



ISC2004 Heidelberg

Tutorial, June 22, 2004

Integrated Performance Analysis of Distributed Computer Systems

Franz-Josef Pfreundt, Fraunhofer ITWM
Kaiserslautern, Germany
Matthias Merz, University of Mannheim, Germany



Dirk Merten

Heinz Kredel
Martin Meuer
Hans Plum
Thomas Waschk



Karl Solchenbach



Matthias Merz



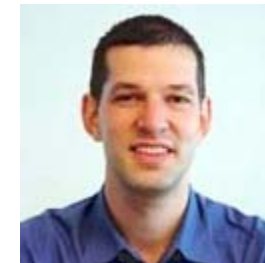
Dimiter Stoyanov



Franz-Josef Pfreundt



Peter Klein



Boris Briehl



1999 first Ideas

Cluster - SMP/ Vector
PC-Cluster enter TOP 500
Clusters good in Linpack
but bad as systems

Need for a System Benchmark

Berkeley 2000
Initiatives to establish new
Benchmarks



2002 IPACS and other projects started (PERC)

Aims:

- Development and distribution of scalable, portable and realistic benchmarks for today's and future parallel computers
- **Benchmark environment** supports the user in selecting, downloading, executing and displaying results of benchmarks
- **Low Level Benchmarks** to characterize computer architectures
- **Application Benchmarks** and **Benchmarks of commercial software** to characterize performance in real use
- **Performance modeling and prediction**



Fraunhofer Institut
Techno- und
Wirtschaftsmathematik



pallas

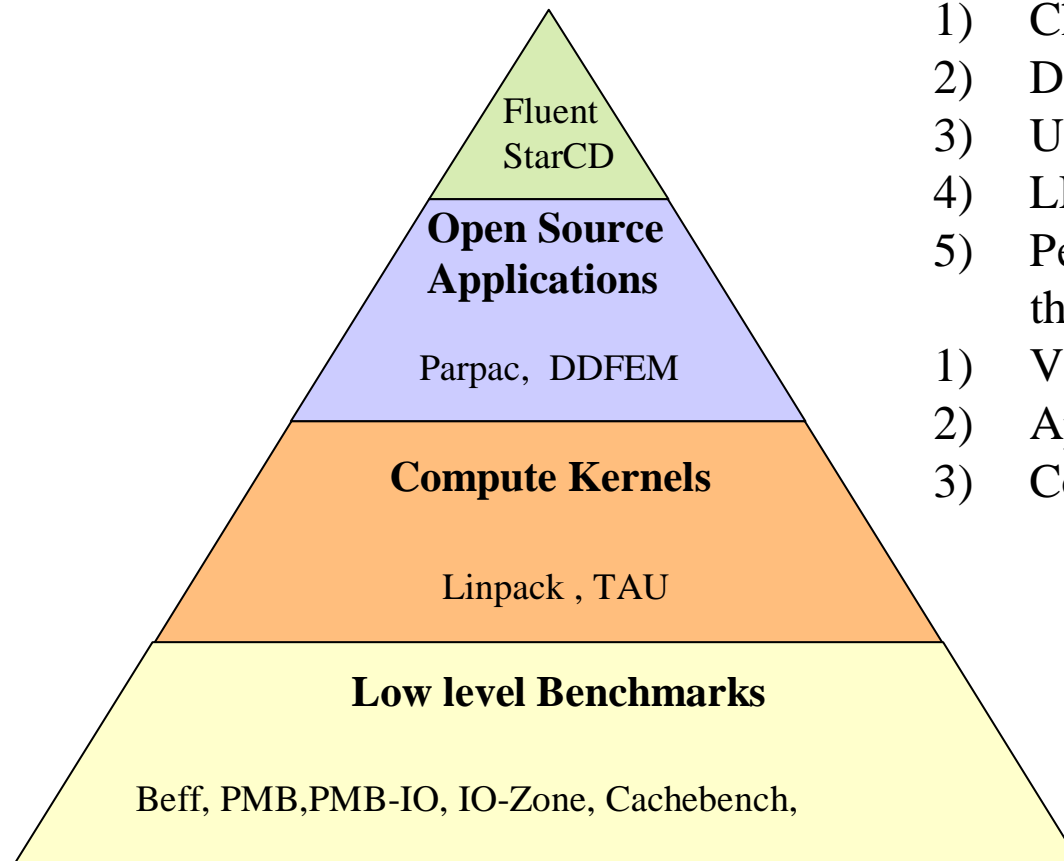
· · T · · Systems ·



NATIONAL ENERGY RESEARCH
SCIENTIFIC COMPUTING CENTER

Advancing Computational Science of Scale—
Producing Real Results





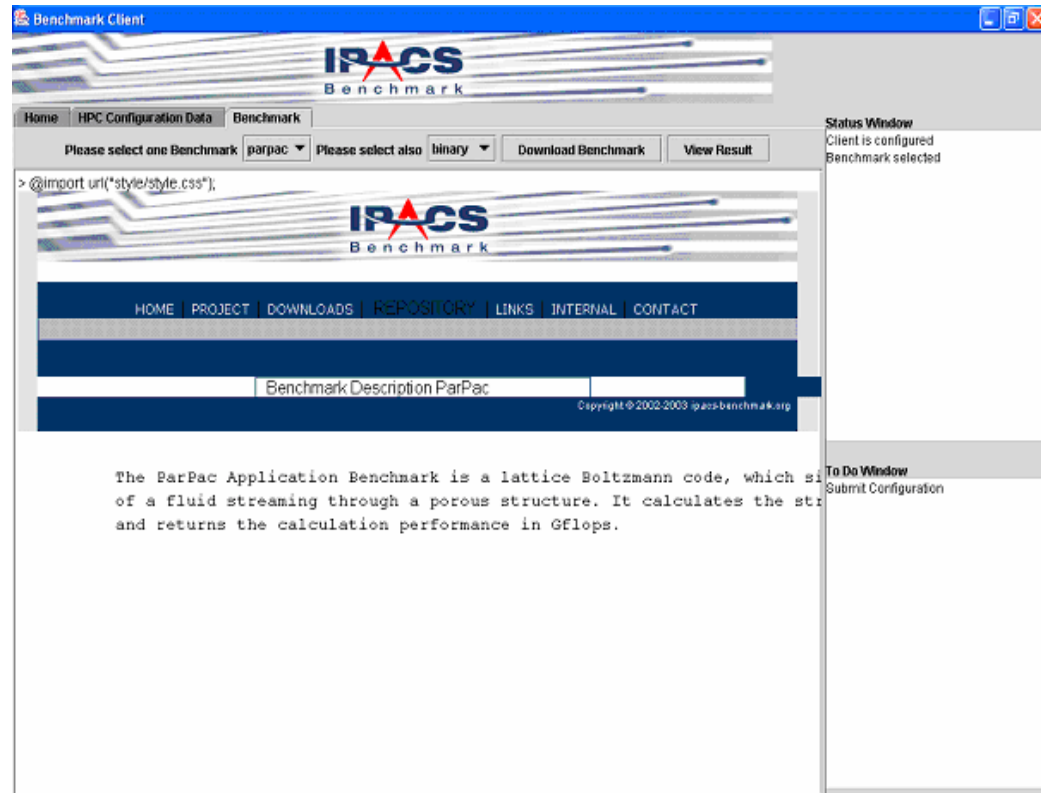
- 1) Characterize distributed systems
 - 2) Develop new benchmarks
 - 3) Understand own benchmarks
 - 4) LLB characterize the system
 - 5) Performance models characterize the applications
-
- 1) Verify Performance Models
 - 2) Apply performance models to
 - 3) Commercial applications

Benchmark Pyramid

- Providing benchmark binaries and sources
- Offering appropriate benchmarks for system configuration of HPC machines
- Storing and presenting benchmark results

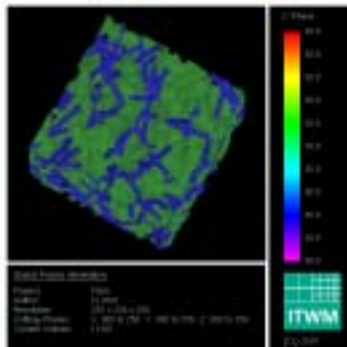


- Active support of the user
- Detection of system configuration
- Selecting and downloading benchmarks
- Configuring benchmarks for HPC machines
- Running benchmarks on HPC machines
- Submitting results to IPACS server

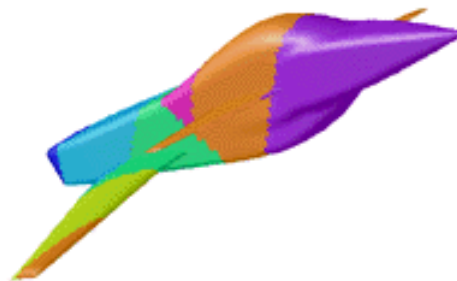


Overview Benchmarks

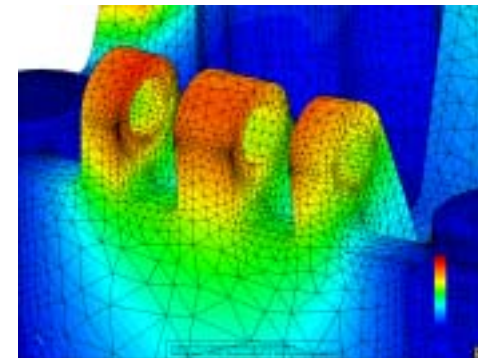
- Low –Level : Cachebench, Beff
- Kernels : TAU
- Application benchmarks : ParPac, DDFEM
- Commercial Codes : Fluent, StarCD



ParPac



Fluent



DDEM

- **CACHEBENCH**

<http://icl.cs.utk.edu/projects/lcbr>

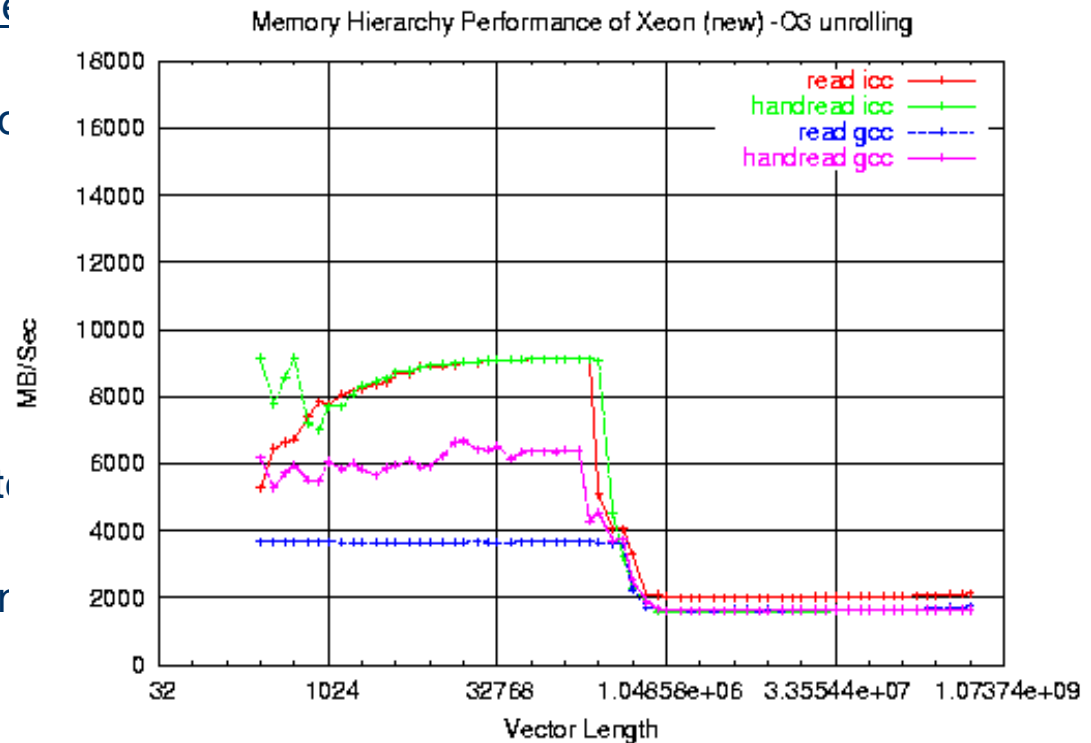
- read, write, modify
- plain and hand-tuned version
- effectiveness of compiler



- **IO Benchmark IOZONE**

<http://www.iozone.org>

- read / write / reread / rewrite
- vary file and record size
- Influence of caches / system



- **PMB**

<http://www.pallas.com/e/products/pmb>

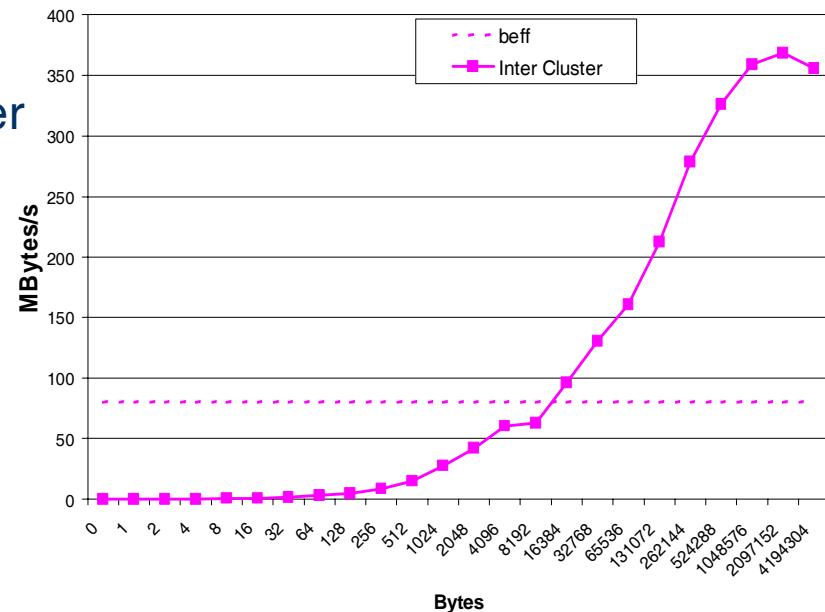
- variety of MPI communication patterns
- point-to-point and collective communication
- Beff: integration and average over length and patterns → single number

Average Bisection Bandwidth

- **PMB-IO**

- MPI IO characteristics
- serial and parallel IO

IPACS Message Passing Benchmark (IBM SP3)





Low-Level Benchmarks Beff

	Myrinet/Linux	Quadrix/Linux	Infiniband	IBM SP3
	LANL	LLNL	Summer 2003	NERSC
2	53	87	129	
4	53	87	128	
8	51	84	128	
16	48	82	131	1node 39
32	45	78	120	2nodes 30
64	42	76	84	
80				5 nodes 21
128	39	72		
160				10 nodes 18
256	34	64		
320				20 nodes 16
384	33	60		
512		58		32 nodes 16
768		48		

Measurements done with support from Linux Networx, NERSC

-
- Application benchmarks

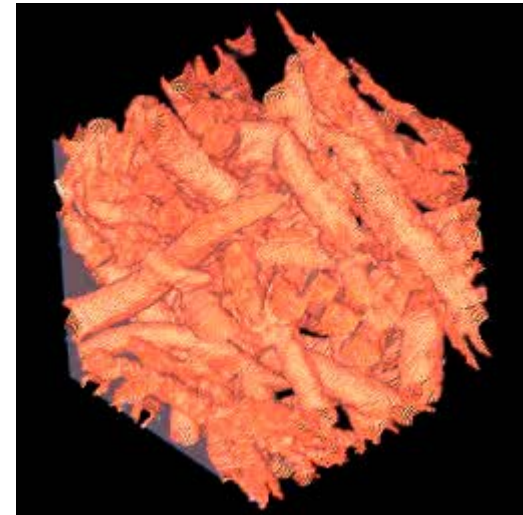


Fraunhofer Institut
Techno- und
Wirtschaftsmathematik

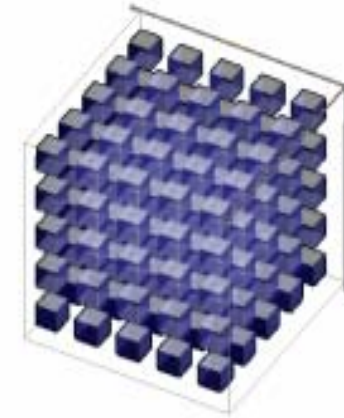
Thanks to NERSC (Berkeley)

ParPac : Parallel Particel Code

- solver for complex 3-dim flow problems
- developed at ITWM during the last 5 Years
- used in industry projects
- based on Lattice Boltzmann approach
- C++ Code, complex data structures
- suited for very complex geometries
- fully parallelized
- automated domain decomposition
- dynamical load balance
- optimized communication paths



Calculation permeability



- Flow through generated porous structure

- Scalability

- Benchmark :

- Grid point calculation rate
- Flops
- I/O-Performance (parallel, serial)

- Calculates Permeability

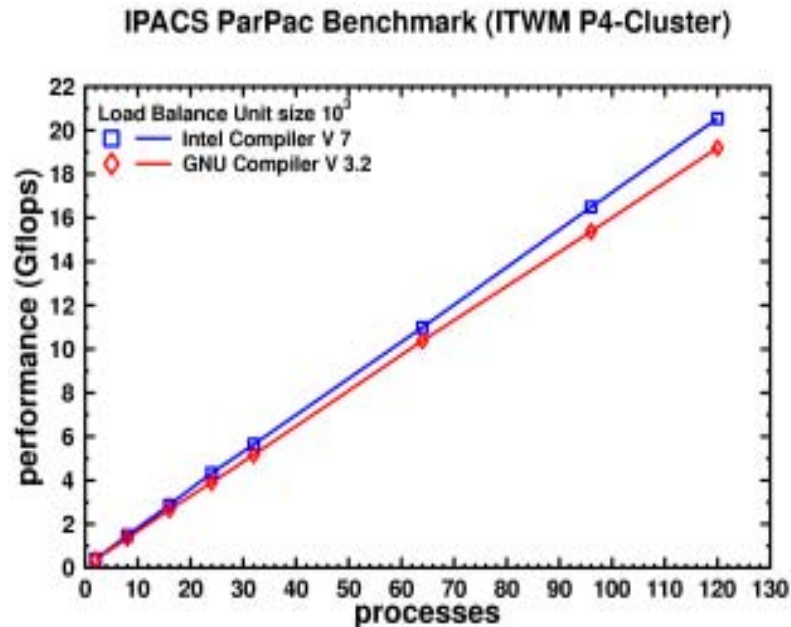
→ Comparison of results

Architecture	Compiler
Intel PIII	GNU Compiler V 2.95.3
Intel P4	GNU Compiler V 3.2
	Intel Compiler V 7
	Microsoft Visual C++ 6.0
IBM SP Power3	GNU Compiler V 3.2.1
	IBM VisualAge Compiler V 6
NEC SX6	C++/SX compiler (in progress)

ParPac-Benchmark results



ITWM Dual P4 2.4 GHz-Cluster (128 CPUs):

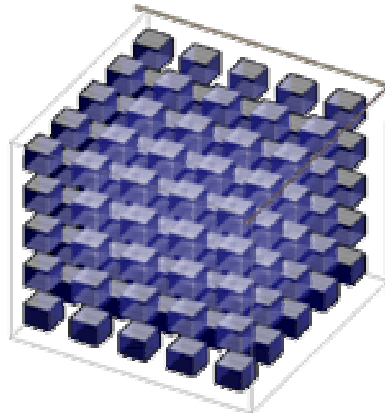


- 614.4 GFlops peak performance (4.8 GFlops/CPU)
- Myrinet

Compiler	Efficiency
GNU Compiler V 3.2	3.3% of Peak
Intel Compiler V 7	3.6% of Peak

- Excellent scaling
- Optimization by the Intel compiler leads to increase of 7%

Benchmark Example (LANL)



```

Konsole <8>
File Sessions Settings Help
=====
ParPac Application Benchmark

BGK Lattice Boltzmann simulation of a viscous fluid
streaming through a porous media

Fraunhofer ITWM - Germany
=====
Benchmark description:
-----
Simulation of the dynamical behaviour of a          * * * * *
viscous fluid streaming through a regular            ----->
porous media under the influence of a                * * * * *
driving pressure gradient and calculation            ----->
of the permeability of the porous media.             * * * * *

System information:
-----
Date_____ : Wed Dec 18 05:45:36 2002
Machine____ : i686
System_____ : Linux
Release_____ : 2.4.19-lanl.18lrx12
Version_____ : #3 SMP Wed Dec 11 15:49:00 MST 2002
Processors___ : 1536

Fluid parameters:
-----
mass density_____ : 1.260e+03 kg/m^3
kinematic viscosity_____ : 1.167e-03 m^2/s
pressure gradient_____ : 7.000e+06 Pa/m

Benchmark parameters:
-----
total no. of grid points_____ : 531441000
total no. of fluid points_____ : 468801736
material porosity_____ : 8.759e-01
size of each load balance unit_____ : 1000

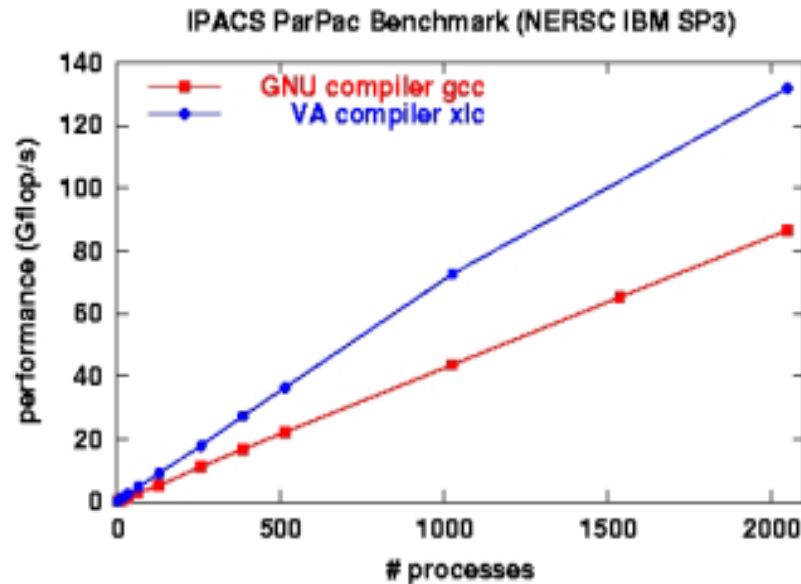
Benchmark results:
-----
calculated permeability of the material: 1.992e-07 m^2

total simulation time_____ : 13.74 s
fluid grid point calculation rate_____ : 338768320.85 1/s
floating point operation rate_____ : 2.022e+02 GFlops

total I/O time for writing dump file_____ : 0.00 s
I/O transfer rate to the disk_____ : 0.00 MBytes/s

End of benchmark
=====
New Konsole
  
```

NERSC Seaborg IBM SP RS/6000 (6080 Power 3+):



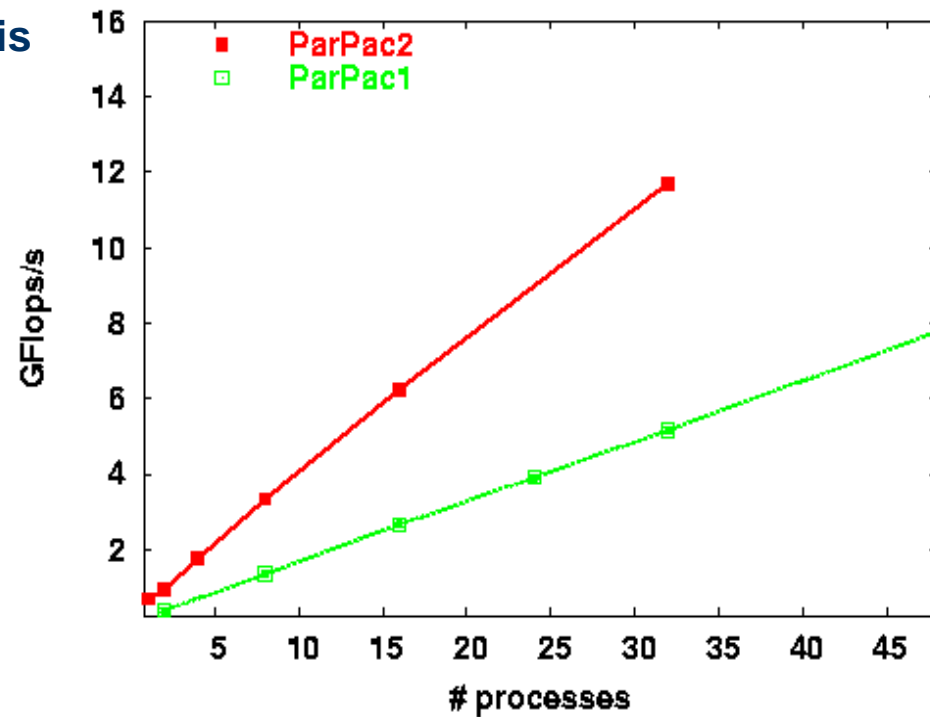
- ❑ 9.1 TFlops peak performance (1.5 GFlops/CPU)
- ❑ IBM SP High Speed Switches

Compiler	efficiency
GNU Compiler V 3.2	2.8% from Peak
IBM VA Compiler V 6	4.7% from Peak

- Good scaling up to 2048 CPUs
- Optimized compiler gives 60% better performance

- Redesigned and improved version is under construction
- optimized L2 Cache Reuse
- point data is grouped according to Cache size
- serial prototype reaches 20% peak performance

Dual P4 2.4 GHz, Myrinet (preliminary)

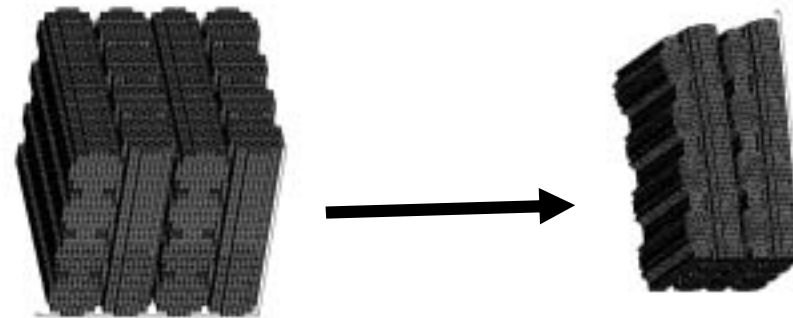


8% Peak Performance

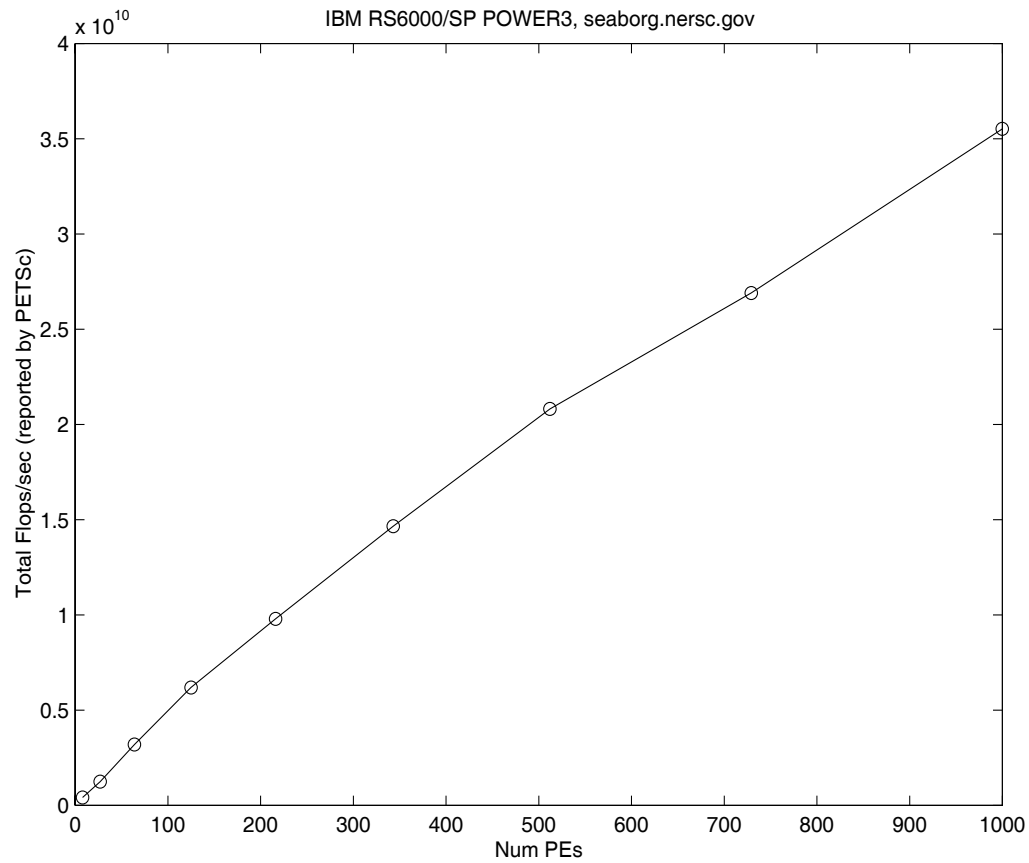
Domain Decomposition FEM

(currently linear elasticity)

- Developed from scratch (C++)
- Requirements :
 - parallel geometry generation
 - parallel Grid Generation
- scalability of benchmark example
- generate comparable results

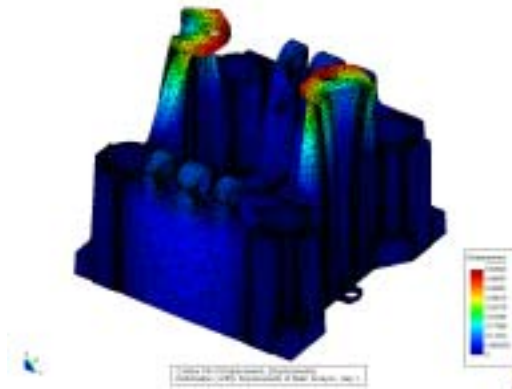
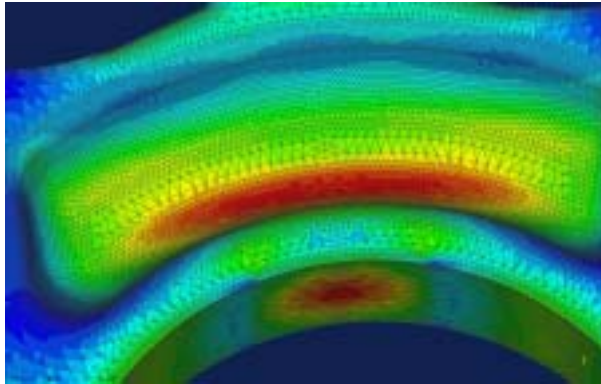


Fiber structure under compression

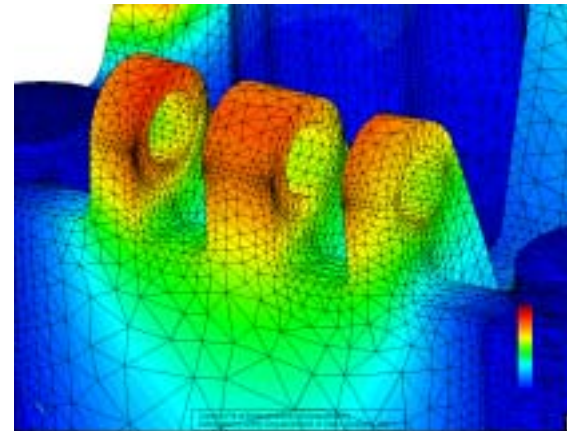
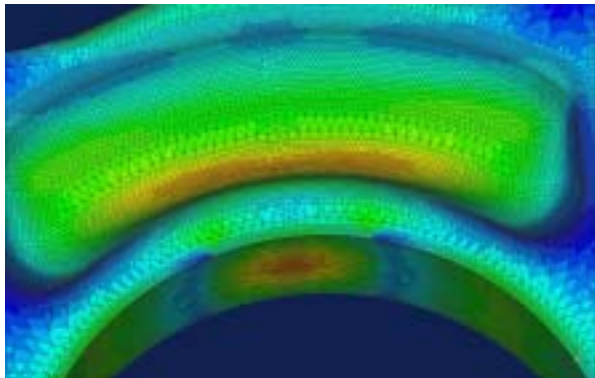


- IBM RS6000/SP POWER3
NERSC

- Peak : 1.5 TF (1000 CPU)
- App : 35 GF
- 2.3 %



Shape Optimization for castings



Speed up against commercial codes

Example:

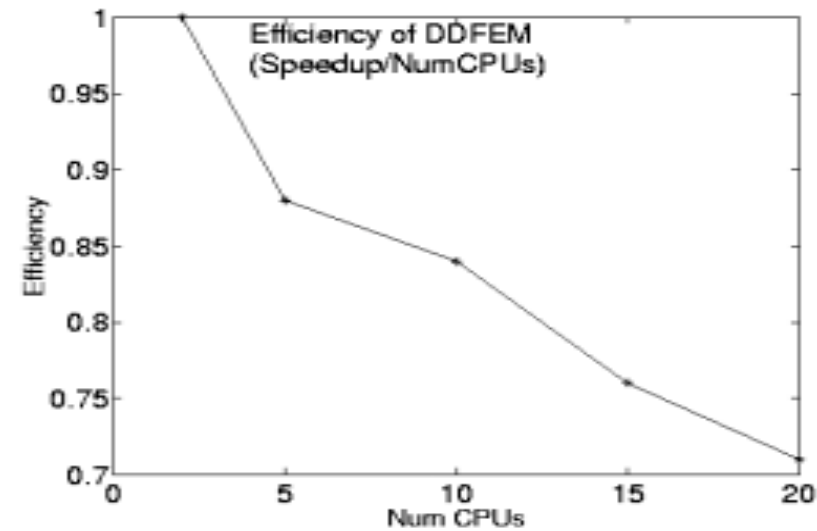
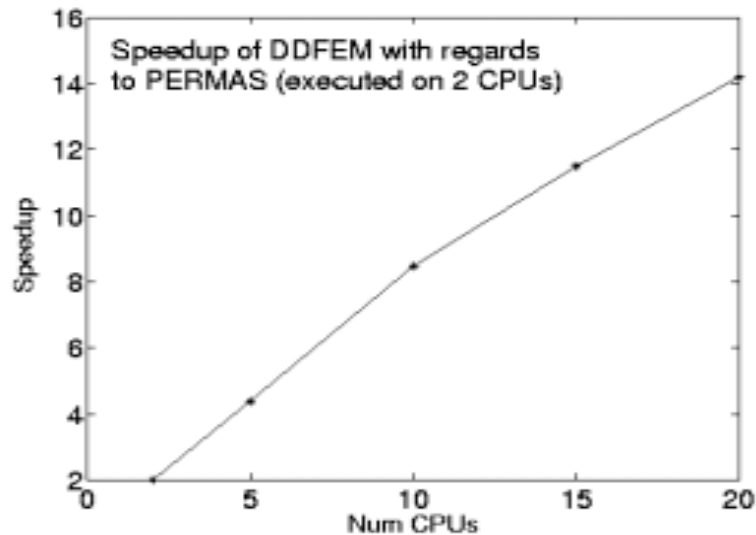
Linear Elasticity

234323 FE-Knoten, 702969 DOF

1199228 linear Tetraederelemente

New DDFEM version with Multigrid

„from hours to minutes



During a shape optimisation process you
Need to run the FEM code 20 times

Taubench.

CFD Benchmark for unstructured Grids

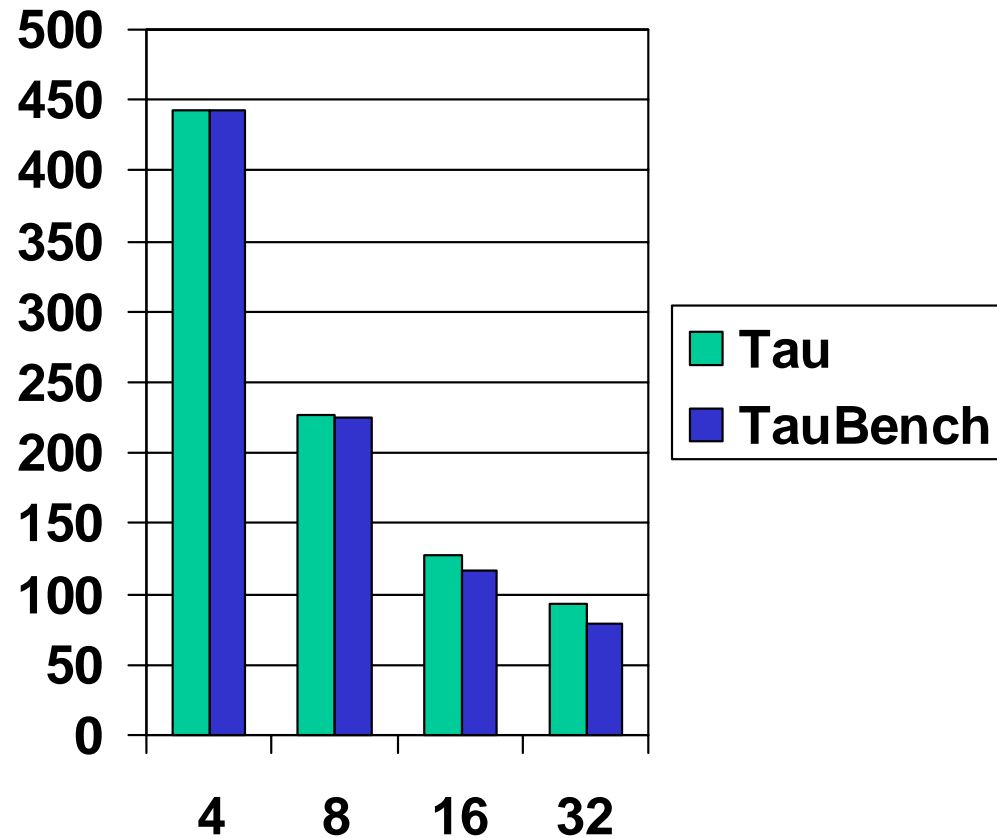
Christian Simmendinger

Alfred Geiger

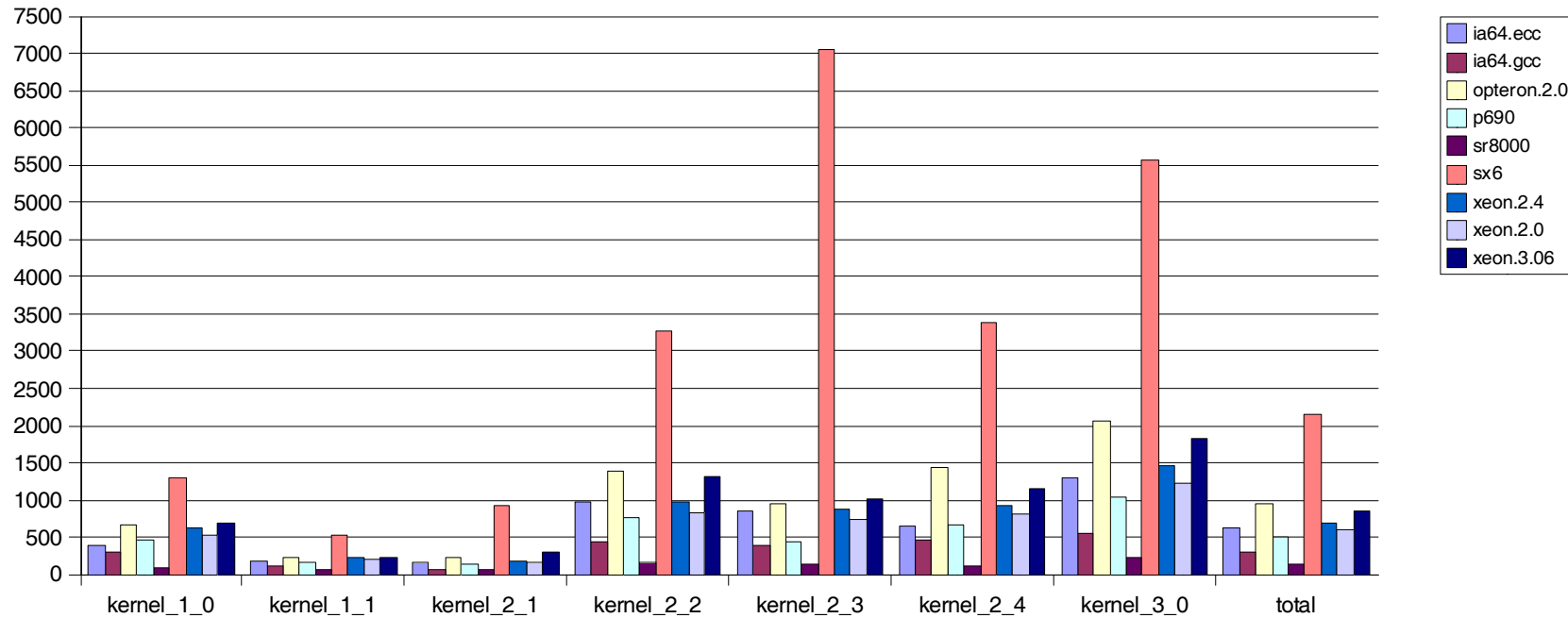
Peter Herchenbach

Scaling of Original and Benchmark Code

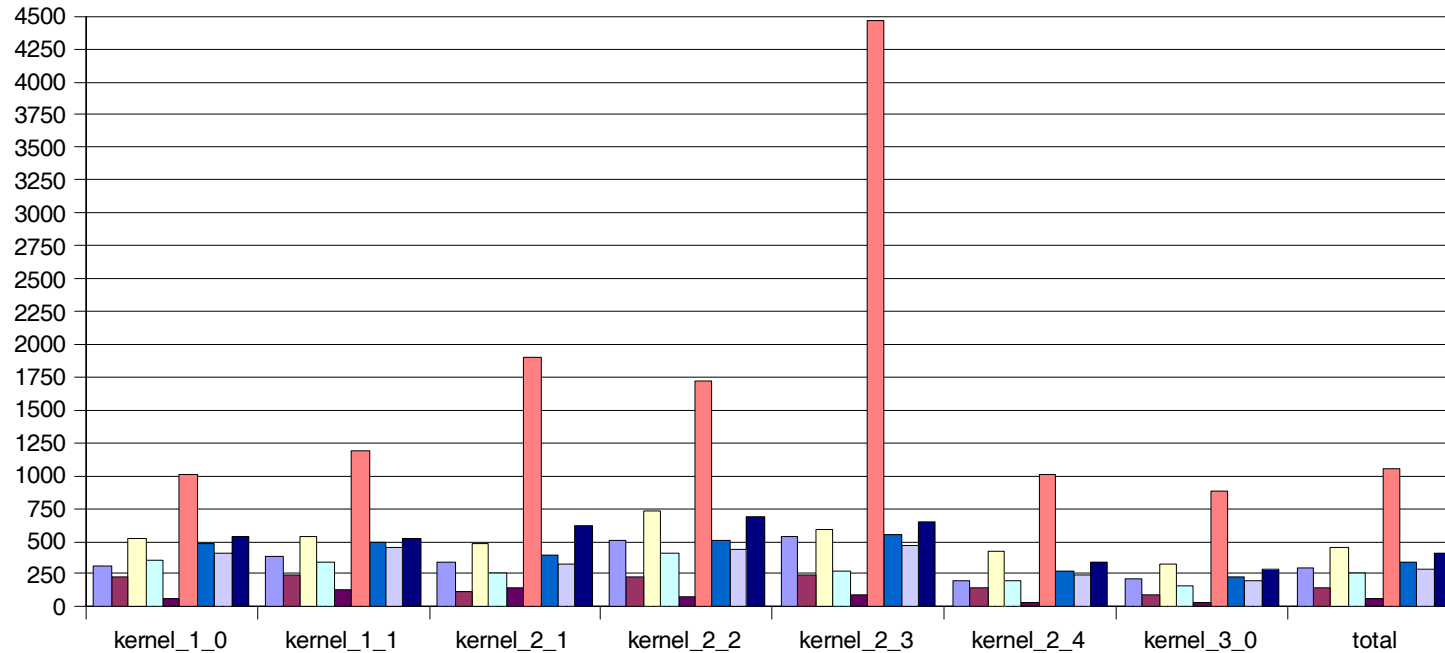
- Mem -> Mem/2 in Pseudo Code
- Neglects boundary overhead (both communication and computation)
- In direct comparison correction is needed ~ $(f1 \cdot mem)^{(2/3)} \cdot f2$



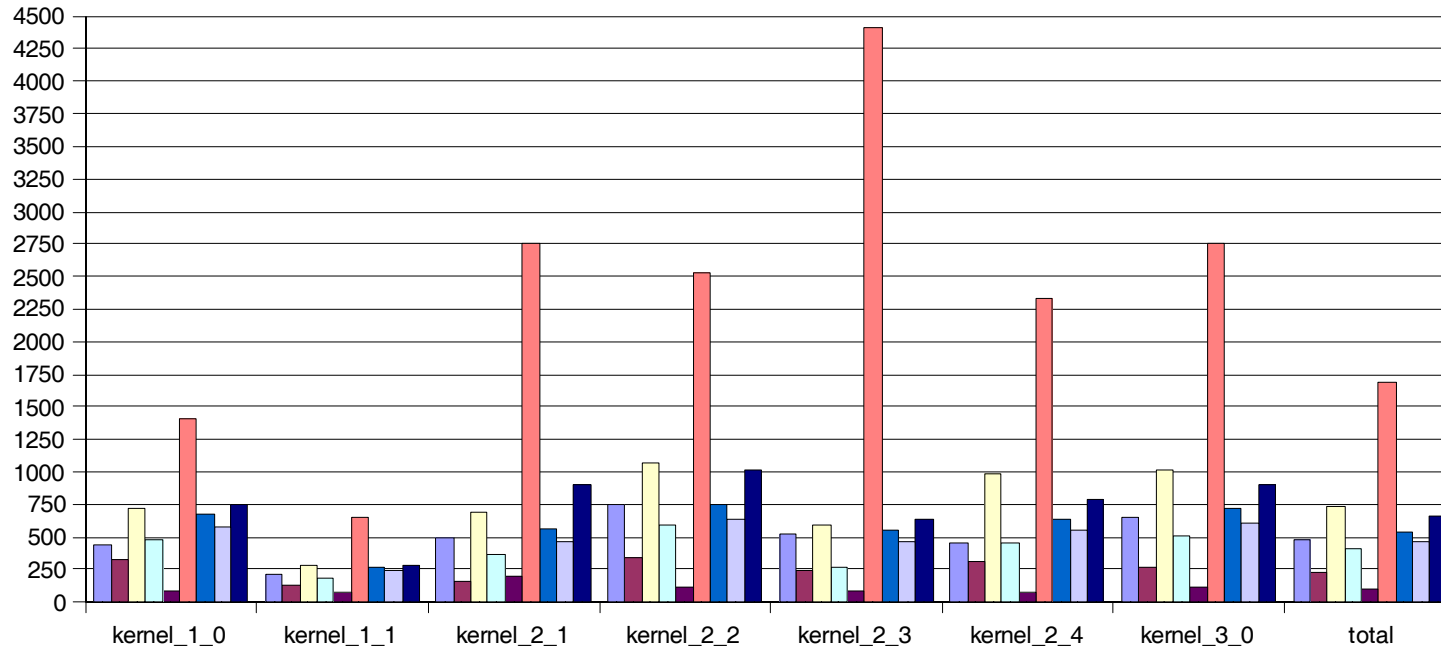
kernel_1_0	398,4	297,66	666,04	446,63	82	1287,84	622,61	529,14	688,04
kernel_1_1	170,22	105,88	236,1	152,43	56,78	526,82	221,14	198,74	229,21
kernel_2_1	167,78	53,49	233,93	125,43	67,21	923,32	190,78	158,47	302,31
kernel_2_2	959,05	438,23	1375,79	767,27	154,06	3263,19	973,12	828,74	1305,07
kernel_2_3	842,26	379,31	942,57	433,23	140,57	7054	871,88	742,02	1010,87
kernel_2_4	649,02	461,48	1425,44	666,1	107,93	3379,89	919,64	796,34	1149,55
kernel_3_0	1304,52	539,77	2062,4	1035,55	232,86	5565,28	1465,23	1225,6	1828,13
total	616,03	284,02	934,02	515,98	123,51	2159,84	685,43	588,03	845,25



kernel_1_0	310,7	232,14	519,43	348,31	63,95	1004,35	485,55	412,66	536,59
kernel_1_1	384,93	239,43	533,9	344,7	128,39	1191,33	500,06	449,42	518,31
kernel_2_1	345,7	110,21	482,01	258,44	138,49	1902,47	393,09	326,52	622,91
kernel_2_2	504,72	230,63	724,04	403,79	81,07	1717,31	512,12	436,14	686,82
kernel_2_3	533,9	240,44	597,48	274,62	89,1	4471,47	552,67	470,36	640,78
kernel_2_4	194	137,94	426,08	199,1	32,26	1010,29	274,89	238,04	343,62
kernel_3_0	207,64	85,91	328,27	164,83	37,06	885,81	233,22	195,07	290,98
total	301,1	138,82	456,52	252,2	60,37	1055,67	335,02	287,41	413,14



kernel_1_0	433,38	323,8	724,53	485,84	89,2	1400,92	677,28	575,61	748,46
kernel_1_1	209,36	130,22	290,38	187,48	69,83	647,95	271,98	244,44	281,9
kernel_2_1	500,62	159,6	698,03	374,26	200,55	2755,08	569,26	472,86	902,07
kernel_2_2	742,03	339,07	1064,48	593,65	119,2	2524,79	752,92	641,21	1009,76
kernel_2_3	526,78	237,24	589,52	270,96	87,92	4411,85	545,3	464,09	632,24
kernel_2_4	447,17	317,95	982,12	458,93	74,36	2328,72	633,63	548,67	792,03
kernel_3_0	644,59	266,71	1019,08	511,69	115,06	2749,93	724	605,6	903,32
total	482,42	222,42	731,44	404,07	96,72	1691,39	536,76	460,49	661,92



-
- Performance Prediction



Fraunhofer Institut
Techno- und
Wirtschaftsmathematik

Thanks to Fluent, Adapco, AVL

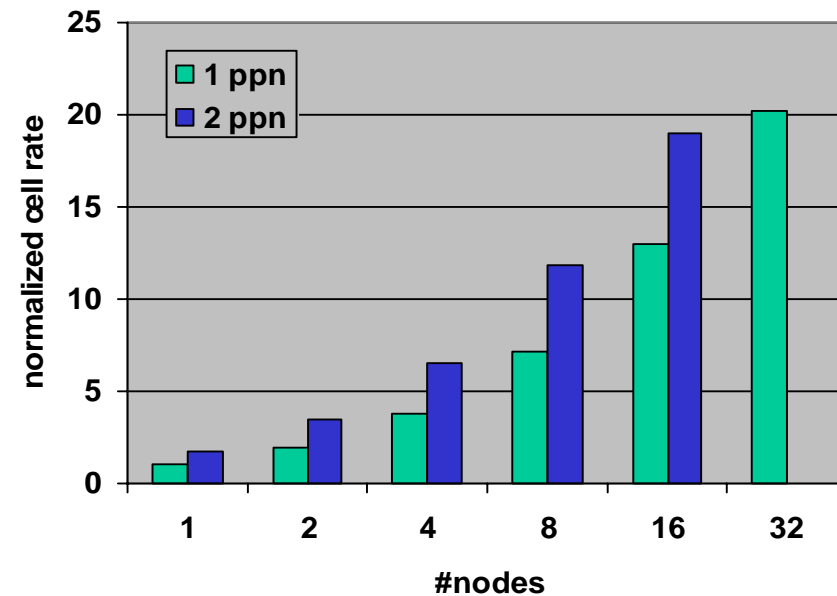
- **CFD finite-volume discretisation**
- **implicit and explicit, various models**
- **METIS, MPI**

- transonic flow around a fighter
- 847,746 hexahedral cells
- coupled explicit solver



Speed-up plot

Dual P4 2,4 GHz, Myrinet

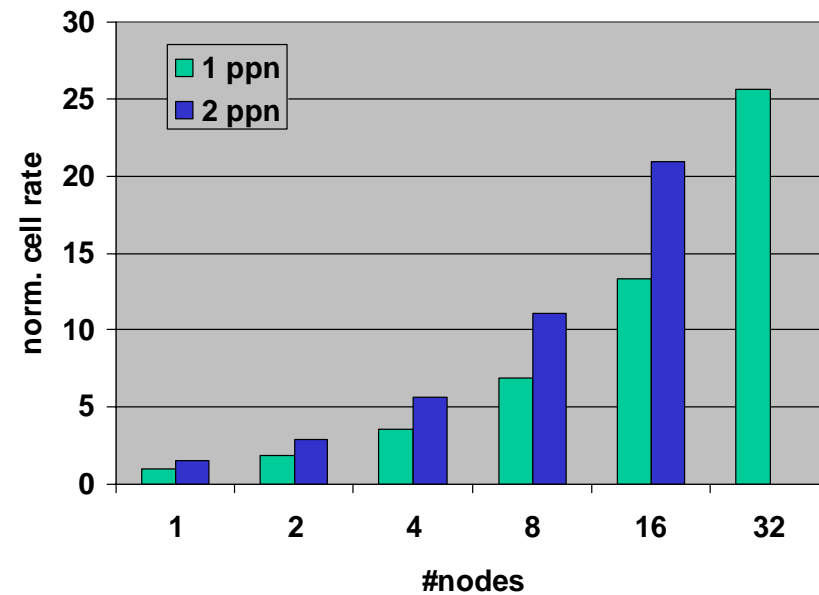


FLUENT: Scalable Benchmark

- Flow through a cube with regular structure
- Number of cells per process is fixed
~350000 hexahedral cells per process
- Geometry can be generated locally

Speed-up plot

Dual P4 2,4 GHz, Myrinet



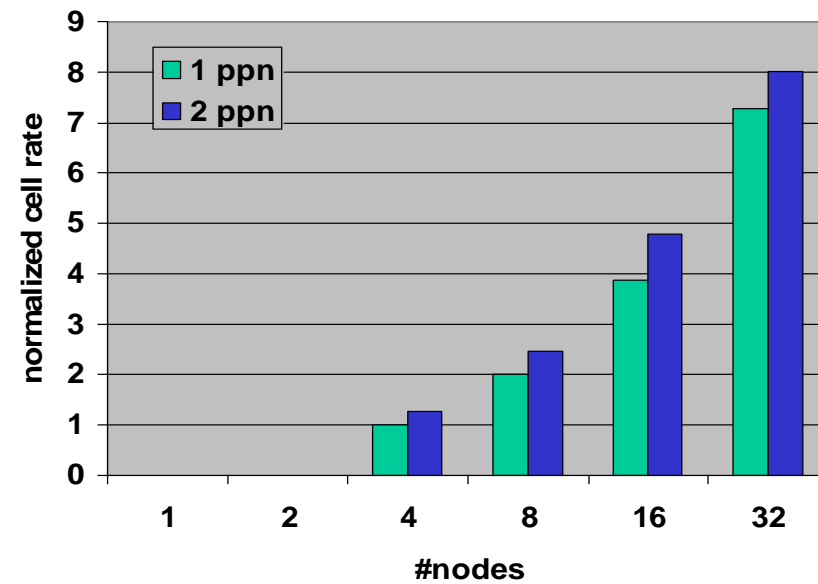
- CFD, finite volume, implicit solver
- METIS, MPI
- problem specific executable (F77)

- turbulent flow around a-class car
- 6 mio. hybrid cells



Speed-up plot

Dual P4 2.4 GHz, Myrinet

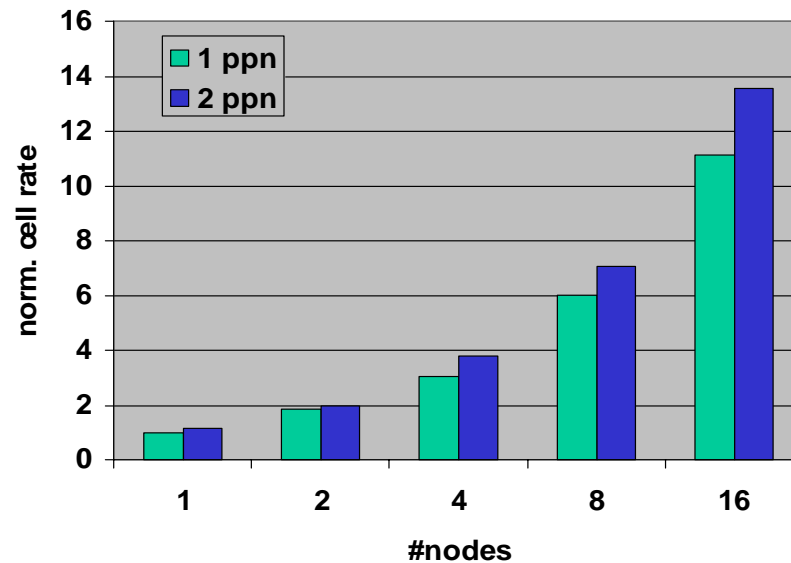


STAR-CD: scalable Benchmark

- Flow through a cube with regular structure
- Number of cells per process is fixed
~350000 hexahedral cells per process
- Geometry can be generated locally

Normalized Cell Rate

Dual P4 2,4 GHz, Myrinet



Performance Modeling

- **Methods to connect low-level benchmark results with application performance**
 - **Easy to use, redone by the user, applicable to commercial code**
 - **First Steps in this direction for CFD applications:**
 - Basic Structure: Loop over many small cells → Total time dominated by
 - Floating point operations T_{flop}
 - L1 Cache loads and stores T_{load} , T_{store}
 - L2 Cache misses and main memory access T_{mem}
-
- $T_{flop} = \#flops / v$ with v : theor. peak performance [Flop / s]
 - $T_{load} = \#loads \cdot 8 \text{ Byte} / \mu_{load}$ with μ_i : Cache bandwidth [Byte / s]
(Cachebench)
 - $T_{mem} = \#mem \cdot linesize / \mu_{mem}$ with linesize: Size of cacheline [Byte],
 μ_{mem} : Mem. bandwidth [Byte / s]
(Cachebench)

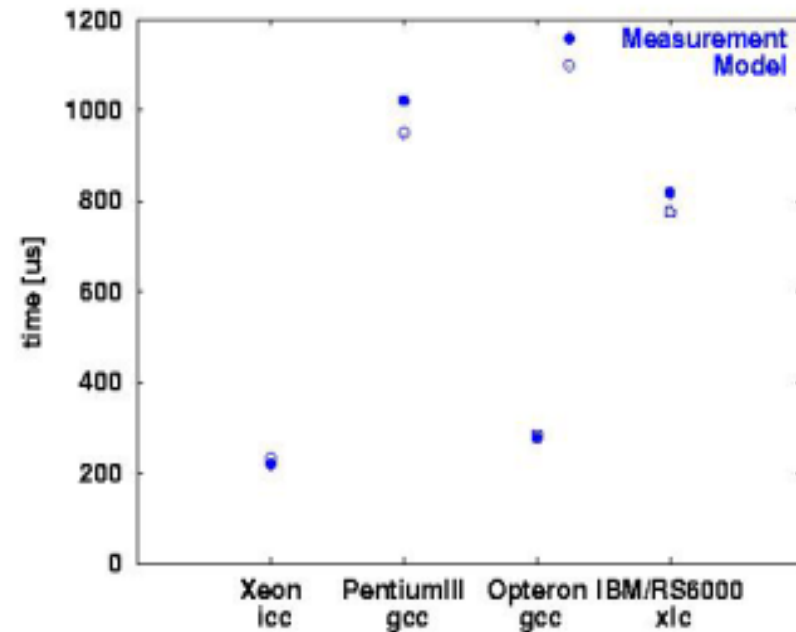
Using hardware counters to determine
#loads, stores, flops and mem as a guideline

→ Performance ParPac (ITWM)

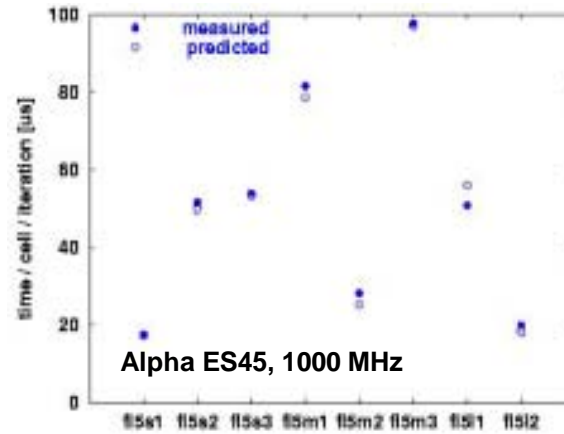
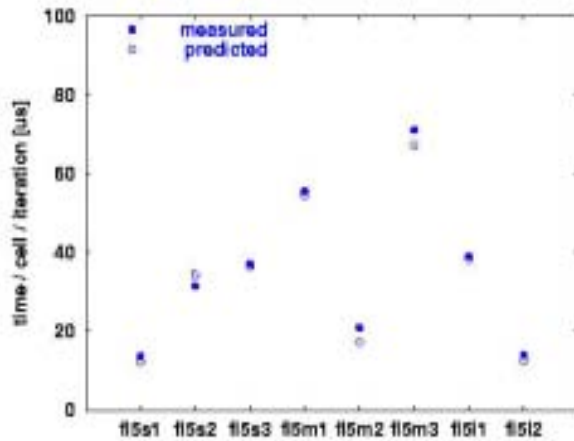
Characteristics of the application
and the case:

- #flops / load = 1
- #stores / load = 1/4
- #mem: measured on P4 dual nodes by keeping bus beasy
- #loads / cell : fitted

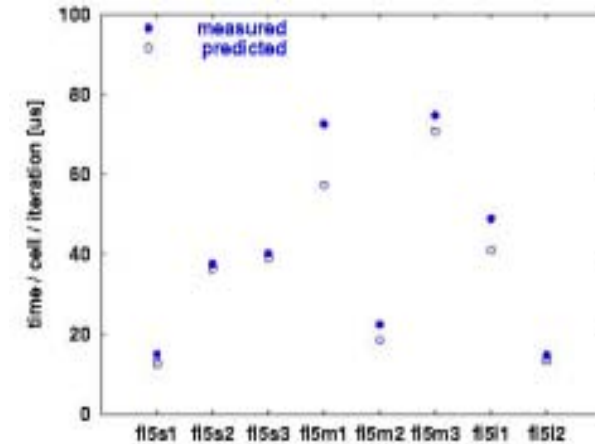
Performance ParPac



Itanium2, 1500 MHz



IBM pSeries690, Power4, 1300 MHz



Case specific information :

- Number of cells
- Number of partition boundary cells
- Number of neighbours per partition
- Number of messages $n \rightarrow$ scales with number of neighbours
- Data transfer \rightarrow scales with number of boundary cells

Need to know partitioning

Naive (optimal) assumptions:

Communication: $T_c = n (\lambda + s / \gamma)$

γ : Network bandwidth

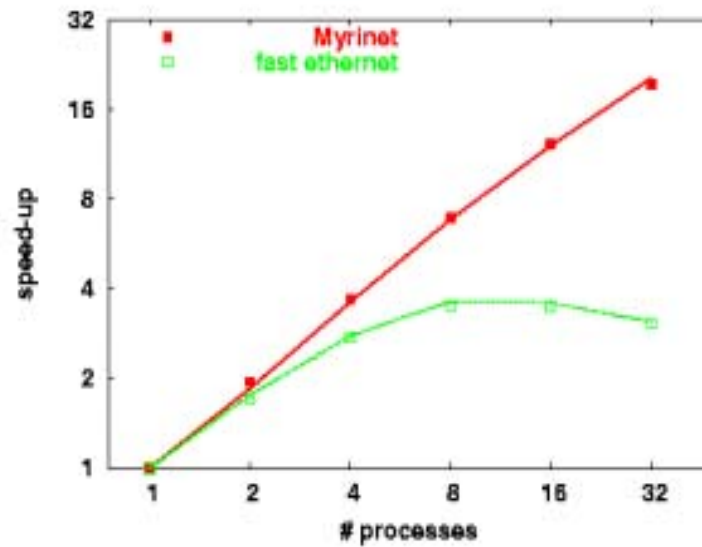
λ : latency (Beff Ping-Pong)

- mean message size s
- optimal communication / minimal #steps

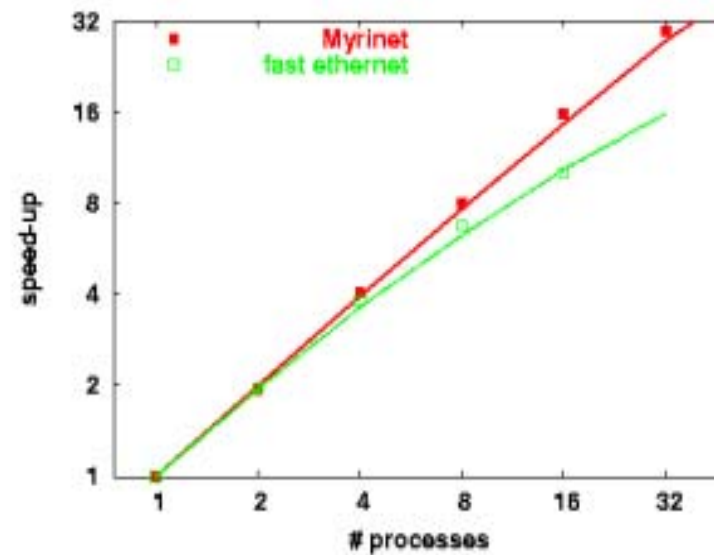
Time for calculation:

- scales with number of processors
- scales with problem size

- Flow in automotive valve port
- Implicit method
- 250000 hybrid cells



- exterior flow field around car
- Implicit method
- 3.6 mio hybrid cells

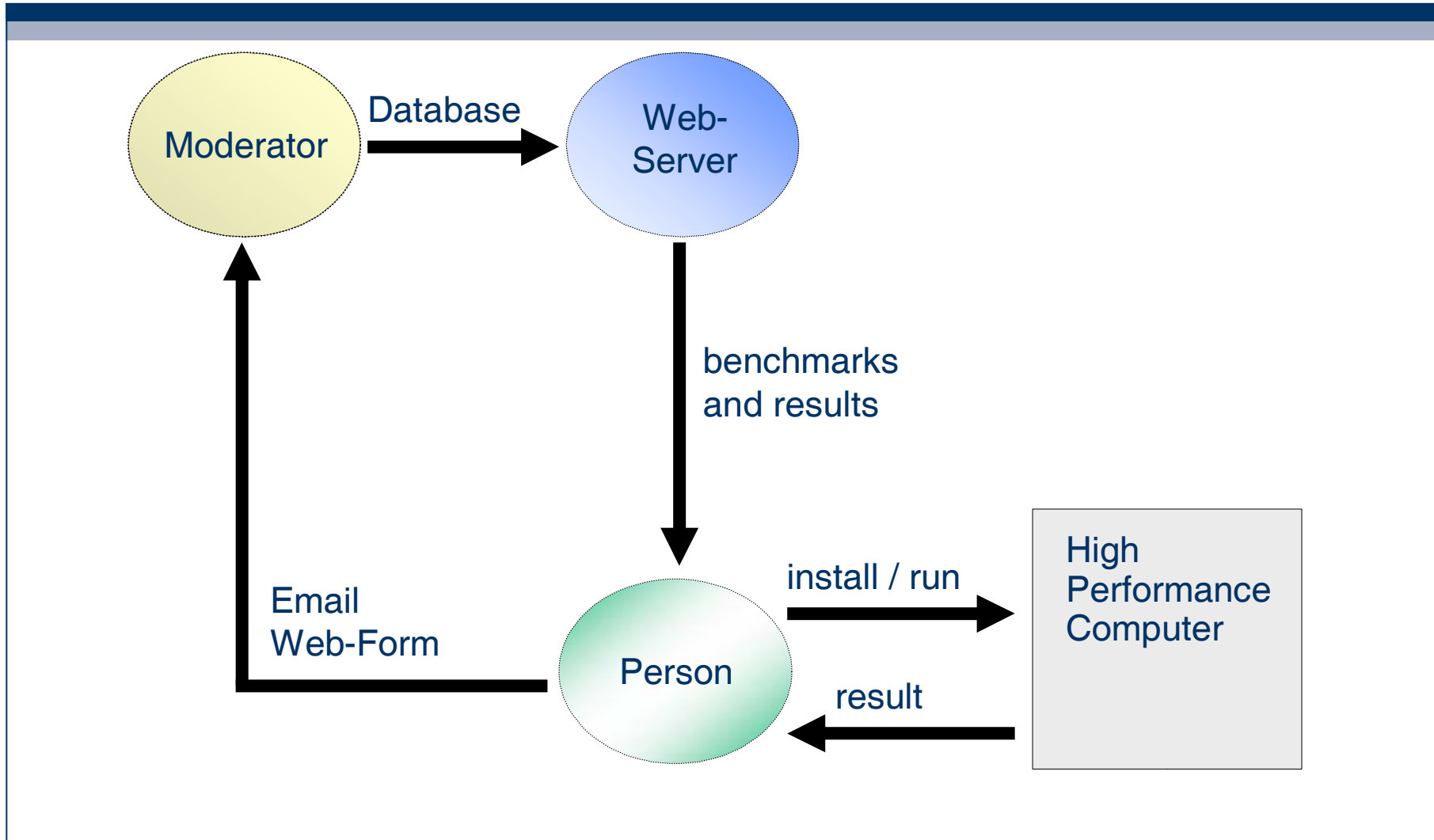




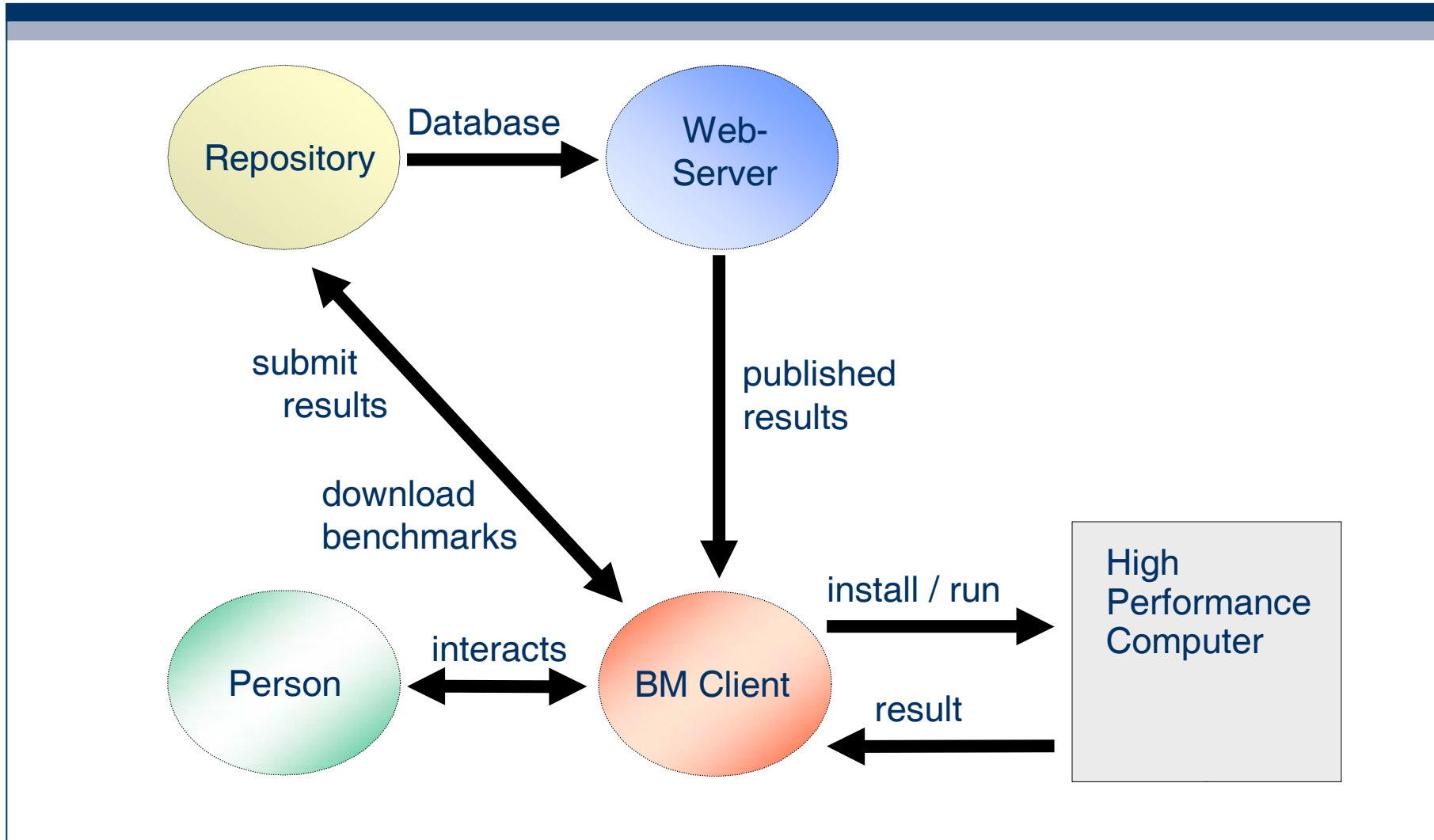
Component Software Architecture

Dr. Heinz Kredel, Matthias Merz
University of Mannheim,
IT-Center, L 15,16
68131 Mannheim, Germany
{kredel/merz}@rz.uni-mannheim.de

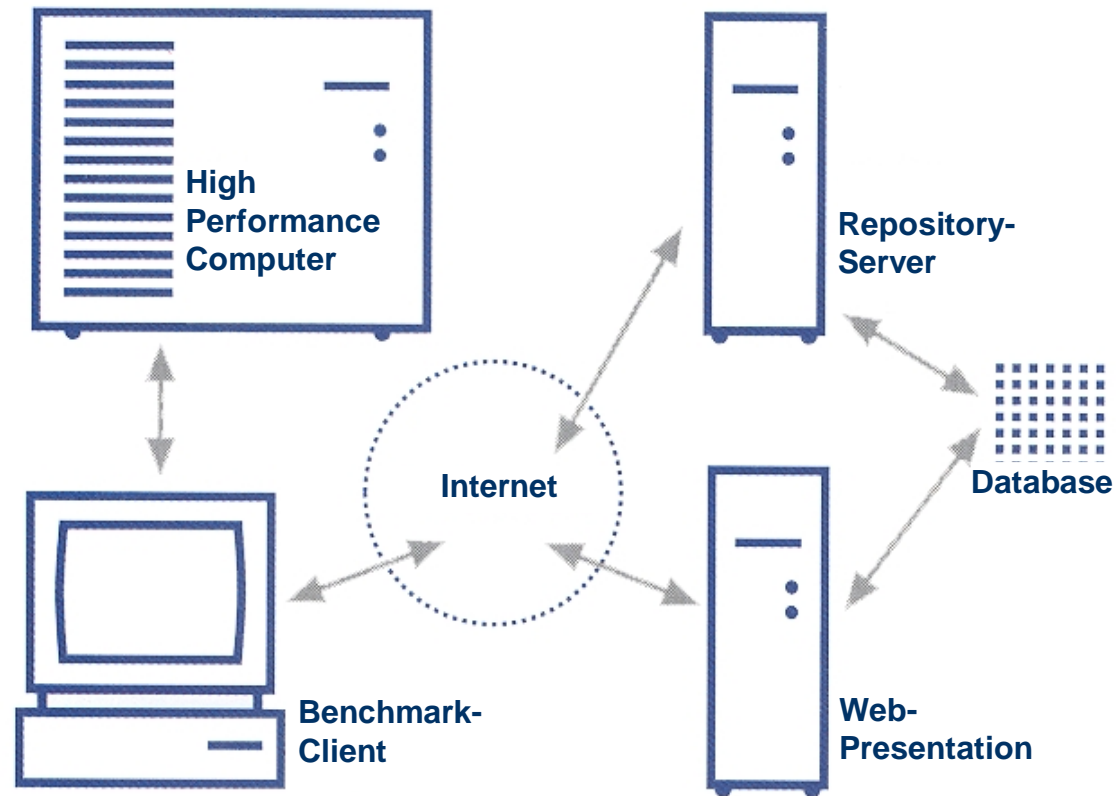
Benchmarking Process - Old



Benchmarking Process - IPACS



Component Software Architecture



Benchmark-Client:

- Provide essential configuration tasks
- Auto detection of current hard- and software environment of the HPC
- Based on this, downloading and executing of pre-configured benchmarks
- Transmitting benchmark results
- Realization:
 - Java (Portability)
 - Java Web Start
 - Data description XML
 - Communication via HTTP-Protocol



The screenshot shows the IPACS Benchmark Client configuration window. The window title is 'IPACS Benchmark Client'. The main content area is divided into several sections for configuration:

- Site Information:** Includes fields for Site Name, Site Area, Site City, Site Contact Person, Site Contact Mail, Site Country, Site Address, Site Web URL, and Site Contact Phone.
- Computer:** Includes fields for Computer Name, Computer Type, Computer Architecture, and Computer Manufacturer.
- Hardware:** Includes fields for Hardware Class, Number of Nodes, CPU Name, CPU Speed, Number of CPUs per Node, Number of Processors, Cache Name, CPU Capacity, CPU Vendor, and CPU IP Operation.
- Software:** Includes fields for MPI Library, OS Name, Compiler C, Compiler FOR, Tool Name, OS Version, and Compiler CXX.


There are also status windows on the right side of the interface, including 'Status Window' and 'To Use Window'.

Repository Server:

- Contains all benchmark files as source and binaries for a variety of architectures
- Manages information about the HPC (e.g. hard- and software configuration and their changes at all times)
- Reception, converting and administration of benchmark-results
- Realization:
 - Servlet-Container accepts HTTP-Request
 - Java-Servlet
 - Parses XML-Data
 - Database access
 - Response
 - Flexible persistence layer

Web-Representation:

- Project presentation
- Just-in-time presentation of Benchmark results
- 3 step presentation of results
- Search and navigation
- Download (Benchmarks)



The screenshot displays the IPACS Benchmark web interface. The browser window title is "IPACS Benchmark - Mozilla". The address bar shows the URL "http://www.ipacs-benchmark.org/beta_ipacs/benchmark.php?comp_id=17". The page features a navigation menu with links for HOME, PROJECT, DOWNLOADS, BENCHMARKS, LINKS, INTERNAL, and CONTACT. The main content area is titled "Computer Characteristics" and is divided into two sections: "Computer Description" and "Computer Configuration".

Computer Description		Computer Configuration	
Name	Test-Cluster	Software ID	Linux
Site	DF	Software ID Version	2.4.23-4480-099
Manufacturer	Compaq	Hardware Name	646
Computer Type	Desktop Cluster	API Library	ipacsb
Computer Architecture	Cluster	C Compiler	7.3
Date of Registration	2004-01-05	C77 Compiler	7.3
		C88 Compiler	<None>
		Last Change	2004-02-15 15:09:13

Below the computer characteristics, there is a section titled "Available Benchmarks Results". Under this section, there is a sub-section for "P881 PingPong (Print details)" with the following data:

# Processors	8
PingPong Max. Bandwidth (MB/sec)	444.85
PingPong Latency (sec)	0.16
PingPong Bandwidth of 1MB (MB/sec)	408.2
Creation Date	2004-01-05 17:10:40

The footer of the page contains the text "Copyright © 2002 ipacsbenchmark.org".



Repository Demonstration

www.ipacs-benchmark.org

Questions ?