Large-Scale Applications in Projects @ HLRS: Lessons to be learned for the future

Rainer Keller, HLRS

ORAP Forum 2007, Strasbourg

29.11.2007

Overview

- Overview of HLRS & of Working Group
- Large Scale Applications
  - HPC-Europa
  - TeraFlop workbench: BEST
  - DEISA & TeraFlop Workbench: Kop3D
- Lessons to be learned
- Conclusion

Overview of HLRS

- HLRS is one of three national HPC centers in Germany.
- We provide compute time to German research & industry.
- Currently a staff of about 100 persons, of which ~30 staff members.

NEC SX-8
72 nodes
576 procs
12.6 TFlops

NEC TX-7
32 procs
512 GB

Cray XD1 Cluster
~176 nodes
Various Interconnects
IPath, Ethernet, MX

DGrid-Cluster
28 nodes Dual-WoodCrest
7 Cell Blades
Coupled via IB

Overview of application areas per machine

+ Pre- / Post processing
+ Post-Mortem Performance Analysis
- Non-Cache aware apps

+ Engineering Application
+ e.g., large, dense Matrices
+ Regular code
- Highly abstract C++ code
- Strided memory access

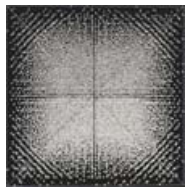
+ Small scale MPI
+ Nice Interconnect
+ Hybrid codes
- Hard to program…

+ ISV codes
+ Medium Scale MPI jobs
- Not as large an SMP…
- Non-Cache aware apps

Overview of WG “Applications, Models and Tools”

- Working Group of seven research members.
- We support application development in the area of:
  - Parallel programming models (OpenMP and MPI)
  - Parallel tools:
    - **Support and development with IDEs (Eclipse/PTP)**
    - **Parallel debugging (Marmot, DDT, Open MPI&Valgrind)**
    - **Parallel performance analysis (Vampir, Opt, Paraver)**
- Currently working on projects such as:

HPC-Europa
Pan-European Research Infrastructure on High Performance Computing



SFB716

int.eu.grid

ParMA

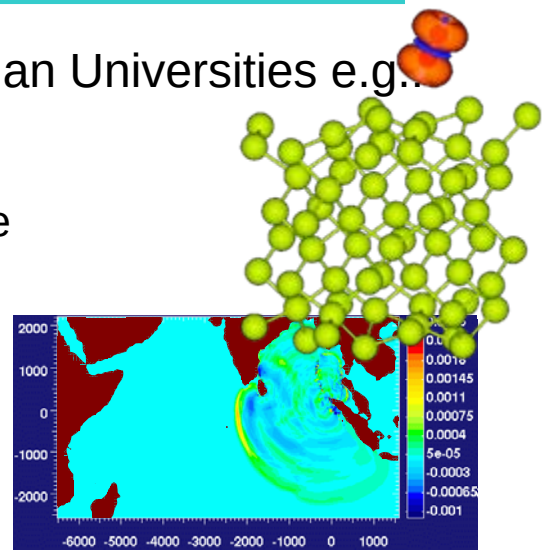
PACX-MPI

OPEN MPI

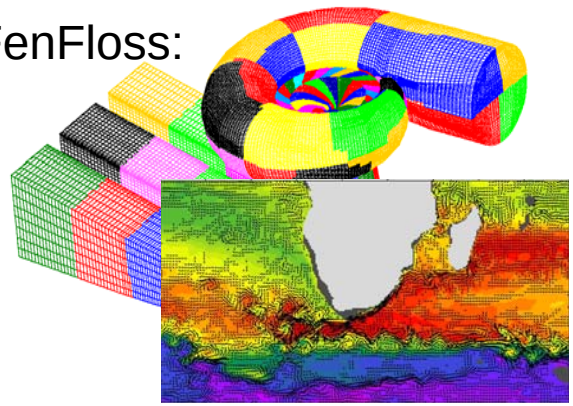
H L R S

Large Scale Applications

- With HPC-Europa, we hosted 112 guests at German Universities e.g.:
  - Oxygen absorption of Silicium:
    Dr. Lucio Colombi-Ciacchi, University of Cambridge
  - Tsunami application:
    Dr. Xing Cai, Simula Research Lab, Norway

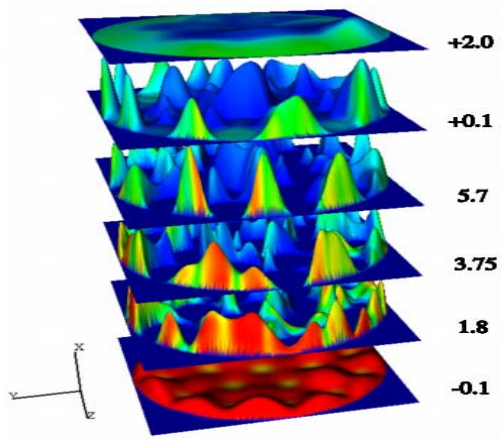


- Within the TeraFlop Workbench, 12 applications are being optimized
  - Instationary flow in a Francis Turbine with FenFloss:
    Dr. Albert Ruprecht, IHS University of Stuttgart
  - The Agulhas thermohaline circulation
    Dr. Arne Biastoch, IFM-Geomar, Kiel

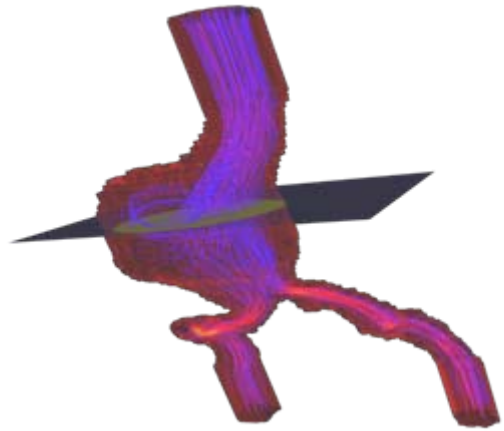


TeraFlop Workbench Applications: BEST

- **B**oltzmann **E**quation **S**olver **T**ool,
  Developed at LSTM & HLRS
  Dr. Peter Lammers, HLRS
- Solver for incompressible flows based on various LB models plus diffusion/chemical reactions
- Trivial grid (orthogonal-equidistant) but
- Well suited for flow through highly complex geometries:



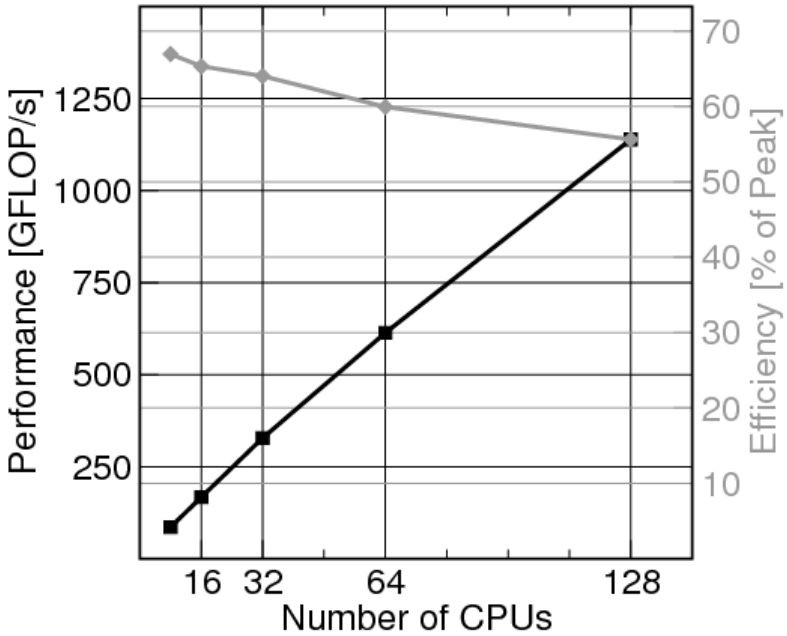
Flow through a fixed bed reactor



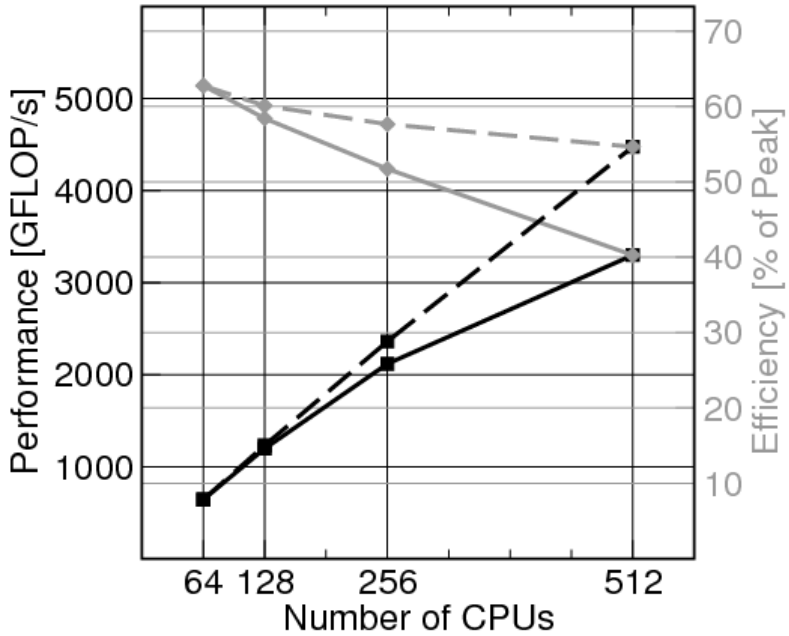
Flow through an aneurysm

TeraFlop Workbench Applications: BEST

- Results of strong scaling on up to 64 nodes NEC SX-8:



126 Mio Grid Points

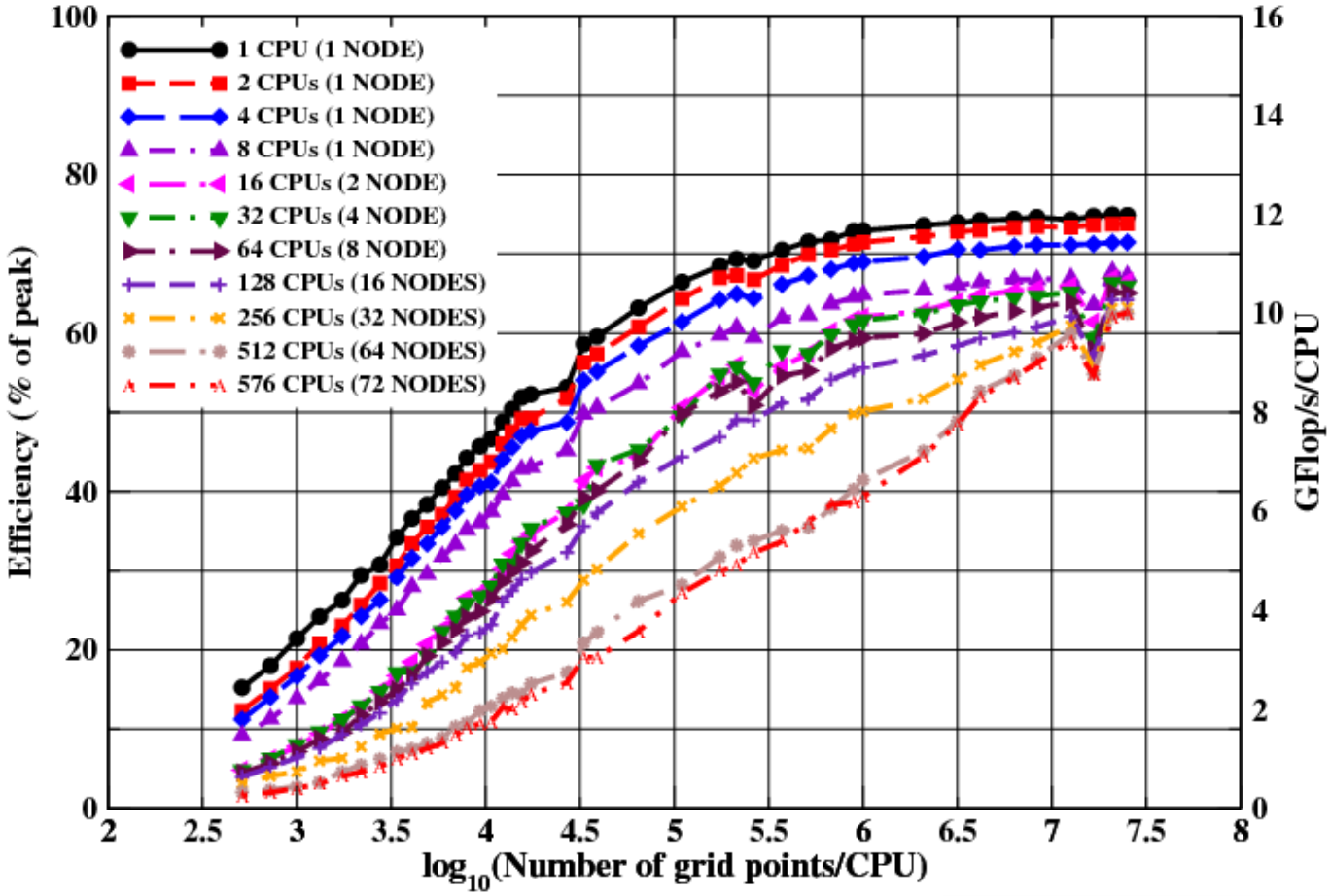


444 Mio Grid Points &
7 Mio Grid Points / node

- Sustained performance (black) -- Efficiency of vector peak (gray)

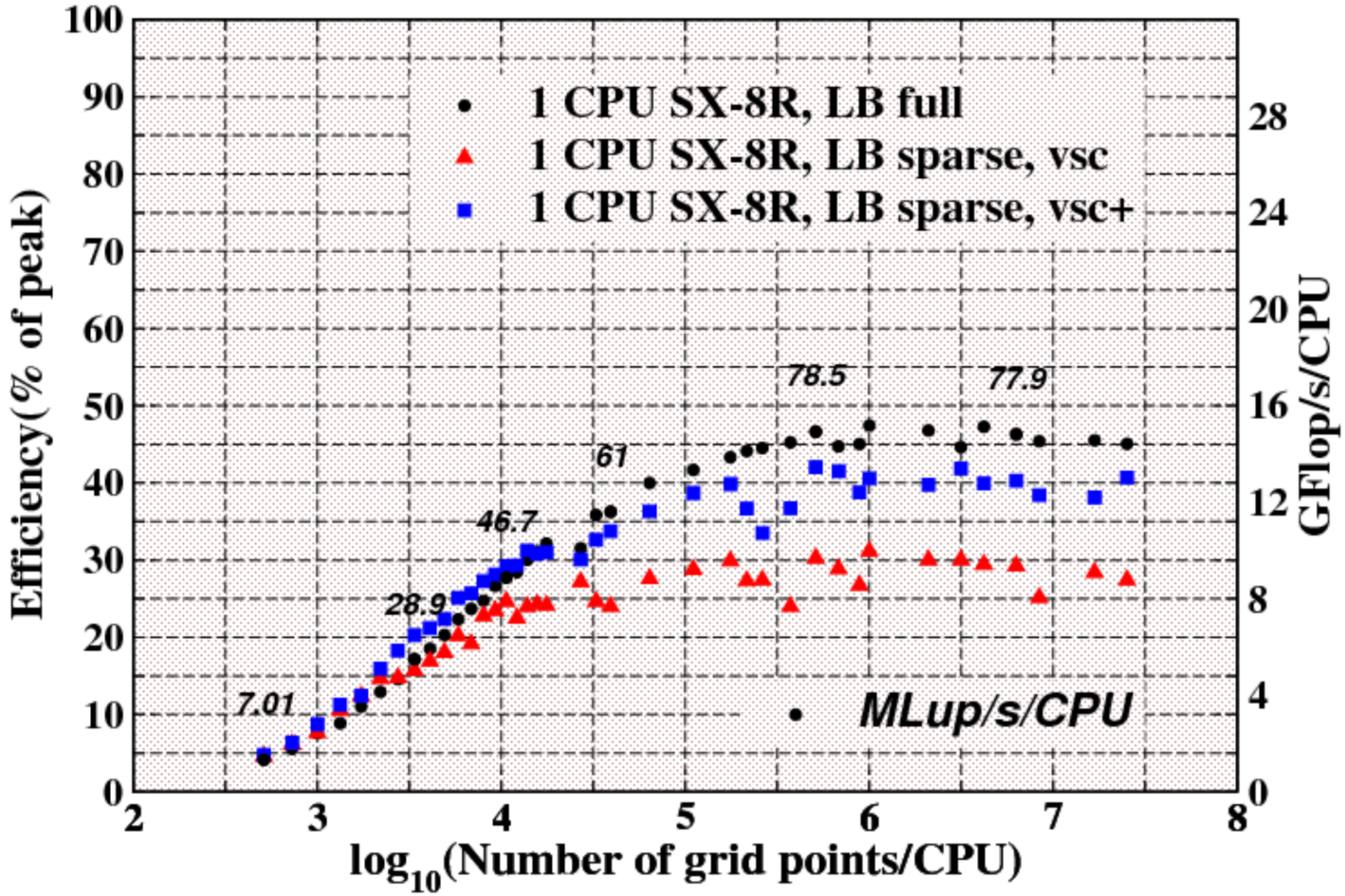
TeraFlop Workbench Applications: BEST

- Results of weak scaling tests for up to the whole machine:



TeraFlop Workbench Applications: BEST

- Performance on Next Upgrade, NEC SX-8R:



Now, a provocative question:

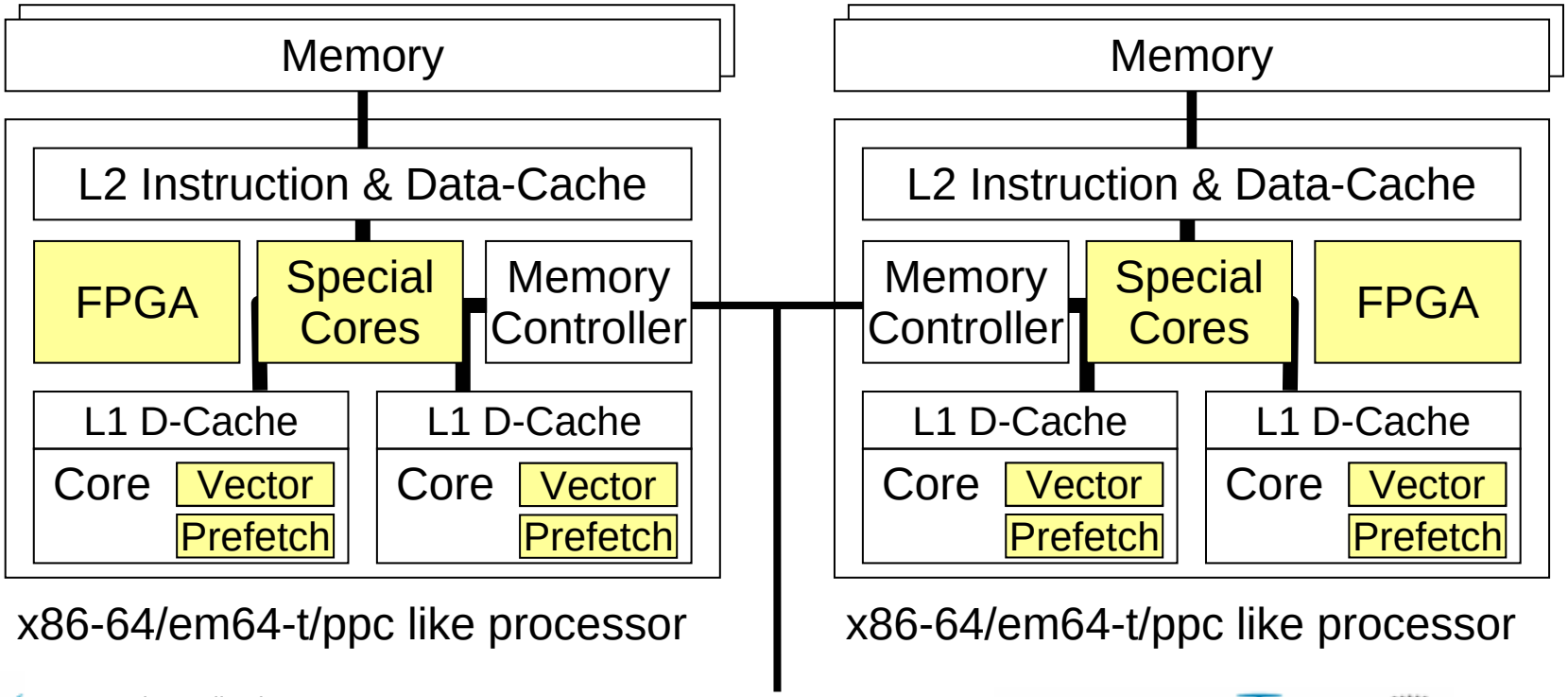
What will future architectures look like?

What will future architectures look like?

- Equally provocative as:

  What will future processors look like?

- Probably a blend of different technologies:



What will programming models look like?

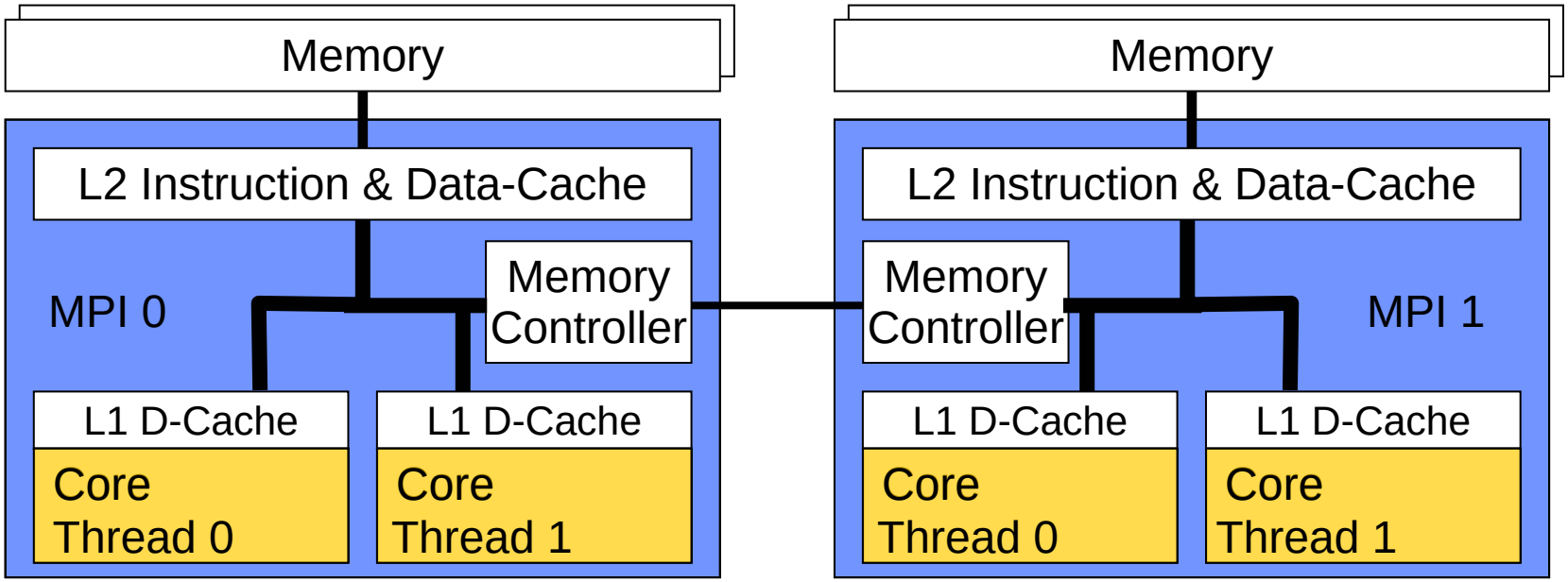
- Have You seen a 10^3 fold improvement in
  - Programming language design?
  - Compiler optimization capabilities?
  - Effectiveness of Human programming capabilities?

- BUT: There has been a steady improvement in Parallel programming paradigms and implementations:
  - Programming languages (additions to C, and Fortran)
  - Compiler optimizations (IPO, Vectorization, Parallelization)
  - Effectiveness of Human…. (well)

- Parallelization paradigms will follow the same route:
  - OpenMP and MPI have been standardized on experience,
  - Both have been conservative (like programming languages),
  - Both will develop **conservatively** to better suite architectures.

What will programming models look like?

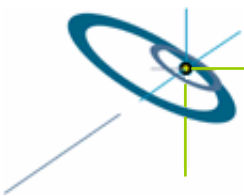
- Applications, libraries & compilers need to be adapted, issues are:
  - Thread-safety of applications & libraries
  - Knowledge of topology and memory hierarchy (coherency?)
  - Flexible pinning of threads to cores



- Applications will need to cope with heterogeneity (or **take advantage**)...

Heterogeneous Parallelism with Kop3D using PACX-MPI

(Harald Klimach, HLRS)

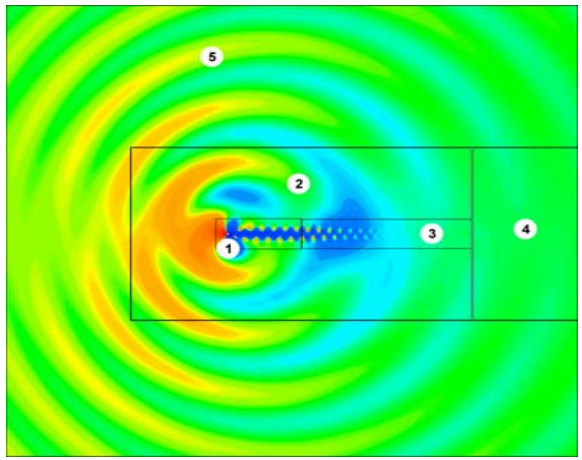


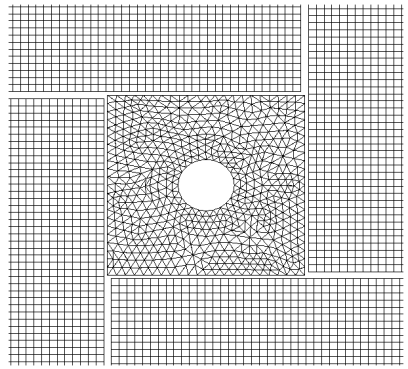
Kop3D: Heterogeneous Parallelism using PACX-MPI

- Jet noise emission is a top agenda in many countries.
- Direct simulation of aeroacoustic phenomena is a multiscale problem:
  - Physical effects generating noise happen at a scale m^{-2}-m^{-3},
  - Whereas propagation of sound happens at a scale of m^{2}-m^{3}.
- Different equations can be used in the simulation:
  - Noise generating flow has to be computed using Navier-Stokes.
  - For wave propagation, linearized Euler equations are sufficient.
- Kop3D does the coupling of these different codes.
- Developed at the Institute for Aero&Gas dynamics (IAG) and HLRS (Harald Klimach).

Kop3D: Heterogeneous Parallelism using PACX-MPI

- Aeroacoustic simulations by coupling different domains:

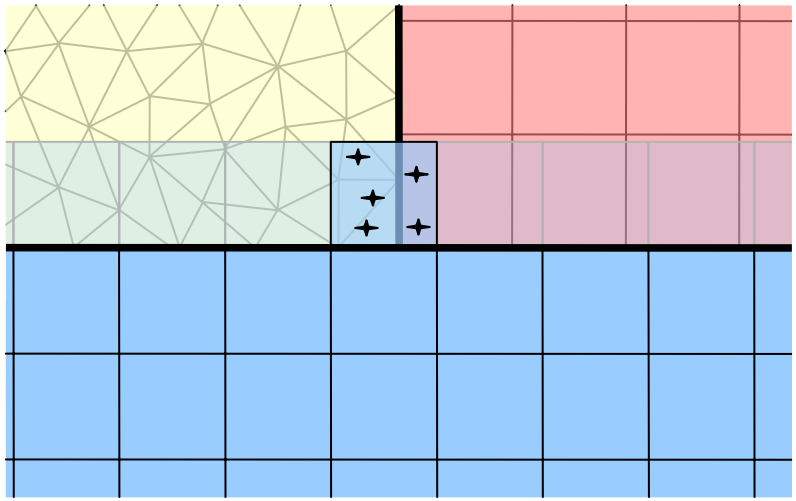




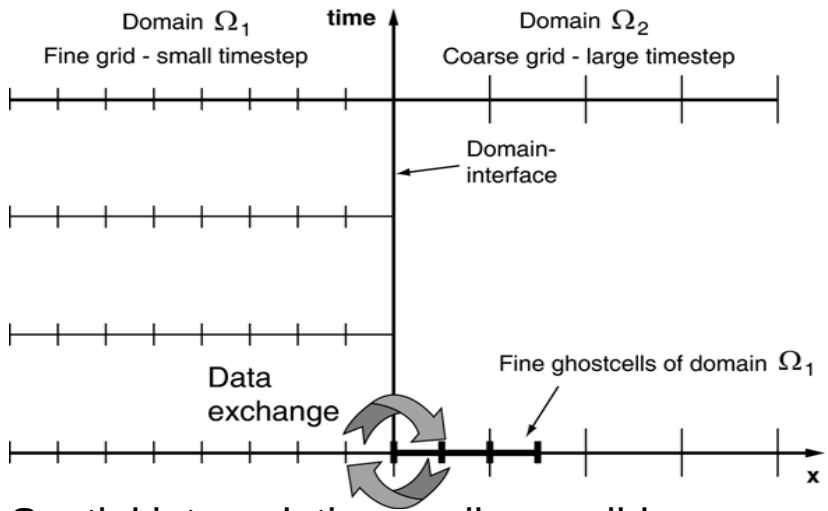
- Three parts:
  - (explicit) solver for structured meshes (for far field)
  - (explicit) solver for unstructured meshes (for complex geometries)
  - Coupling (using ghost cells)
- High order scheme: ADER (**A**rbitrary high order using **DER**ivatives)
- Several discretization methods (Discontinuous Galerkin, Finite Volume, Finite Differences)

Kop3D: Coupling of domains

- KOP3D uses a coupling in space and time.
- Neighboring domains can therefore have:
  - different spatial discretization
  - different timesteps
  - different solving equations



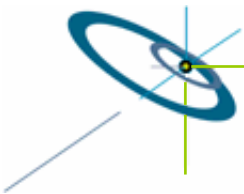
Spatial interpolation easily possible even across several domains, by only exchanging the values at discrete points.



Spatial interpolation easily possible even across several domains, by only exchanging the values at discrete points.

Kop3D: Coupling of domains

- Compute each domain in parallel
- Each domain may be distributed onto several processors itself
- Data exchange with simple MPI point to point communication
- Few synchronization points due to sub-cycling
- MPI communicator for each domain for internal communication for nearest neighbor communication



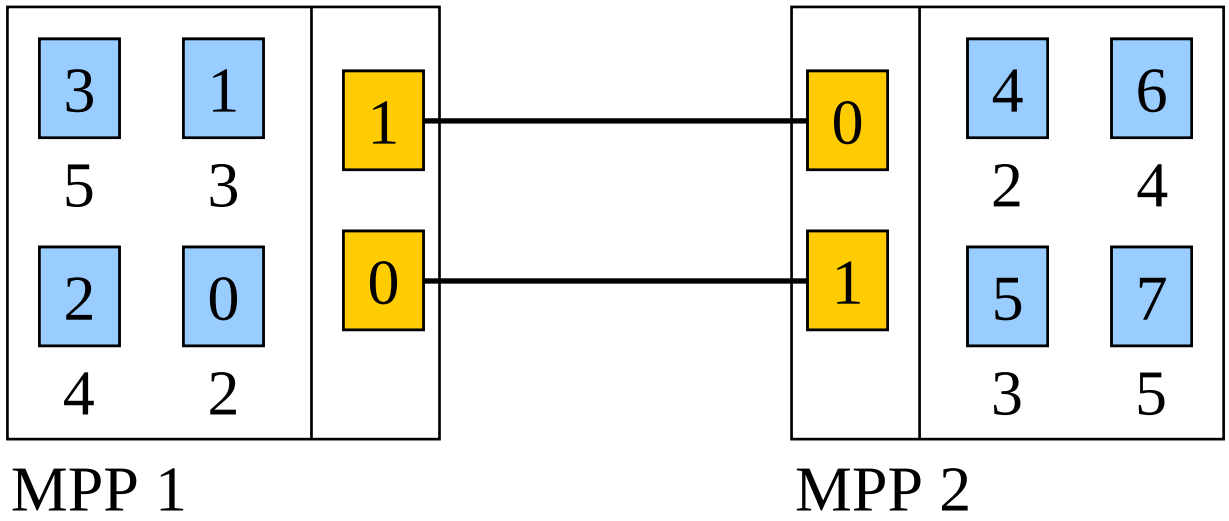
Kop3D: Coupling of domains

- **Goal**: map heterogeneous domains on the different architectures:
  - Structured code with Finite Differences:
    - **Easily vectorizable and**
    - **Due to regularity: good performance on vector architectures**
  - Unstructured code with Discontinuous Galerkin:
    - **High locality makes heavy use of caching**
    - **Runs well on PC-clusters**

PACX-MPI

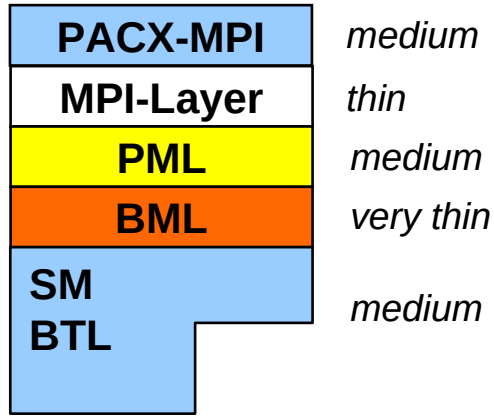
PACX-MPI: Overview

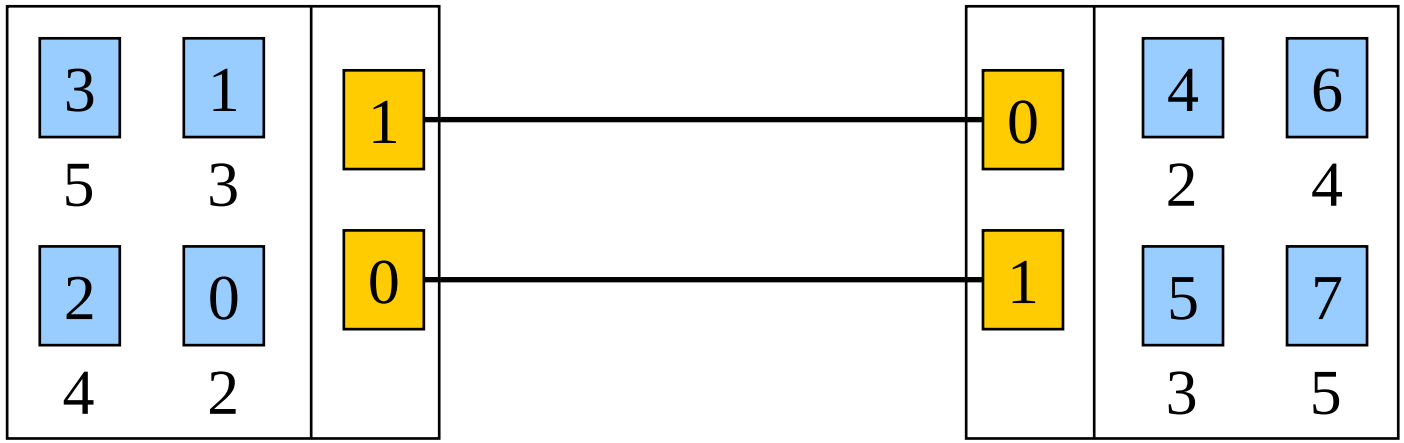
- A middleware to seamlessly run MPI-applications on a network of parallel computers (originally dev. in 1995 to connect Vector+MPP).
- PACX-MPI is an optimized standard-conforming MPI-implementation, application just needs to be **recompiled**(!)
- For C: pre-processor renaming: `MPI_Send` becomes `PACX_Send`.
- For Fortran: Function replacement @ link-step.



PACX-MPI: Overheads of PACX-MPI on top of Open MPI

- The additional overhead in case of **local** operations is **low**.
- The additional overhead in case of **non-local** operations is **prohibitive**.





Kop3D: Test case

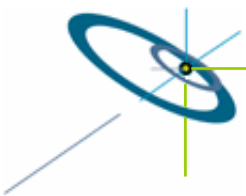
- Aeroacoustic 3D test case: scattering of sound waves at a sphere of 1 m radius.
- Unstructured Mesh:
  9874 DG elements around the sphere (5.5x3.5x3.5 m)
- Structured Mesh:
  ~42 Mio. FD cells for the far field (102.2x57x57 m)
- Spatial discretization with a 8^{th} order scheme
- Coupled architectures: NEC-SX8 with its frontend (IA64)

Kop3D: Performance

	0xSX	1xIA64	total
UNSTRUCT:	0,00	**2993,67**	2993,67
STRUCT:	0,00	23887,32	23887,32
KOP:	0,00	1012,37	1012,37
waiting:	0,00	0,00	0,00
KOP calculating time:			**1012,37**
Total CPU time:			**27893,35**
Total elapsed:			**27924,78**

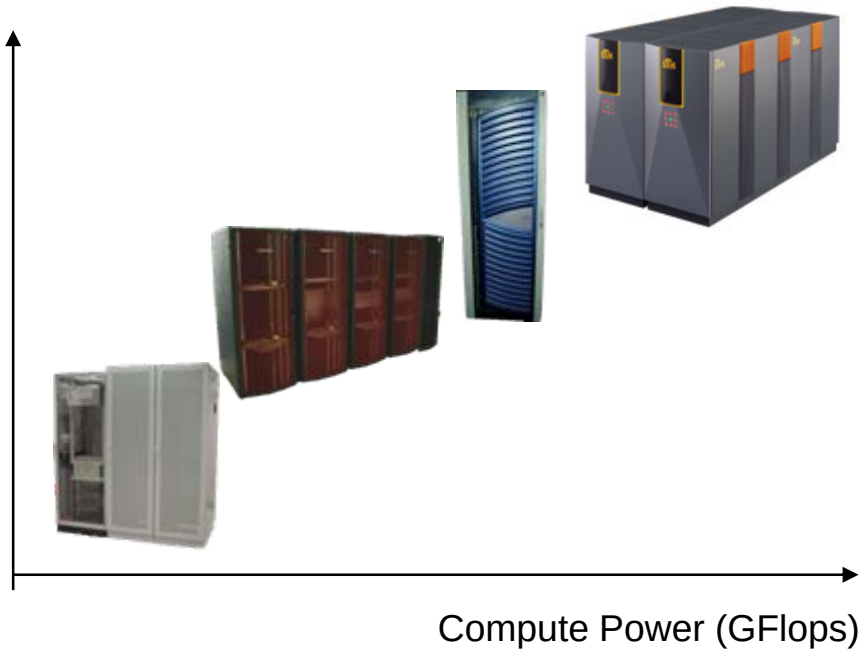
	1xSX	0xIA64	total
UNSTRUCT:	7745,82	0,00	7745,82
STRUCT:	**2870,66**	0,00	2870,66
KOP:	321,15	0,00	321,15
waiting:	0,00	0,00	0,00
KOP calculating time:			**321,15**
Total CPU time:			**10937,62**
Total elapsed:			**10966,23**

coupled	1xSX	1xIA64	total
UNSTRUCT:	0,00	3019,24	3019,24
STRUCT:	2869,46	0,00	2869,46
KOP:	368,11	186,10	554,21
waiting:	10,98	153,22	164,20
KOP calculating time:			**390,02**
Total CPU time:			**6278,72**
Total elapsed:			**3207,09**



Lessons to be learned

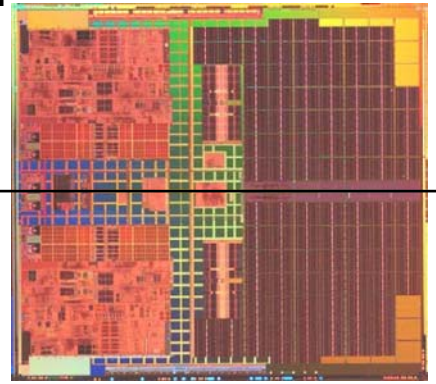
- Compute power has never been so abundant.
- Compute power has never been so varied (w/ regard architecture).
- Computers will (probably) never again be so easily programmable?



- To quickly achieve good performance on many systems, one has to abstract concepts with regard to:
  - Parallelization (MPI, OpenMP, Threads, Microtasking)
  - Compiler-specifics
  - OS-dependencies

Lessons to be learned

- To get performance on future processors, users will
  - Need to Parallelize!
  - Worry about memory hierarchies
  - Worry about thread placement
  - Use specialized libraries (petsc, ATLAS, liboil)

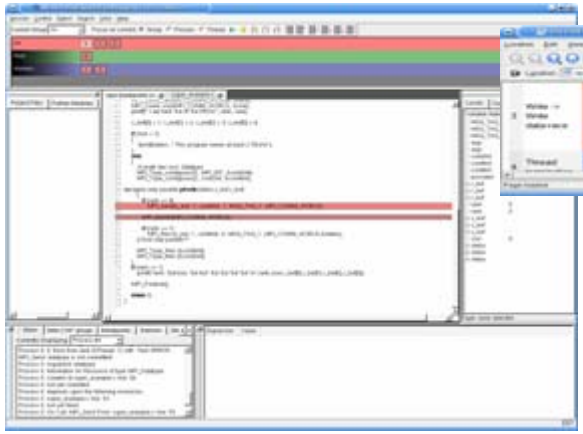


- However, there will be applications with algorithms that suite one particular architecture best (Multi- … Scale/Physics/Reaction)

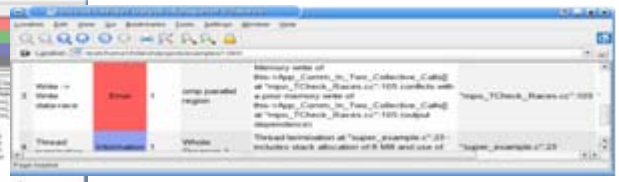
  Are they suited for Coupling?

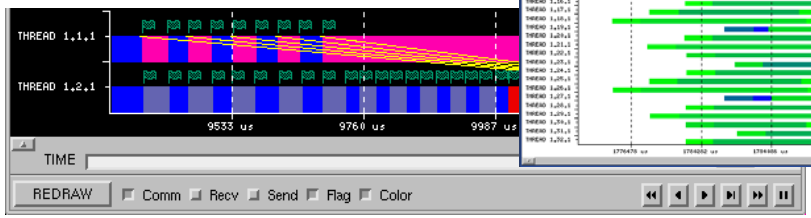
Lessons to be learned

- In order to achieve good performance, users will need **better tools**!

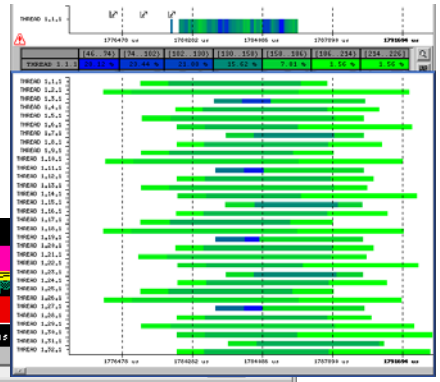


Marmot & DDT

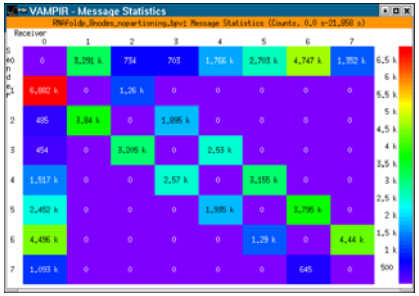




Marmot & Intel Thread Checker



Paraver & MPI-Peruse information



Vampir

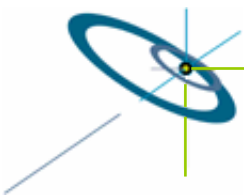
```
==11278==          read of size 1
==11278==          02321E: memcpy (../memcheck/mac_replace_strmem.c:256)
==11278==          0F6: MPID_SHMEM_send_short (mpich/../shmemshort.c:70)
.. 2               MPIch-functio      d ...
==11278==          92BA: MPI_Send     (mpich/src/pt2pt/send.c:91)
==11278==          048F28: main (      :4)
==11278== Address 0x4158B0EF is 3 b     a block of size 40 alloc'd
==11278==    at 0x4002BBCE: malloc (../../coregrind/vg_replace_malloc.c:160)
==11278==    by 0x8048EB0: main (mpi_murks.c:39)
```

MPI Memory-Debugging in Open MPI

Conclusion

- To gain performance improvements with future processors, one has
  - To parallelize,
  - To take into account complexity of multiple cores,
  - To take into account complexity of multiple memory hierarchies.
- Current Shared memory and distributed memory programming paradigms **are** suited for multi-core.
- All software hierarchies need to be adapted to follow suite.
- Or will we all do NVIDIA Graphics card programming???

Thank You very much.



Acknowledgements

- Peter Lammers, HLRS
- Xing Cai, Simula Labs
- Lucio Colombi-Ciacchi, IZBS, Freiburg / Karlsruhe
- Harald Klimach, HLRS
- Sabine Roller, HLRS