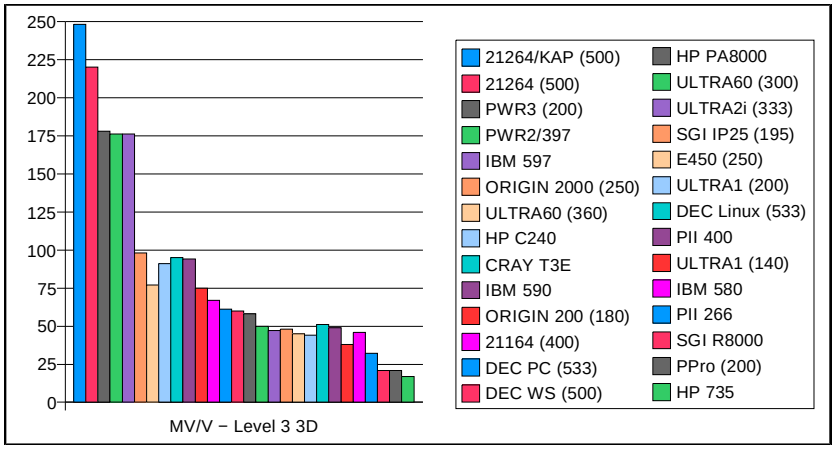


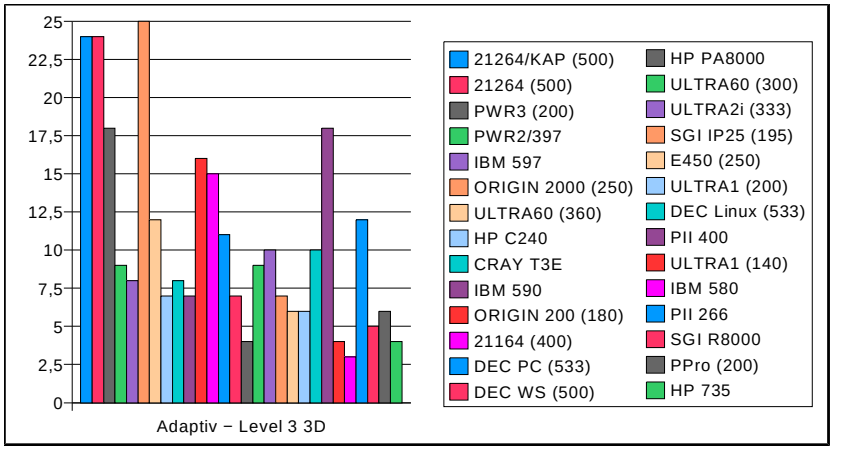


FEAST Indices I

Framework of different numerical linear algebra tests for different problem sizes, using the Sparse Banded BLAS routines

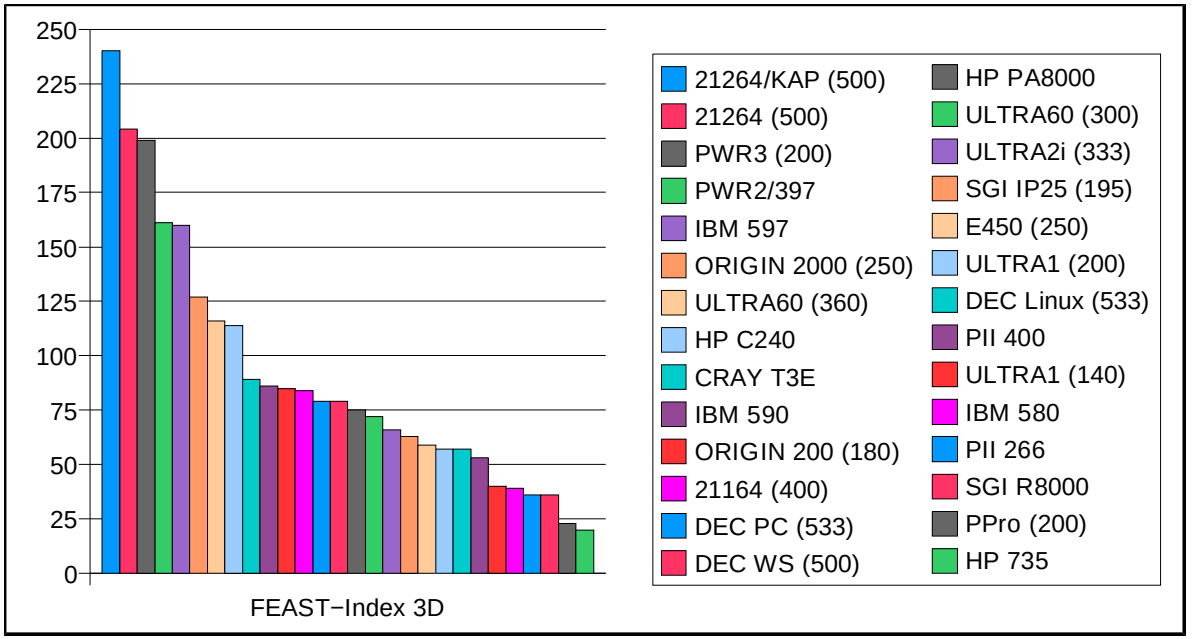




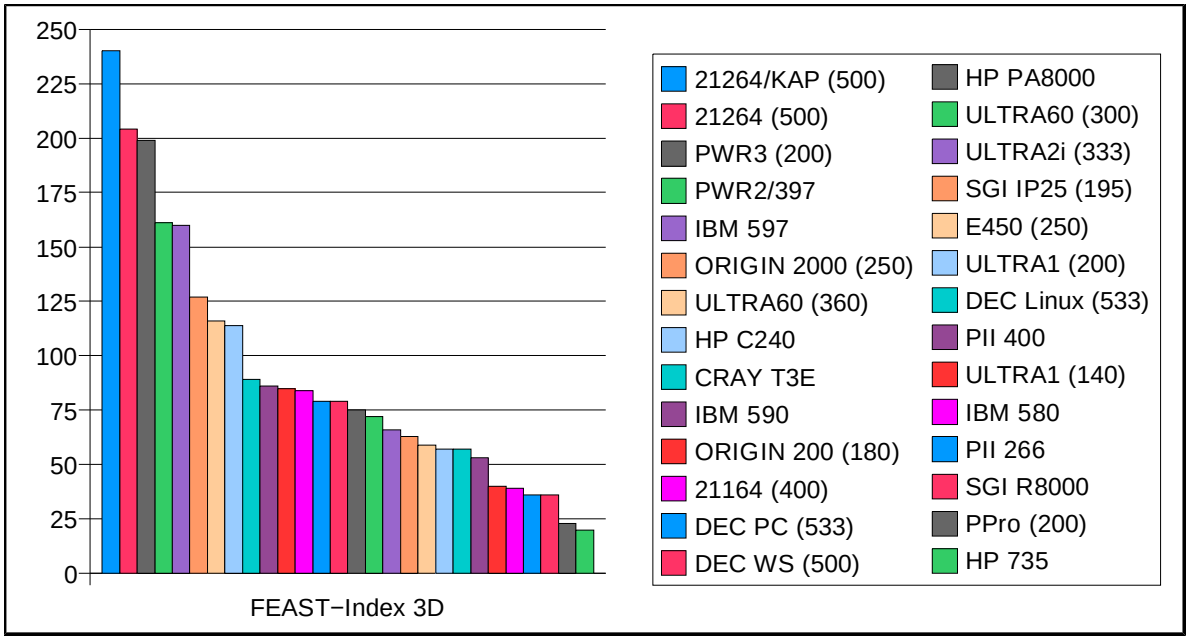


FEAST Indices II

Observations:



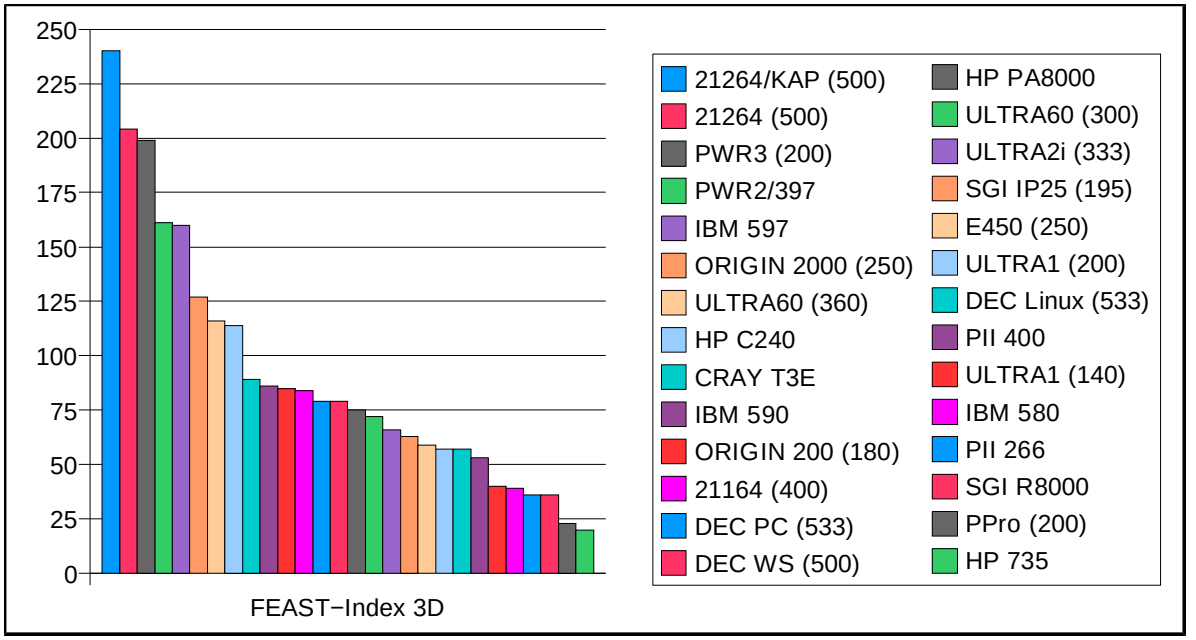
FEAST Indices II



Observations:

- MFLOP rate for standard sparse techniques far away from peak performance, depending on problem size and numbering of the unknowns

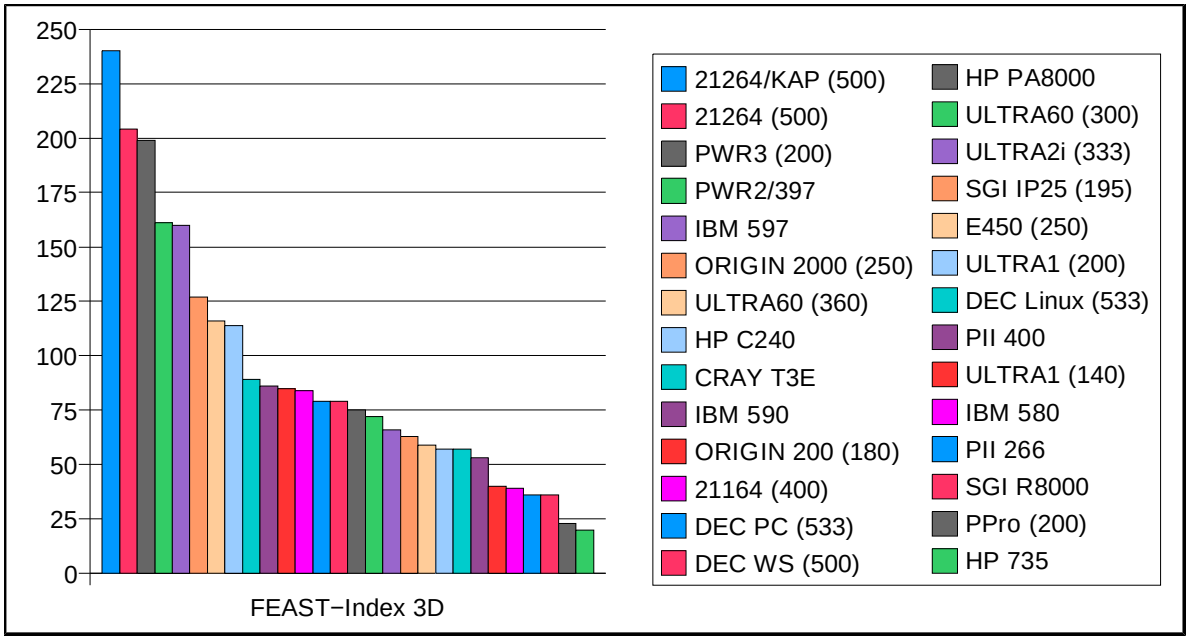
FEAST Indices II



Observations:

- MFLOP rate for standard sparse techniques far away from peak performance, depending on problem size and numbering of the unknowns
- higher performance achievable by appropriate techniques

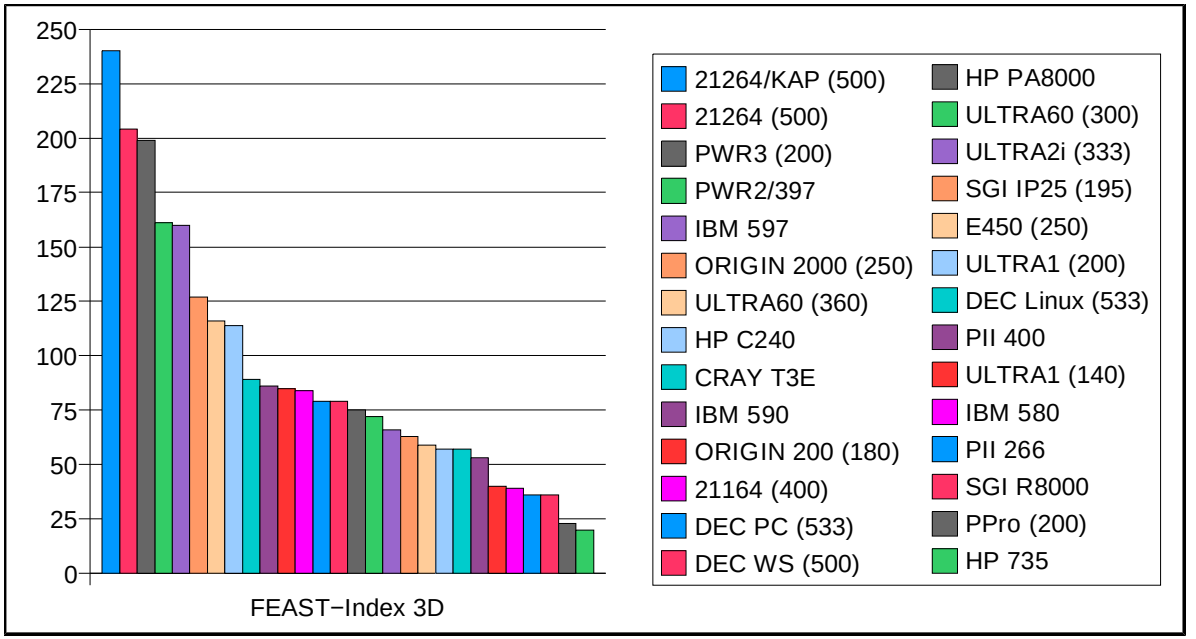
FEAST Indices II



Observations:

- MFLOP rate for standard sparse techniques far away from peak performance, depending on problem size and numbering of the unknowns
- higher performance achievable by appropriate techniques
- realization highly architecture dependent (processor, operating system, compiler version), no easy way

FEAST Indices II



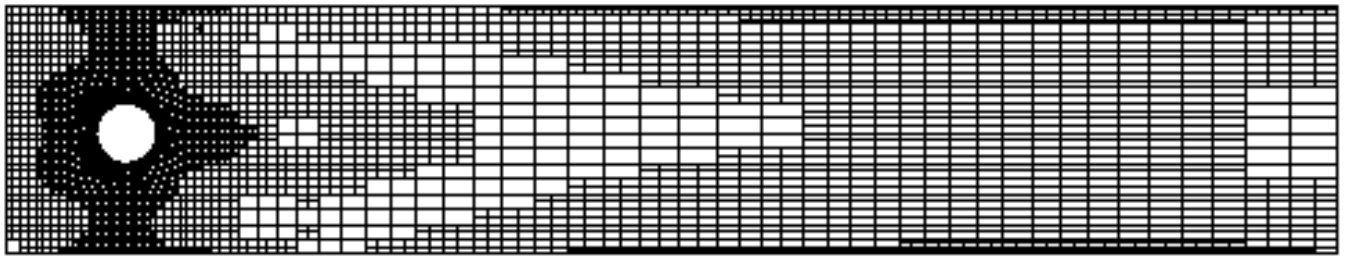
Observations:

- MFLOP rate for standard sparse techniques far away from peak performance, depending on problem size and numbering of the unknowns
- higher performance achievable by appropriate techniques
- realization highly architecture dependent (processor, operating system, compiler version), no easy way
- simple PC partially faster than supercomputer nodes

Solver

Hierarchical Divide and Conquer

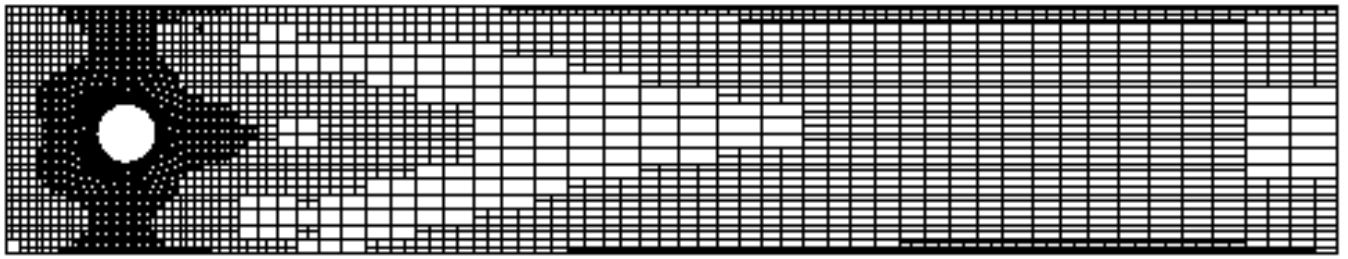
Optimal grid for drag–coefficent via a posteriori error estimation:



(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

Hierarchical Divide and Conquer

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

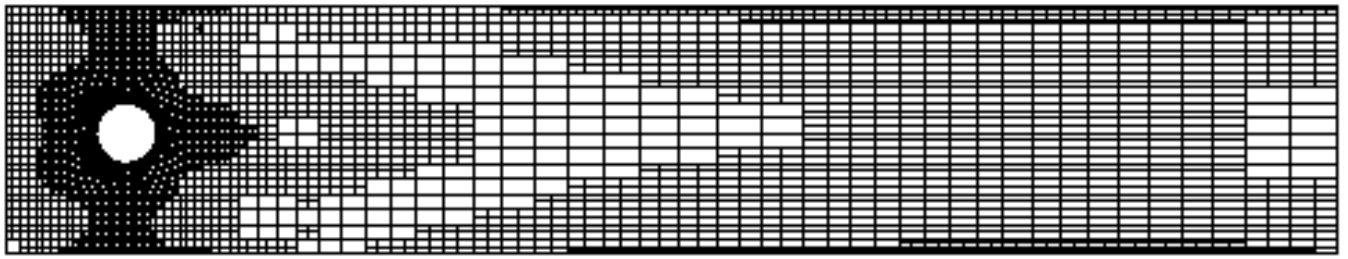


(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

- adaptive techniques only locally near boundaries

Hierarchical Divide and Conquer

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

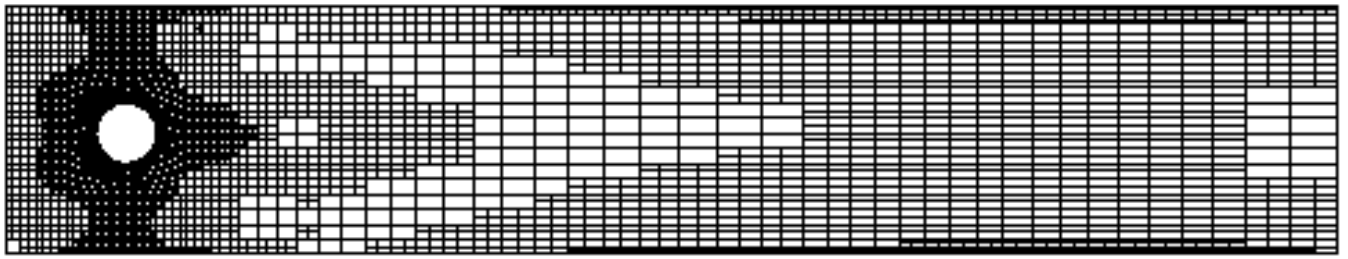


(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

- adaptive techniques only locally near boundaries
- regular substructures (90 %) in interior

Hierarchical Divide and Conquer

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

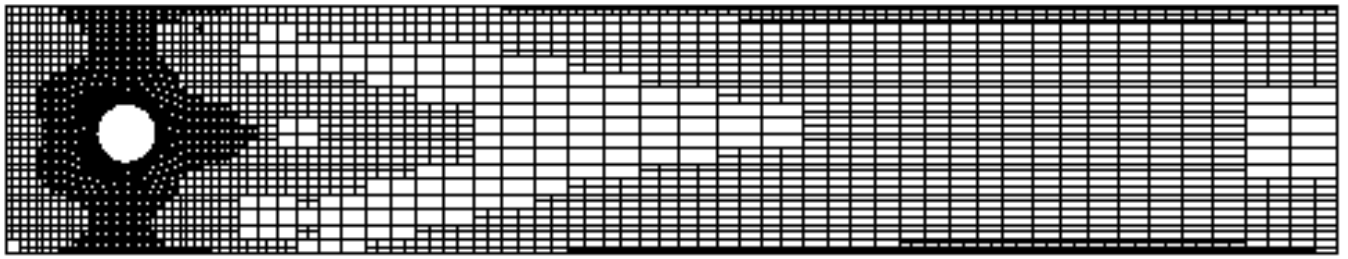


(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

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

Hierarchical Divide and Conquer

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

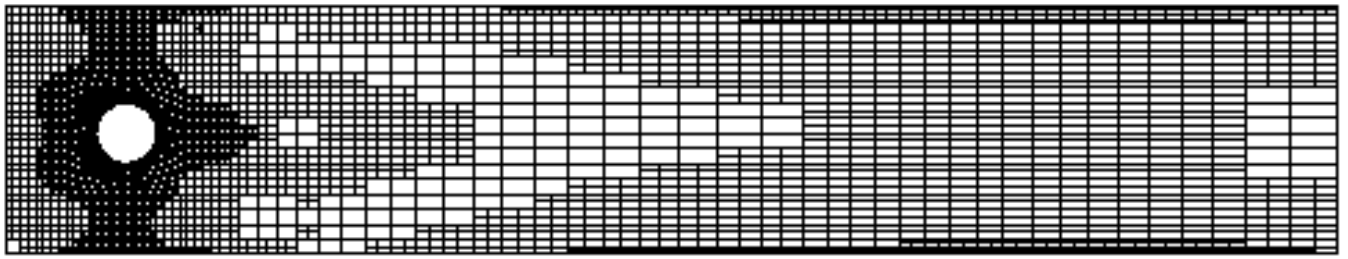


(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

- adaptive techniques only locally near boundaries
- regular substructures (90 %) in interior
- typical example for CFD
- recursive **Divide and Conquer** strategy

Hierarchical Divide and Conquer

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



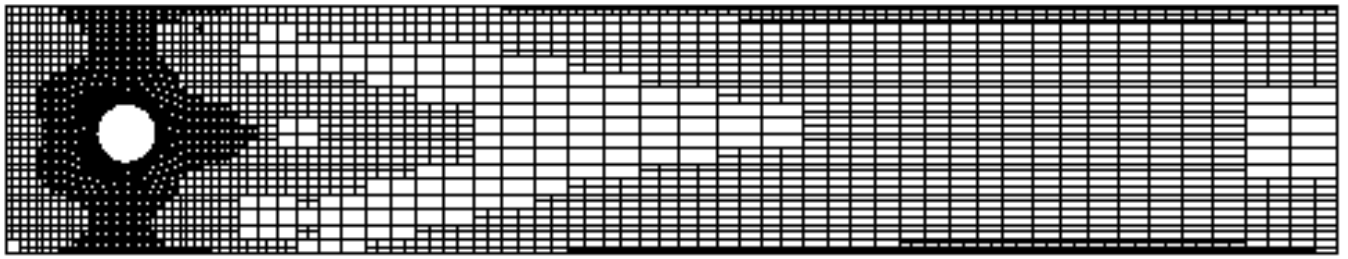
(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

- 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)

Hierarchical Divide and Conquer

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

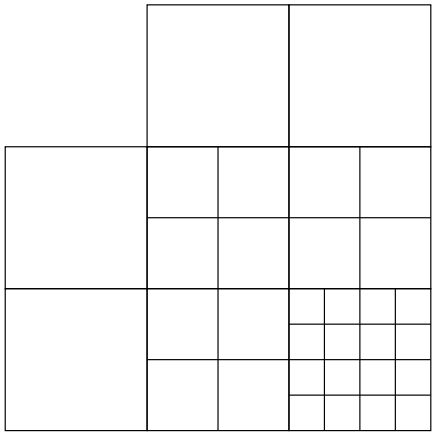


(ROLAND BECKER, INST.F.ANG.MATH., UNIVERSITÄT HEIDELBERG)

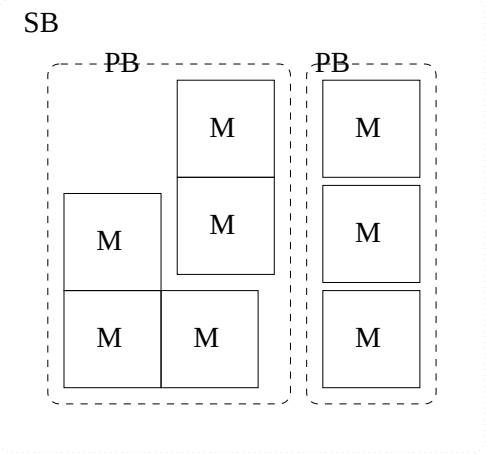
- adaptive techniques only locally near boundaries
- regular substructures (90 %) in interior
- typical example for CFD

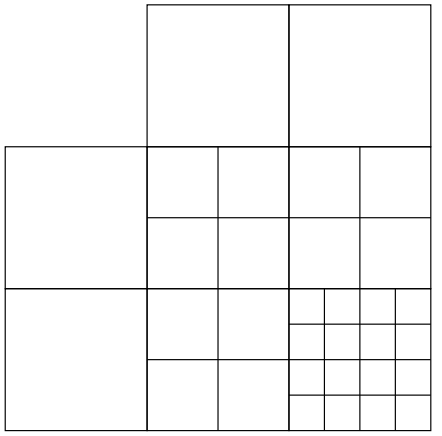
- recursive **Divide and Conquer** strategy
- find locally structured parts (high performance)
- find locally anisotropic parts (hide locally, robustness)

Hierachical Structures

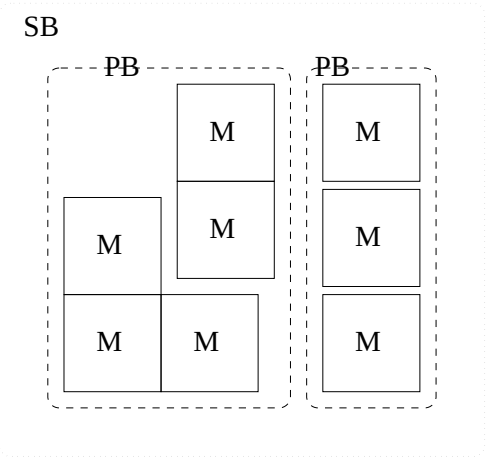


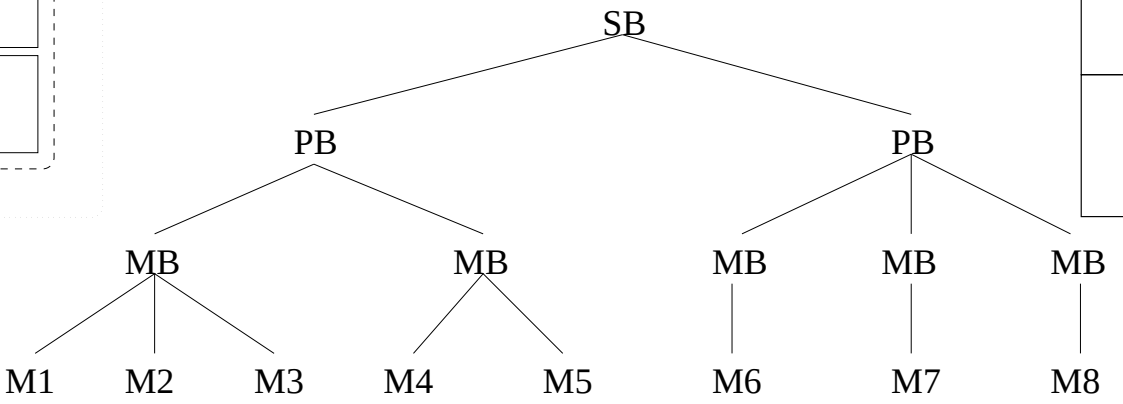
Hierachical Structures

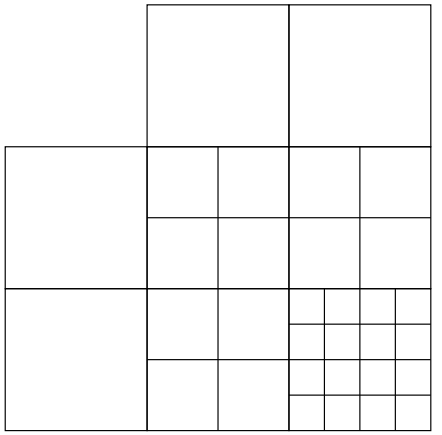




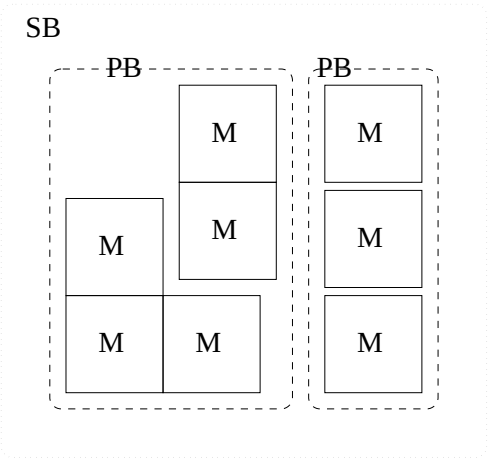
Hierachical Structures

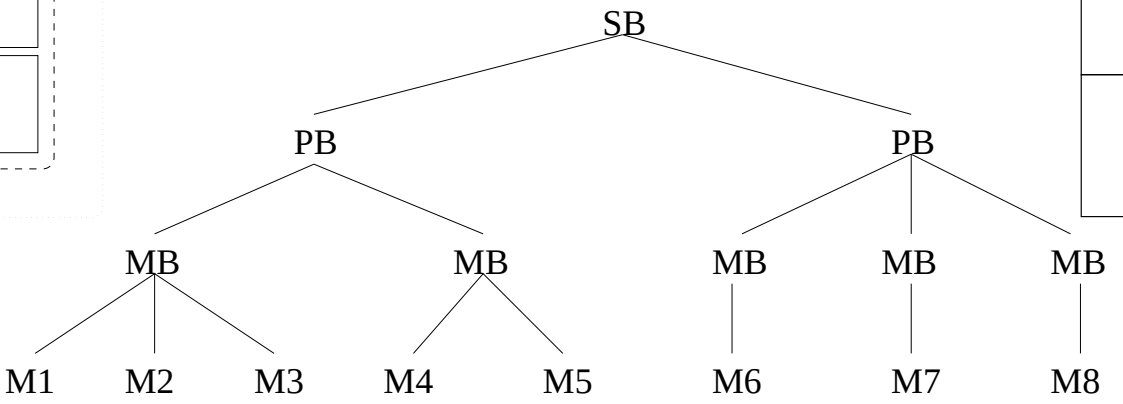


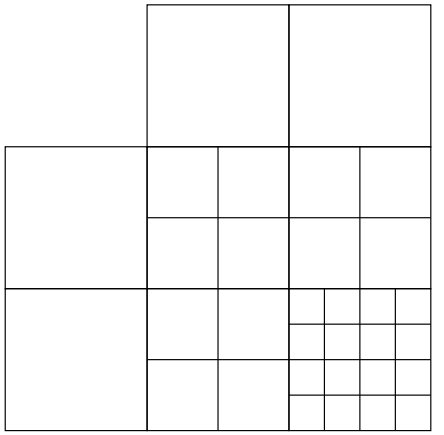




Hierachical Structures

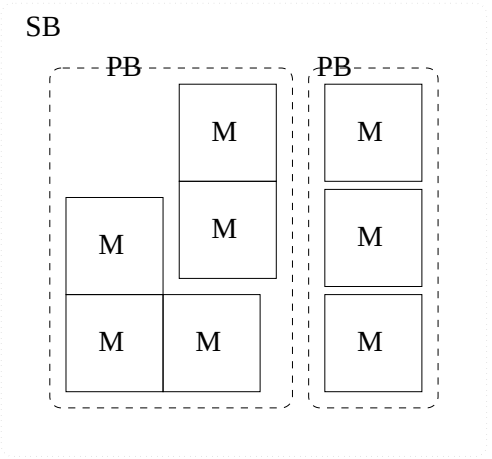


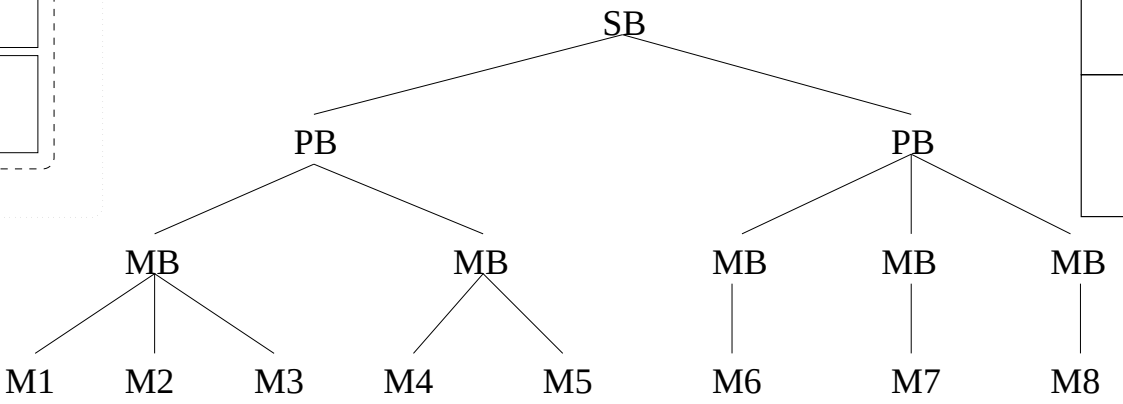


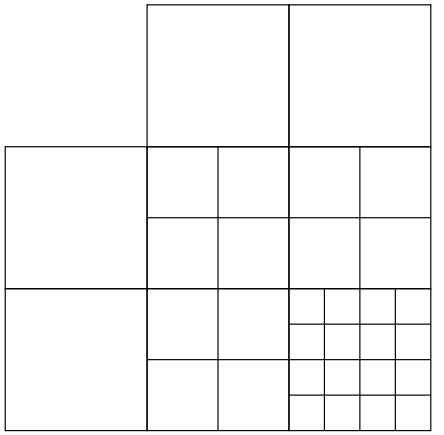


Structure parts: atomic units, matrix units, parallel units, subdomains, domain

Hierachical Structures



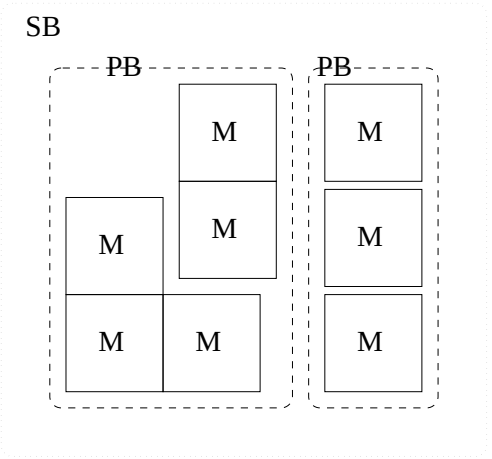


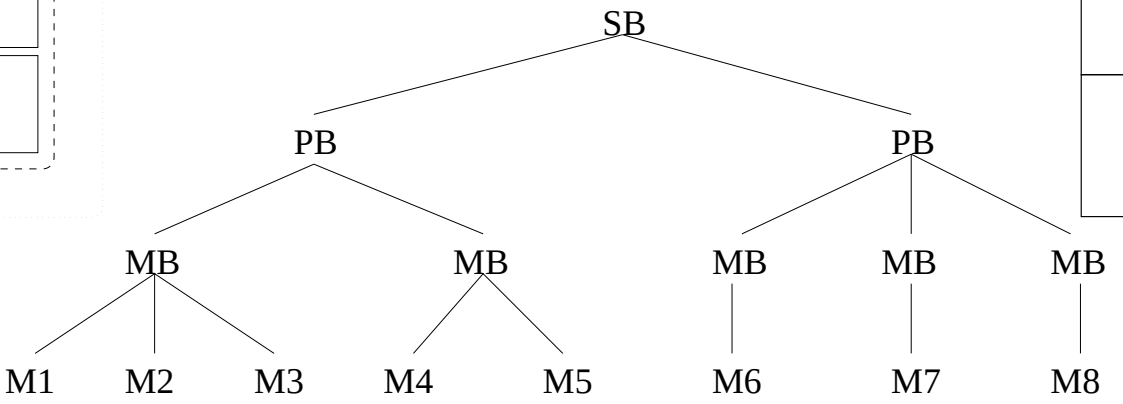


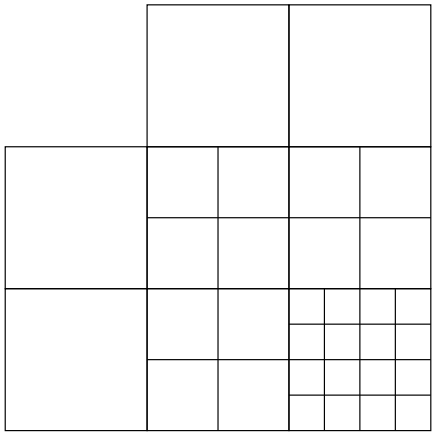
Structure parts: atomic units, matrix units, parallel units, subdomains, domain

- regular structures

Hierachical Structures



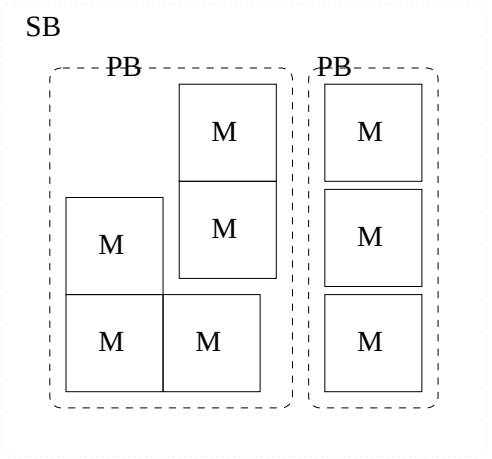


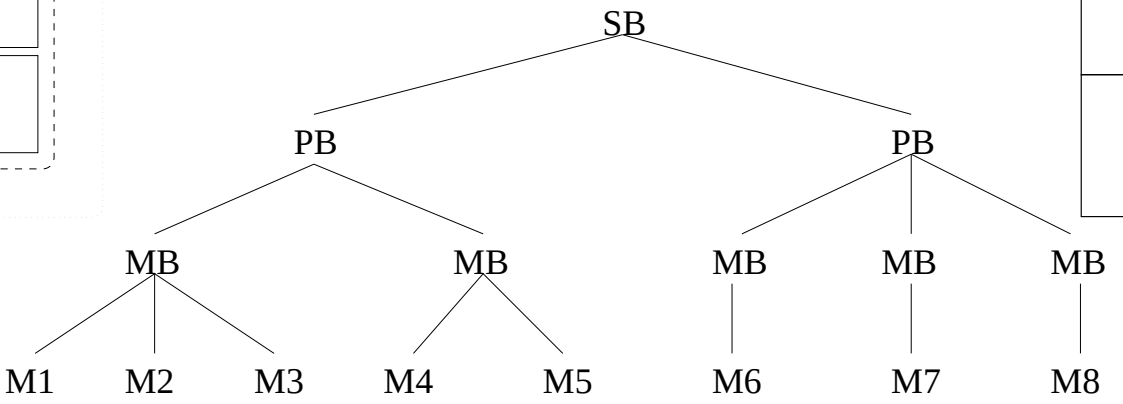


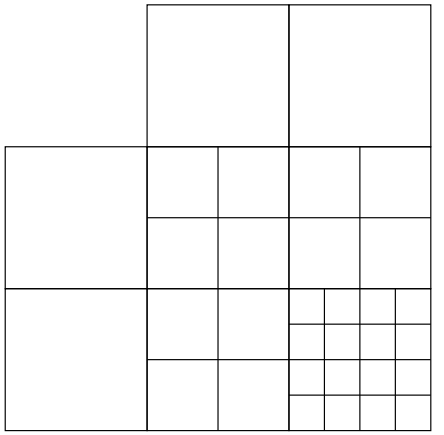
Structure parts: atomic units, matrix units, parallel units, subdomains, domain

- regular structures
- high performance linear algebra possible

Hierachical Structures



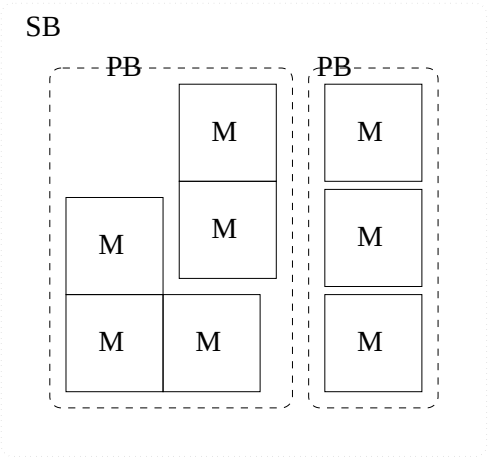


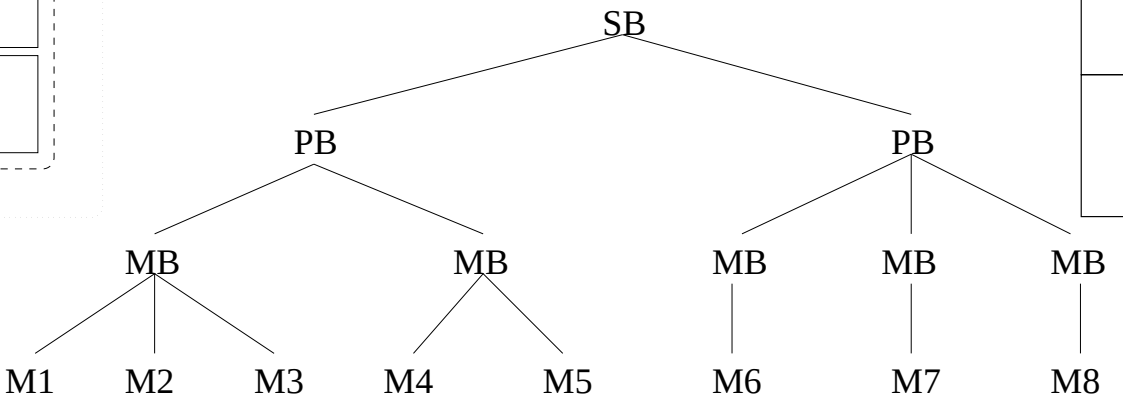


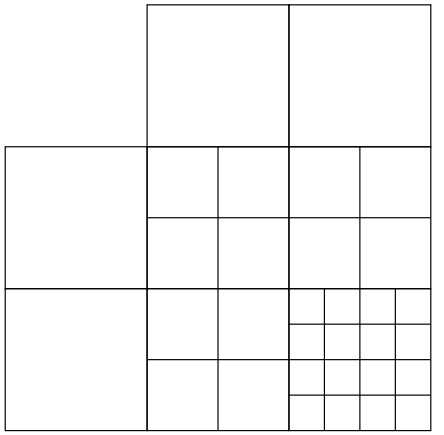
Structure parts: atomic units, matrix units, parallel units, subdomains, domain

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

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?

Concepts for iterative parallel solvers

1. Standard multigrid

2. Standard Domain Decomposition

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of ’recursive’ smoothers only blockwise?

2. Standard Domain Decomposition

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of ’recursive’ smoothers only blockwise?
   - complicated geometries with local anisotropies?

2. Standard Domain Decomposition

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of ’recursive’ smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of ’recursive’ smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition
   - good ratio for communication and arithmetic work

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of ’recursive’ smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition
   - good ratio for communication and arithmetic work
   - implementation (overlap, coarse grid problem, 3D)?

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of ’recursive’ smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition
   - good ratio for communication and arithmetic work
   - implementation (overlap, coarse grid problem, 3D)?
   - bad convergence rate w.r.t. multigrid

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of 'recursive' smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition
   - good ratio for communication and arithmetic work
   - implementation (overlap, coarse grid problem, 3D)?
   - bad convergence rate w.r.t. multigrid

$\Longrightarrow$ ScaRC (Scalable Recursive Clustering)

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of 'recursive' smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition
   - good ratio for communication and arithmetic work
   - implementation (overlap, coarse grid problem, 3D)?
   - bad convergence rate w.r.t. multigrid

$\Longrightarrow$ ScaRC (Scalable Recursive Clustering)

- Hide recursively alle anisotropies in more 'local' units (**robustness**)

Concepts for iterative parallel solvers

1. Standard multigrid
   - parallelization of 'recursive' smoothers only blockwise?
   - complicated geometries with local anisotropies?
   - too few arithmetic vs. data exchange
2. Standard Domain Decomposition
   - good ratio for communication and arithmetic work
   - implementation (overlap, coarse grid problem, 3D)?
   - bad convergence rate w.r.t. multigrid

$\Longrightarrow$ ScaRC (Scalable Recursive Clustering)

- Hide recursively alle anisotropies in more 'local' units (**robustness**)
- Perform most of Linear Algebra tasks on 'local' units (**efficiency**)

ScaRC (Scalable Recursive Clustering)

Standard multigrid with
(recursively defined)
block smoothers

ScaRC (Scalable Recursive Clustering)

Standard multigrid with
(recursively defined)
block smoothers

plus

Standard Domain Decomposition
with minimal overlap,
sequence of coarse grid
problems via multigrid

ScaRC (Scalable Recursive Clustering)

Standard multigrid with
(recursively defined)
block smoothers

plus

Standard Domain Decomposition
with minimal overlap,
sequence of coarse grid
problems via multigrid

plus

Embedded into
standard CG–method

ScaRC (Scalable Recursive Clustering)

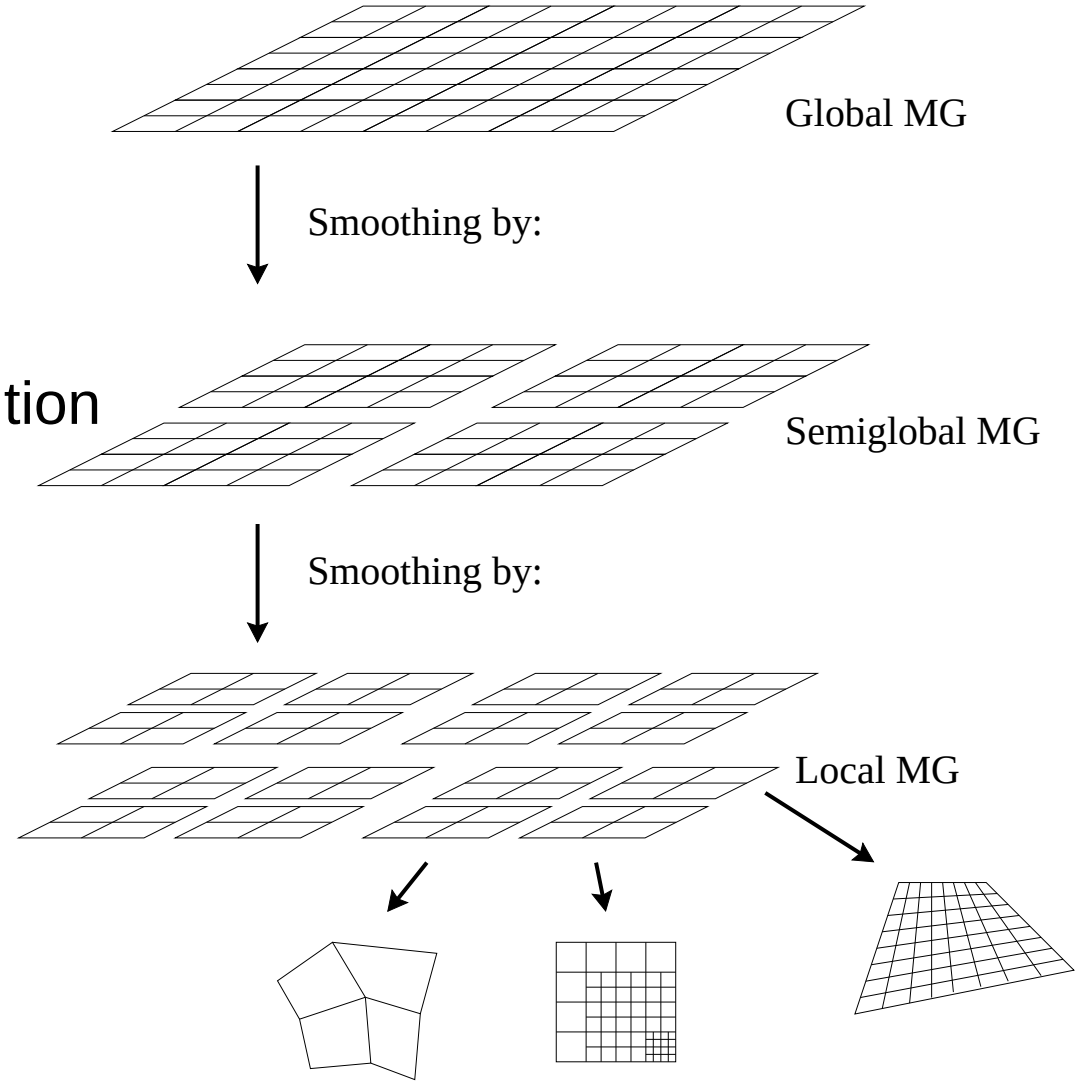
Standard multigrid with (recursively defined) block smoothers

plus

Standard Domain Decomposition with minimal overlap, sequence of coarse grid problems via multigrid

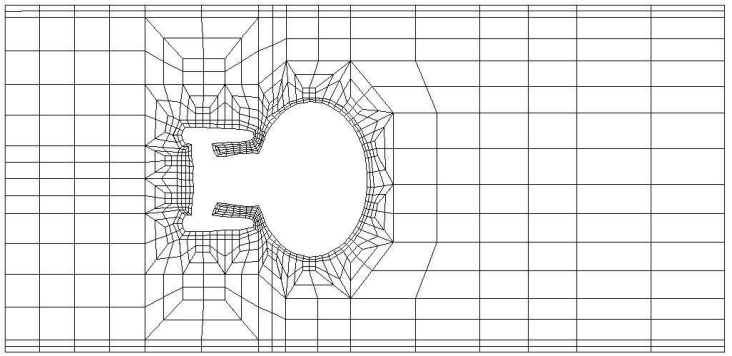
plus

Embedded into standard CG–method



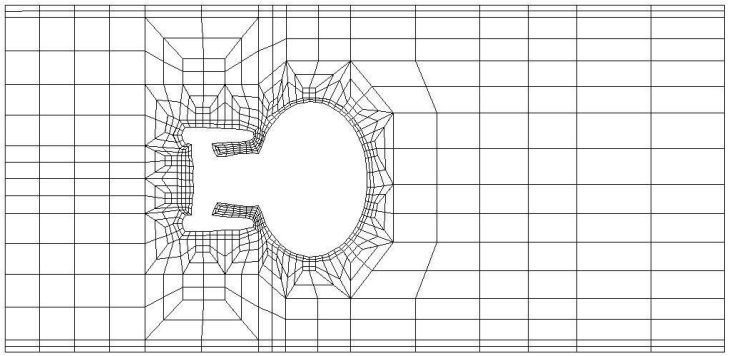
Example: Realization of ScaRC in FEAST

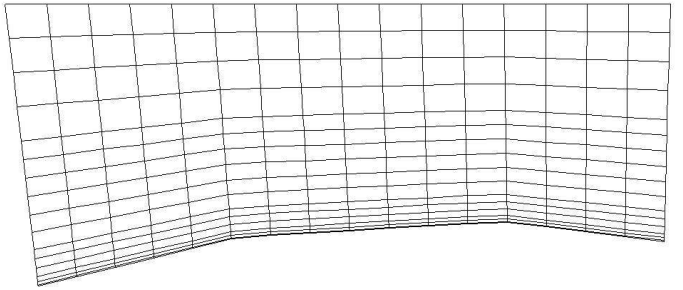
2D decomposition and zoomed (macro) element (LEVEL 3) with locally anisotropic refinement towards the wall



Example: Realization of ScaRC in FEAST

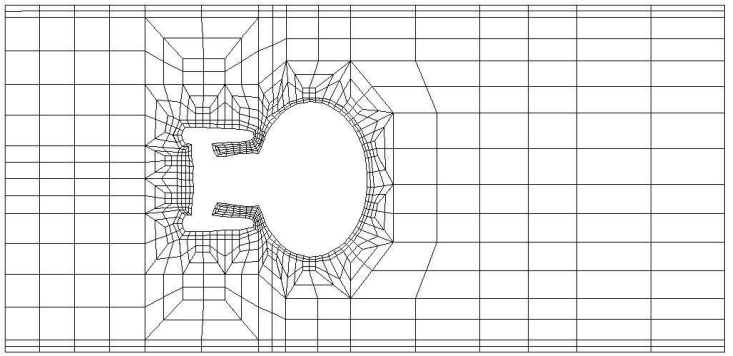
2D decomposition and zoomed (macro) element (LEVEL 3) with locally anisotropic refinement towards the wall

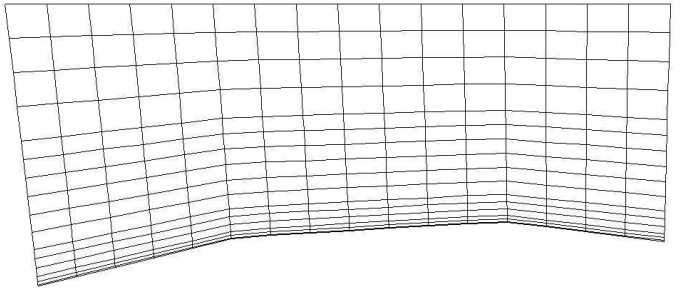




Example: Realization of ScaRC in FEAST

2D decomposition and zoomed (macro) element (LEVEL 3) with locally anisotropic refinement towards the wall

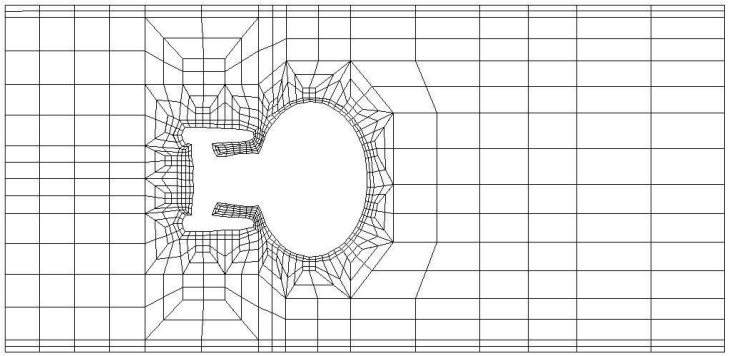


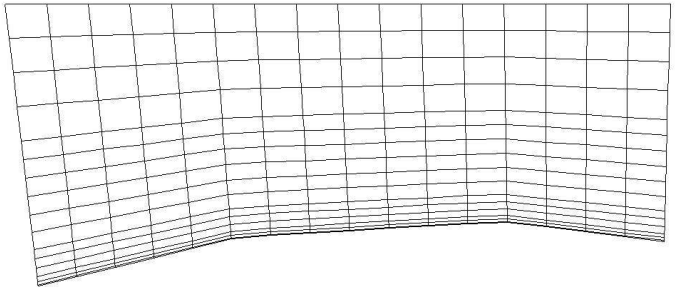


ScaRC-CG solver (smoothing steps: 1 global ScaRC ; 1 local ‘MG-TriGS’) for locally (an)isotropic refinement

Example: Realization of ScaRC in FEAST

2D decomposition and zoomed (macro) element (LEVEL 3) with locally anisotropic refinement towards the wall





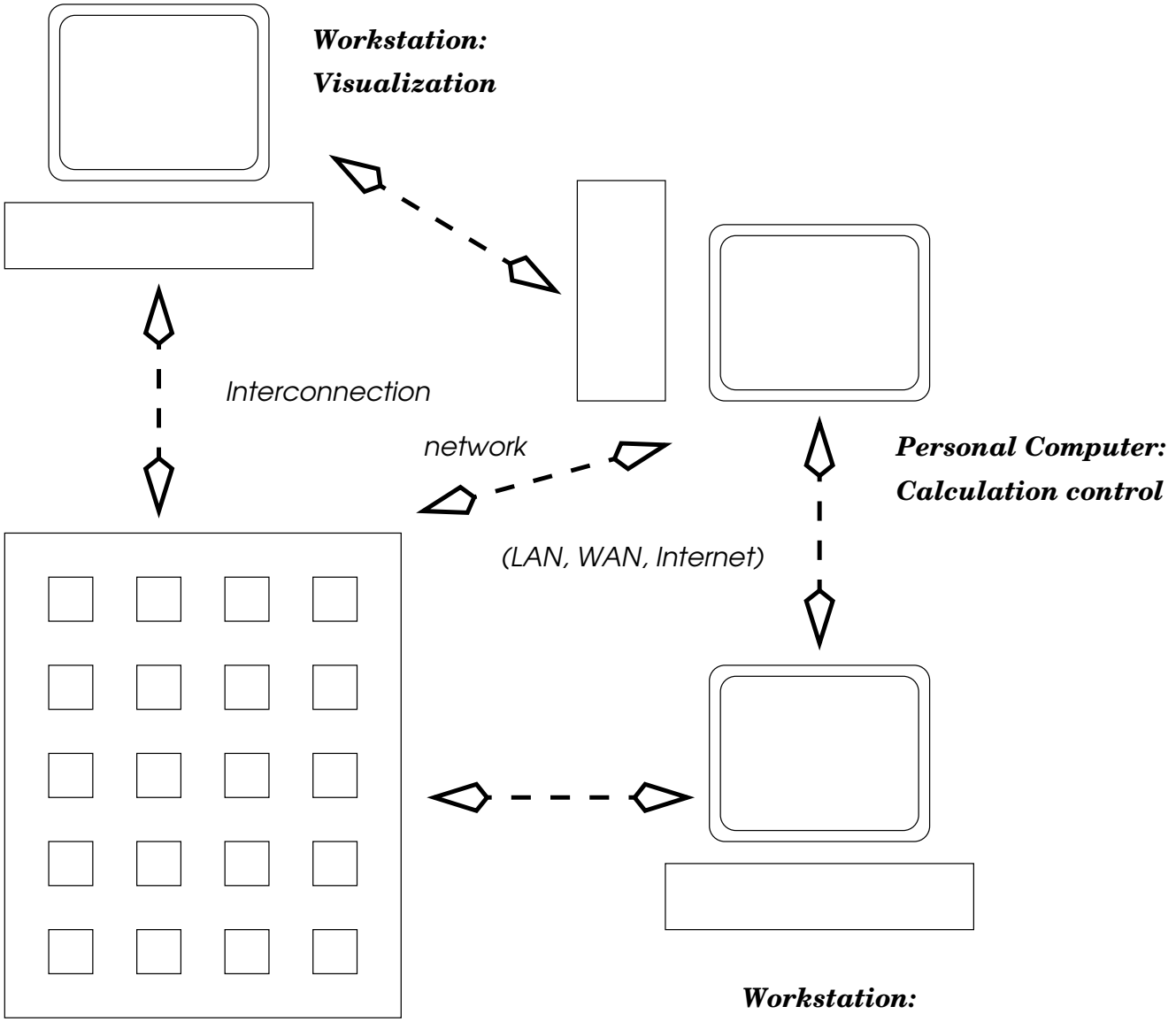
ScaRC-CG solver (smoothing steps: 1 global ScaRC ; 1 local 'MG-TriGS') for locally (an)isotropic refinement

Global (parallel) convergence rates

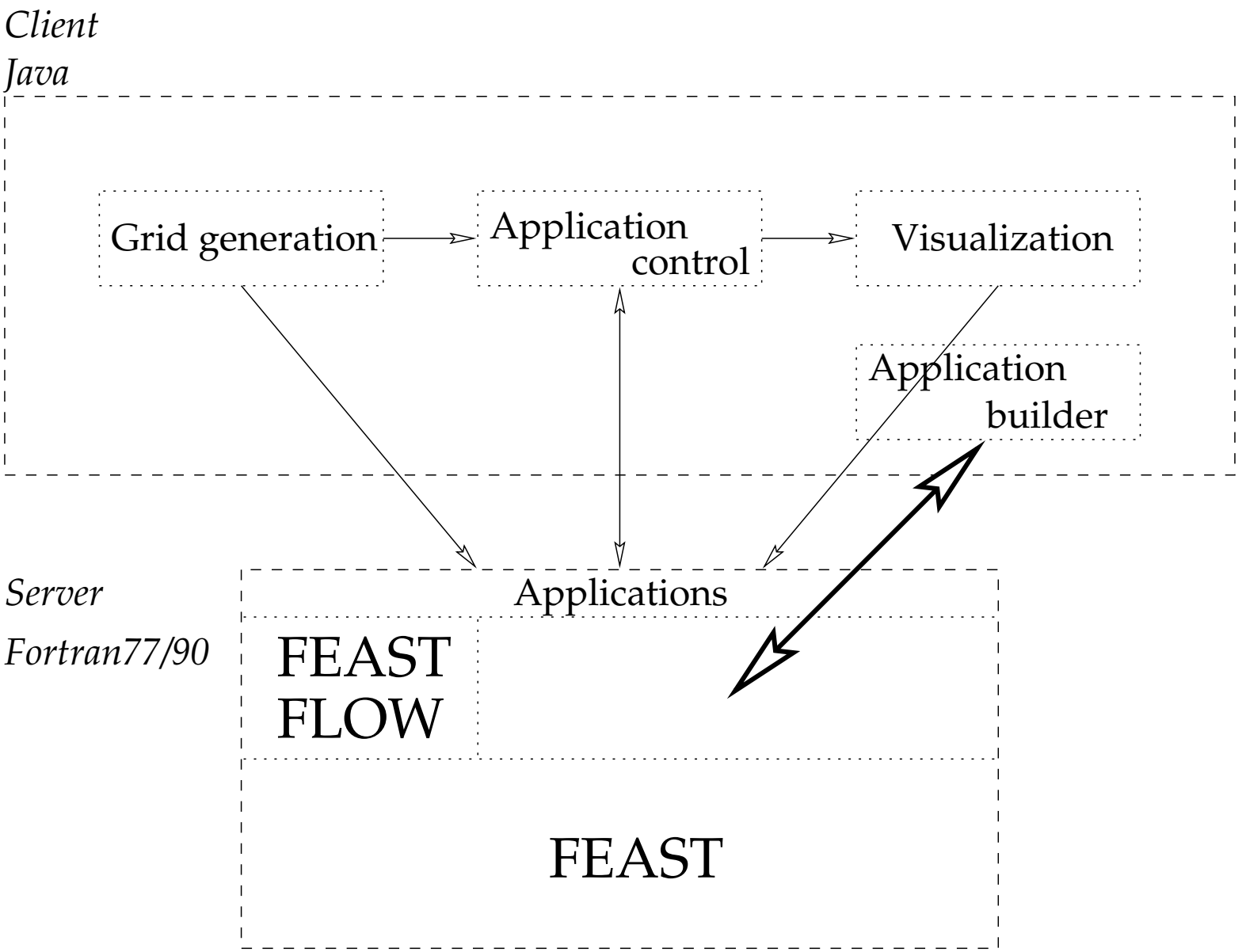
$\#NEQ$	Dirichlet 'Velocity'		Neumann 'Pressure'	
	$AR \approx 10$	$AR \approx 10^6$	$AR \approx 10$	$AR \approx 10^6$
$210,944$	0.17 (**8**)	0.18 (**8**)	0.21 (**9**)	0.15 (**8**)
$843,776$	0.17 (**8**)	0.17 (**8**)	0.20 (**9**)	0.17 (**8**)
$3,375,104$	0.18 (**9**)	0.19 (**9**)	0.22 (**10**)	0.22 (**10**)
$13,500,416$	0.19 (**9**)	0.18 (**9**)	0.23 (**10**)	0.23 (**10**)

FEAST structure

Overview


Workstation:
Visualization
Interconnection
network
(LAN, WAN, Internet)
Personal Computer:
Calculation control
Workstation:
Grid generation & editing
Parallel Computer:
Calculation

Overview


Client
Java
Grid generation
Application control
Visualization
Application builder
Server
Fortran77/90
Applications
FEAST
FLOW
FEAST

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views
- input of boundaries, nodes, elements per mouse or direct input

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views
- input of boundaries, nodes, elements per mouse or direct input
- all of the common edit functions like cut, copy, paste, scale, rotate

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views
- input of boundaries, nodes, elements per mouse or direct input
- all of the common edit functions like cut, copy, paste, scale, rotate
- integrity check, if the complete domain covered, if all of the grid parameter like aspect ratio, parametrization, node density fulfiled

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views
- input of boundaries, nodes, elements per mouse or direct input
- all of the common edit functions like cut, copy, paste, scale, rotate
- integrity check, if the complete domain covered, if all of the grid parameter like aspect ratio, parametrization, node density fulfiled
- interface for external grid generators

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views
- input of boundaries, nodes, elements per mouse or direct input
- all of the common edit functions like cut, copy, paste, scale, rotate
- integrity check, if the complete domain covered, if all of the grid parameter like aspect ratio, parametrization, node density fulfiled
- interface for external grid generators
- import functions for CAD formats like DXF

DeViSoRGrid 2D/3D

(**De**sign & **Vi**sualization **So**ftware **R**esource)

- edit in several windows with different views
- input of boundaries, nodes, elements per mouse or direct input
- all of the common edit functions like cut, copy, paste, scale, rotate
- integrity check, if the complete domain covered, if all of the grid parameter like aspect ratio, parametrization, node density fulfiled
- interface for external grid generators
- import functions for CAD formats like DXF
- boundary parametrization for 3D

DeViSoRVision 2D/3D

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots
  - particel tracing

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots
  - particel tracing
  - shaded plots for 2D

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots
  - particel tracing
  - shaded plots for 2D
  - grid distortion for CSD applications

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots
  - particel tracing
  - shaded plots for 2D
  - grid distortion for CSD applications
- builtin output formats
  - PostScript for import in publications

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots
  - particel tracing
  - shaded plots for 2D
  - grid distortion for CSD applications
- builtin output formats
  - PostScript for import in publications
  - automatic MPEG generation

DeViSoRVision 2D/3D

- modular system, definition of basic classes for default visualizations to reduce the effort adding new functionality
- built in visualizations
  - isolines, isosurfaces, isovolumes
  - cutlines, cutsurfaces
  - vector plots
  - particel tracing
  - shaded plots for 2D
  - grid distortion for CSD applications
- builtin output formats
  - PostScript for import in publications
  - automatic MPEG generation
  - CAD formats for further processing in CAD tools

DeViSoRControl I

DeViSoRControl I

- control of preprocessing, simulation and postprocessing, call of the components DeViSoRVision for visualization and DeViSoRGrid for preprocessing

DeViSoRControl I

- control of preprocessing, simulation and postprocessing, call of the components DeViSoRVision for visualization and DeViSoRGrid for preprocessing
- possibilty to control a machine remotely

DeViSoRControl I

- control of preprocessing, simulation and postprocessing, call of the components DeViSoRVision for visualization and DeViSoRGrid for preprocessing
- possibilty to control a machine remotely
- adaption to existing solver FeatFlow, but also to further solvers with a universal description format with respect to menu structure and help system

DeViSoRControl II

universal protocol for communication with the numerical kernel

DeViSoRControl II

universal protocol for communication with the numerical kernel

- communication of the components among each other, e.g. DeViSoRVision imports a CAD geometry und sends it to grid for further processing

DeViSoRControl II

universal protocol for communication with the numerical kernel

- communication of the components among each other, e.g. DeViSoRVision imports a CAD geometry und sends it to grid for further processing
- forwarding and changing of parameter values

DeViSoRControl II

universal protocol for communication with the numerical kernel

- communication of the components among each other, e.g. DeViSoRVision imports a CAD geometry und sends it to grid for further processing
- forwarding and changing of parameter values
- forwarding of grid geometries

DeViSoRControl II

universal protocol for communication with the numerical kernel

- communication of the components among each other, e.g. DeViSoRVision imports a CAD geometry und sends it to grid for further processing
- forwarding and changing of parameter values
- forwarding of grid geometries
- forwarding of solution vectors and values

DeViSoRControl II

universal protocol for communication with the numerical kernel

- communication of the components among each other, e.g. DeViSoRVision imports a CAD geometry und sends it to grid for further processing
- forwarding and changing of parameter values
- forwarding of grid geometries
- forwarding of solution vectors and values
- simulation control like tape recorder, forward, play, rewind, e.g. calculate 2 time steps, solution not satisfying, 3 time steps back, increase number of smoothing steps, continue calculation