FEAST

(Finite Element Analysis & Solution Tools)

M. Altieri, Ch. Becker, S. Kilian, A. Runge, S. Turek

Institut für Angewandte Mathematik,
Universität Heidelberg

email: {featflow}@gaia.iwr.uni-heidelberg.de
WWW: http://www.iwr.uni-heidelberg.de/featflow

Verbesserungsansätze für die numerische Lösung von partiellen DGL

- Verbesserung und Kontrolle der **Diskretisierung** $\Longrightarrow$ weniger Unbekannte
- Effiziente und robuste **Lösungsverfahren** $\Longrightarrow$ verbesserte Konvergenzraten
- Problem– und Hardware–angepaßte **Implementierung**

Mathematik vs Software Engineering ???

- Alle Komponenten sind gleichbedeutend!
- Alle Komponenten müssen aufeinander abgestimmt sein!
- Alle Komponenten haben theoretische und praktische Aspekte!

Beobachtung

- **Peak Performance** steigt mit jeder neuen Prozessorgeneration
- in 10 Jahren ein einzelner PC schneller als komplette CrayT3E heute

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**

- Speichergröße steigt ebenso, aber **nicht** Zugriffszeit

$\Longrightarrow$ Kluft zwischen CPU-Zeit und Speicherzugriff wird größer!

Fragen:

- Welche Rechenleistung erzielen heutige Codes?
- Sind spezielle Strategien notwendig, um “High Performance“ zu erreichen?

Performance Tests I

MFLOP–Raten der Matrix/Vektor–Multiplikation ($\frac{20N}{T}$) in FEAT-FLOW mit verschiedenen Knotennumerierungen (BMBF–Projekt "Flow around a car"):

Computer	N	**TwoLevel**	**Koord.**	**CM**	**stoch.**
	3484	25	22	22	**21**
SUN U450	13688	20	23	22	**19**
(250 Mhz)	54256	15	17	17	**13**
	216032	14	16	16	**6**
	862144	15	15	16	**4**

Resultate:

- identische Konvergenzraten (Jacobi–Glätter)
- identischer Berechnungsaufwand (gleiche Anzahl an arith.Op.)
- aber sehr **unterschiedliche** Laufzeiten
- weit von "Peak Performance" entfernt
- problemgrößenabhängig

Performance Tests II

In Mehrgitter/Krylov–Verfahren Matrix/Vektor–Operationen besonders CPU–intensitiv

Test verschiedener Matrix/Vektor–Multiplikationstechniken:

- **sparse (FEAT)**: Nichtnull–Einträge werden mit Position gespeichert, reguläre Verfeinerung (TL)
- **sparse (Adaptiv)**: wie oben aber mit "stochastischer" Numerierung (ST)
- **blocked band (FD-V).**: Speicherung als Diagonale, blockweise Anwendung ("BLAS 2+")
- **blocked band (FD-A).**: wie oben, aber mit konstanten Diagonalen

Test verschiedener Glätter (**TRIDI,GS, JAC**)

Test eines kompletten Mehrgitter–Zyklus (**MG**) (geschätzt)

Performance Tests II (Fort.)

MFLOP-Raten verschiedener Operationen in Abhängigkeit von der Problemgröße:

Computer	N	DAXPY	MV-V	MV-A	TRI-V	TRI-A
IBM RS6000	4K	308	166	475	169	323
(166 Mhz)	16K	97	167	326	142	230
'SP2'	512K	96	166	364	141	241
	1M	97	158	353	138	237
DEC Alpha	4K	295	104	384	87	248
(433 Mhz)	16K	56	98	295	73	159
'CRAY T3E'	512K	51	63	275	51	150
	1M	52	50	256	44	136
SUN U450	4K	128	211	435	146	130
(250 Mhz)	16K	132	64	419	61	129
	512K	29	41	172	37	71
	1M	29	38	149	35	68
INTEL PII	4K	48	59	112	51	70
(266 Mhz)	16K	27	36	93	32	60
'Home PC'	512K	18	33	69	26	43
	1M	17	33	67	26	42

Performance Tests II (Fort.)

Geschätze MFLOP–Raten eines kompletten MG–Zyklus über alle Problemgrößen:

Computer	Sch	MGTRIGS	MGTRI	MV	DAXPY	Total
IBM RS6000	Adaptiv					17.15
(166 Mhz)	FEAT					54.07
'SP2'	FD-V	141.75	142.99	159.81	98.91	141.99
	FD-A	229.4	221.94	353.15	98.91	234.29
	Index					141.02
DEC Alpha	Adaptiv					10.52
(433 Mhz)	FEAT					27.66
'CRAY T3E'	FD-V	57.57	59.37	58.82	54.53	58.8
	FD-A	133.69	140.8	264.71	54.53	147.14
	Index					69.39
SUN U450	Adaptiv					13.48
(250 Mhz)	FEAT					17.38
	FD-V	39.89	46.52	43.64	38	42.46
	FD-A	89.69	86.88	177.52	38	98.49
	Index					48.45
INTEL PII	Adaptiv					11.4
(266 Mhz)	FEAT					13.41
'Home PC'	FD-V	25.82	28	32.34	17.81	27.03
	FD-A	43.86	43.05	68.93	17.81	45.22
	Index					27.84

Performance Tests III
Divisionsfreie Num. LA

MFLOP-Raten für Jacobi-Vorkonditionierer:

Computer	N	klassisch	Hilfsvektor	Skalierung
IBM RS6000	4K	31	244	420
(166 Mhz)	64K	31	73	103
'SP2'	1M	31	72	105

- Dividieren vs. Caching-in und Pipelining
- große Beschleunigung 'leicht' erreichbar

Computer	N	klassisch	geblockt	'1 Block'
IBM RS6000	4K	33	32	33
(166 Mhz)	64K	32	32	33
'SP2'	1M	32	32	33

MFLOP-Raten für Tridiagonal-Vorkonditionierer **mit Division**

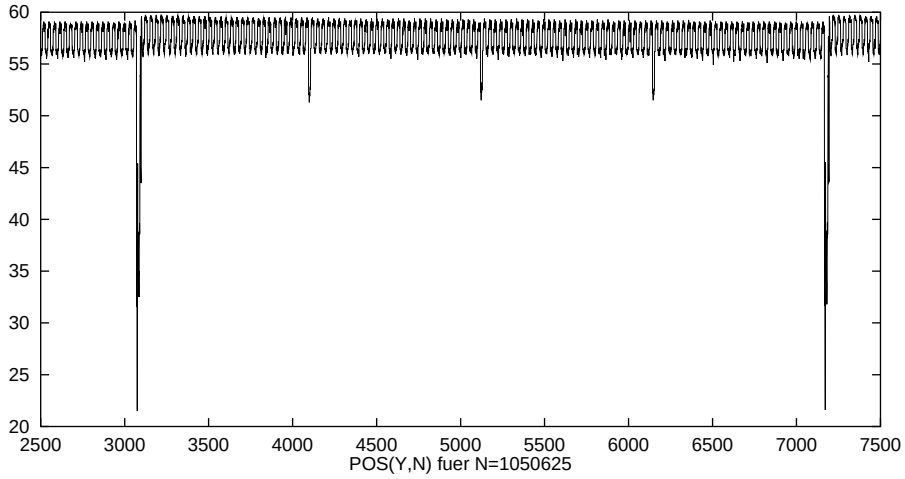
Computer	N	klassisch	geblockt	'1 Block'
IBM RS6000	4K	144	130	141
(166 Mhz)	64K	80	105	150
'SP2'	1M	80	106	150

MFLOP-Raten für Tridiagonal-Vorkonditionierer **ohne Division**

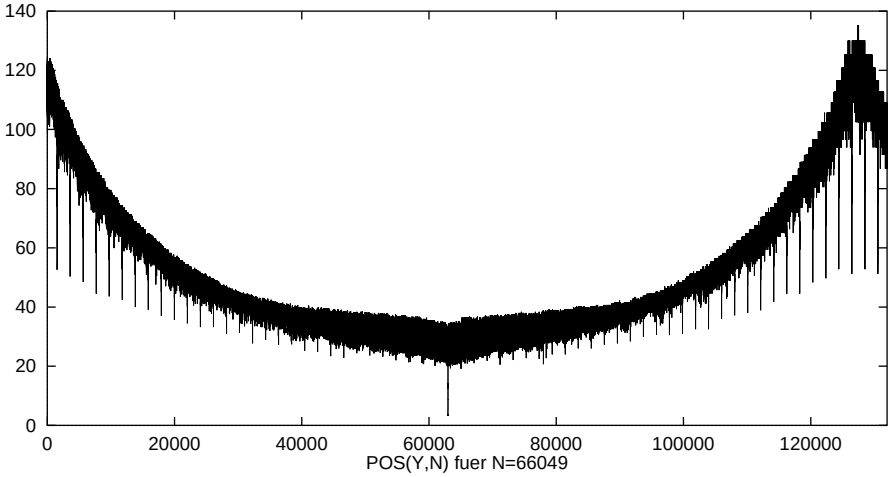
Performance Tests IV

Cache–Test für Vektor/Vektor–Addition (Variation des Speicherabstandes):

Cray T3E (20MF/s):



Sun U450 (3MF/s):



- Effizienz in hohem Maße abhängig von Speicherposition
- kleine “Fenster“ mit höherer Rechenleistung
- Kontrolle der Speicherposition?

Folgerungen

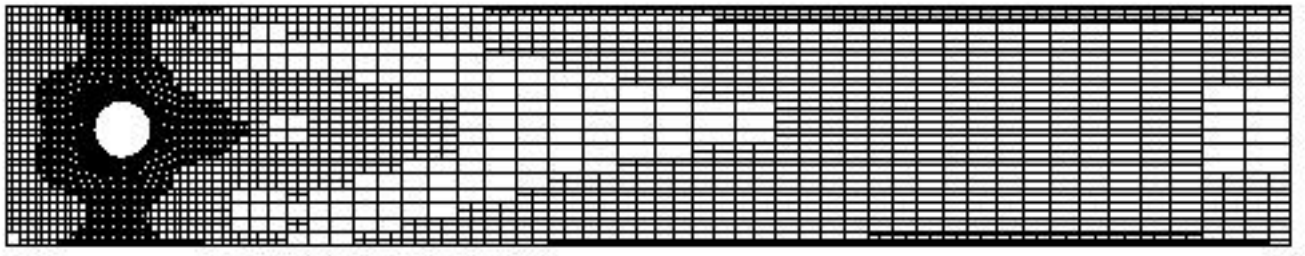
- Iterative Verfahren mit klassischer Implementierung oft sehr weit entfernt von “Peak Performance“
- Standard Sparse–MV–Operationen können nicht mit hoher Rechenleistung durchgeführt werden
- Unterschied zwischen PC und “number cruncher“ nicht so groß für gebräuchliche Software
- Prozessorfortschritt nicht so einfach nutzbar
- “banded block“ FD–Techniken oft überlegen (m.B.a. Effizienz und Problemgröße), aber wie einsetzbar in FEM Codes?

$$\Downarrow$$

moderne Numerik muß aktuelle Rechnerentwicklungen explizit berücksichtigen $\Longrightarrow$ **FEAST**

Hierarchisches “Divide and Conquer“

Optimales Gitter für den Drag–Koeffizienten via a posteriori Fehlerschätzung:

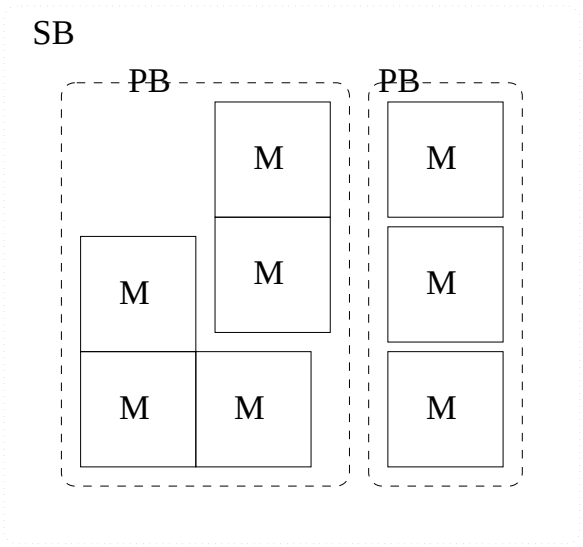


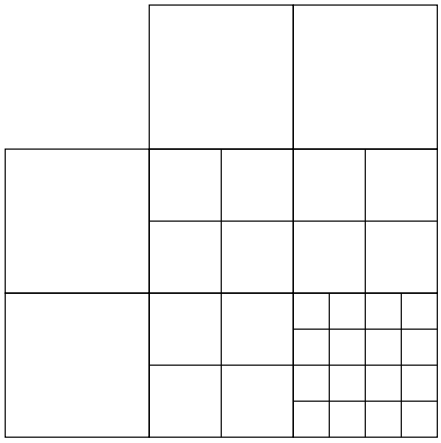
(Roland Becker, Inst.f.Ang.Math., Universität Heidelberg)

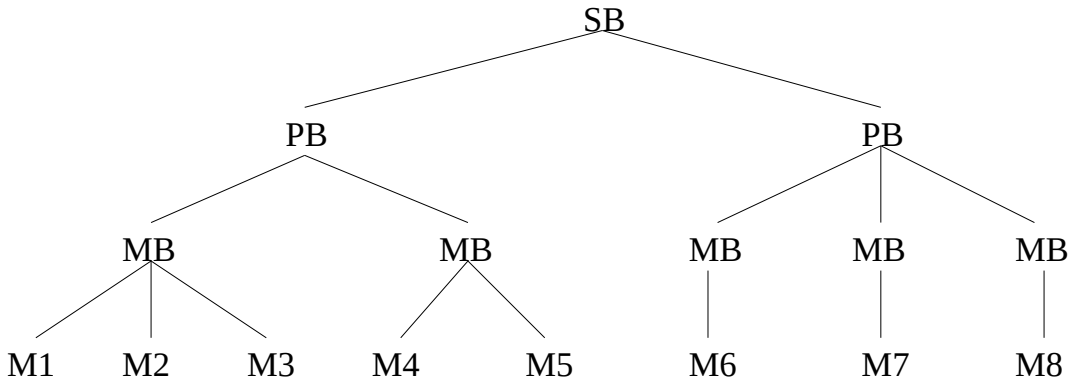
- adaptive Techniken nur lokal in Randnähe
- reguläre Strukturen (90 %) im Innern
- typisches Beispiel für CFD

- rekursive **Divide and Conquer** Strategie
- “Finde lokal struktierte Teilgebiete“ (Rechenleistung)
- “Finde lokale Anisotropien“ (lokal “verstecken“, Robustheit)

Hierachische Strukturen







Strukturen: “atomic units“, Matrixblocks, Parallelblocks, Gebiet

- reguläre Strukturen
- High Performance Linear Algebra möglich
- parallele Ausführung möglich

Frage: Passender Löser?

Löserstruktur

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

- generalisiertes Mehrgitter–/ Domain–Decomposition–Verfahren
- **rekursive** Kombination von **globaler** Defektkorrektur mit **lokalen** iterativen Verfahren/direkten Lösern
- Informationsaustausch über "globales" Mehrgitter
- Matrix–Vektor–Anwendung nur **lokal** auf "Matrixblocks"
- adaptiver Gebrauch verschiedener Glätter in Abhängigkeit von der "Härte" des Teilgebiets (Grad der Anisotropie)
- automatisches Expertensystem für Auswahl der lokalen Löser
- verschiedene Konzepte für Adaptivität (Makros)

Löserstruktur (Fort.)

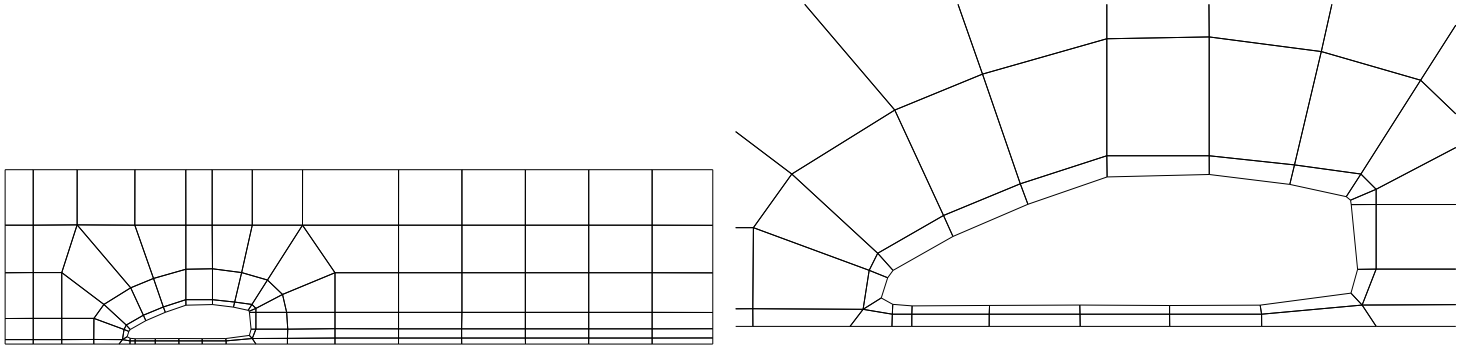
Resultate:

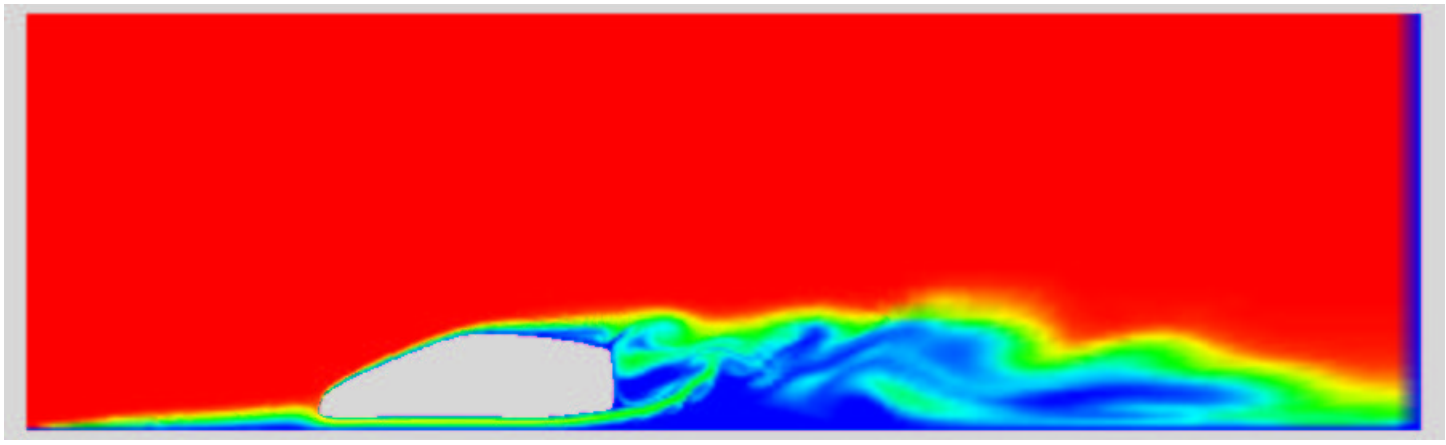
- gute Konvergenzraten
  - "unabhängig" von Gittergröße und Anzahl der Teilgebiete
  - kein Überlapp (Implementierung !!!)
- Anwendung unserer neuen "block banded"–Techniken

  $\Rightarrow$ "High Performance" möglich
- robuste Behandlung von Anisotropien und komplexen Details
- "automatische" (Blackbox) Parallelisierung
- Loadbalancing?

Numerisches Beispiel für ‘two-level ScaRC’

BMBF-Projekt: Ein schneller Navier–Stokes Löser für die Fahrzeug–Aerodynamik (Daimler-Benz AG)





	N_l	$ScaRC_{glo}$	
$h_n = 5 \cdot 10^{-4}$	8	**6**	0.093
	16	**6**	0.096
$AR \sim 10$	32	**7**	0.116
$h_n = 1 \cdot 10^{-5}$	8	**6**	0.092
	16	**6**	0.090
$AR \sim 320$	32	**6**	0.080
$h_n = 5 \cdot 10^{-7}$	8	**5**	0.063
	16	**4**	0.031
$AR \sim 10^4$	32	**4**	0.021

	N_l	GS_{loc}		TRI_{loc}	
$AR \sim 1$	64	**4**	0.03	**6**	0.09
	128	**4**	0.03	**6**	0.09
$AR \sim 10^2$	64	**11**	0.27	**7**	0.13
	128	**19**	0.47	**7**	0.13
$AR \sim 10^4$	64	**46**	0.74	**11**	0.25
	128	**95**	0.86	**11**	0.25
$AR \sim 10^1$	64	**75**	0.83	**5**	0.04
	128	**79**	0.83	**5**	0.04
$AR \sim 10^3$	64	**100**	0.87	**5**	0.04
	128	**171**	0.92	**5**	0.05
$AR \sim 10^5$	64	**205**	0.93	**5**	0.04
	128	**447**	0.97	**5**	0.05

Globales und lokales Konvergenzverhalten von ScaRC in Abhängigkeit von lokalen Verfeinerungsstrategien

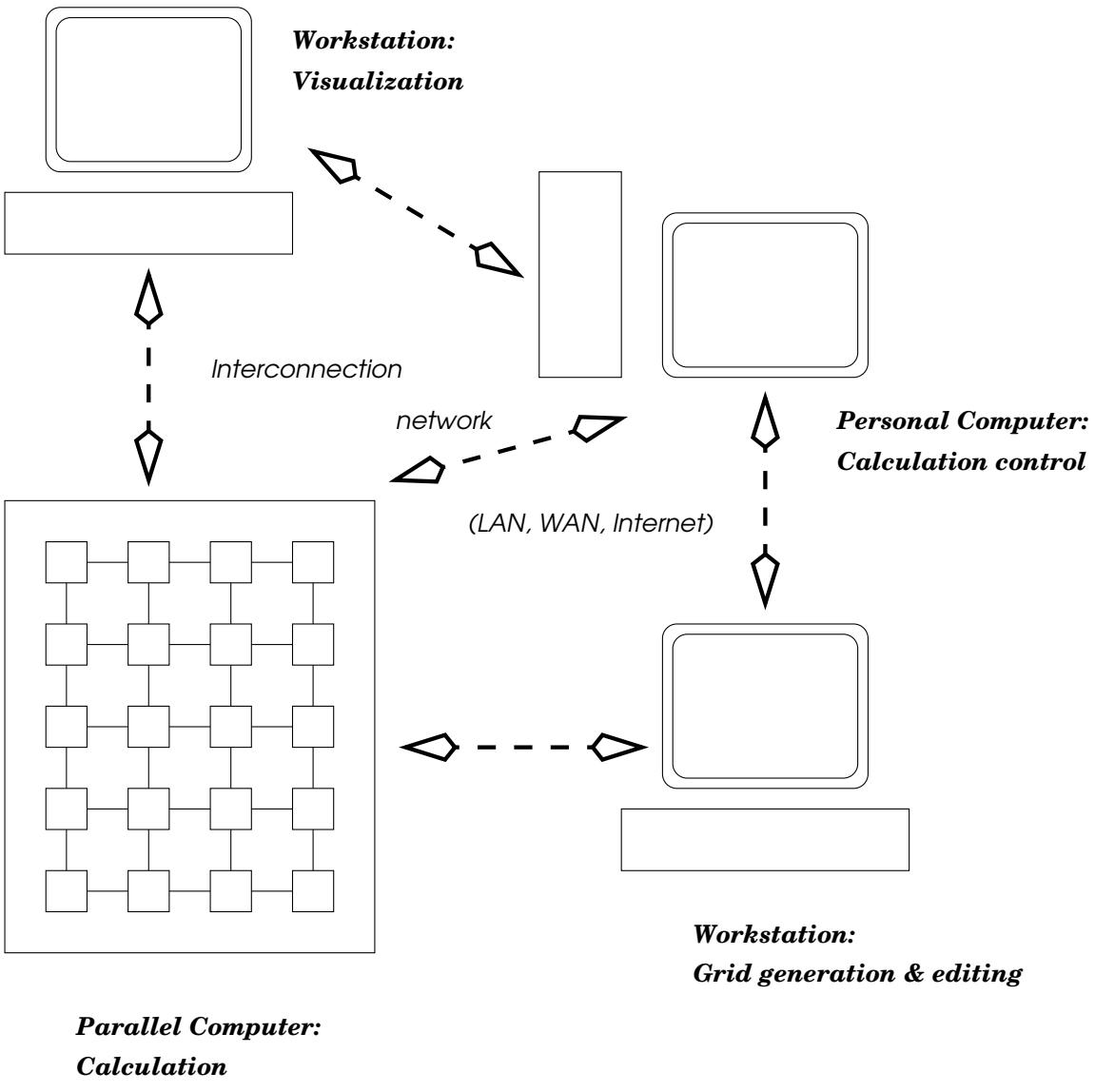
Bemerkung: MFLOP–Raten für lokales Mehrgitter/MV

DeViSoR

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

Ziel: Realisierung des Pre– und Postprocessings, einheitliche Oberfläche für FEM–Packages, offenes Konzept

Implementierung: Java



Preprocessing: DeViSoRGrid

- Erzeugen und Verwalten von Randbeschreibungen/Gitterstrukturen
- Automatische Generierung und Modifizierung von Grobgittern
- Realisierung von Importfunktionen für externe Programme

Processing: DeViSoRControl

- Steuerung der Applikation
- Steuerung des Preprocessings, Online–Änderung der Konfiguration
- Steuerung des Postprocessings, Online–Visualisierung

Postprocessing: DeViSoRVision

- Gittervisualisierung
- Ergebnisvisualisierung mittels Techniken wie Shading, Isolinien, Partikel–Tracing
- Animationsmöglichkeiten für nichtstationäre Probleme