

Clustering T3Es for Metacomputing Applications

*Michael Resch, Thomas Beisel, Holger Berger, Katrin Bidmon,
Edgar Gabriel, Rainer Keller, Dirk Rantzau,*



*Allmandring 30
D-70550 Stuttgart
Germany*
resch@hirs.de

ABSTRACT:

This paper presents an environment which enables the coupling of different supercomputers to overcome the limitations of a single computing system. This requires an extension to MPI, since MPI provides no interoperability-features. A library called PACX-MPI is presented which provides the user with a distributed MPI environment with most of the important functionality of standard MPI. First results were achieved coupling the Cray T3E at the Pittsburgh Supercomputing Center and a Cray T3E at the High Performance Computing Center in Stuttgart. More recently the coupling of two Cray T3E at the High Performance Computing Center in Juelich and one Cray T3E of Stuttgart succeeded.

The first application we have run on the coupled T3E's is the flow simulation package URANUS. Problems and strategies with respect to metacomputing are discussed briefly for this code. Another application using this library is a Molecular Dynamics Code called P3T-DSMC. This program was developed for general particle tracking problems. We will present in this paper results for both applications.

KEYWORDS:

Metacomputing, Distributed MPI, Flow simulation, Monte Carlo Methods, Cray T3E

Overview

A difficulty in simulating very large physical systems is, that even massively parallel processor systems (MPP) with a large number of nodes or parallel vector processors (PVP) may have not enough memory and/or not enough performance. There are many examples of these *grand-challenge* problems: CFD with chemical reactions or crash simulations of automobiles with persons inside. A possible way to overcome these limitations is to couple different computational resources. Although there are different definitions of *metacomputing*, this is how in the following we use this term.

In summer 1996 a G7 project motivated an attempt to couple Cray T3E's at Pittsburgh Supercomputing Center (PSC) and the University of Stuttgart (RUS) across the Atlantic Ocean to establish a virtual

system with 1024 nodes and a theoretical peak performance of 675 GFlops. The PACX-MPI library [1] from RUS was used to allow an application to use standard MPI calls and work on this virtual machine. At Supercomputing'96 it was shown that the approach worked for a small demonstration experiment. During spring 1997 PACX-MPI was further developed and first application results could be presented at CUG '97 [6] .

At Supercomputing'97 the further progress was presented. A metacomputer consisting of the same machines like a year before and connected with a dedicated network connection with a bandwidth of two 2Mbits/second enabled the demonstration of two real-life applications.

The first application is a flow solver developed at the University of Stuttgart [2] and adapted for parallel computation by RUS [5]. It was coupled to a visualization tool developed by RUS to allow collaborative working and visualization [3].

The second application was an object-oriented Direct Simulation Monte Carlo Code which was developed at the Institute for Computer Applications (ICA1) of University Stuttgart for general particle tracking problems. [4] .

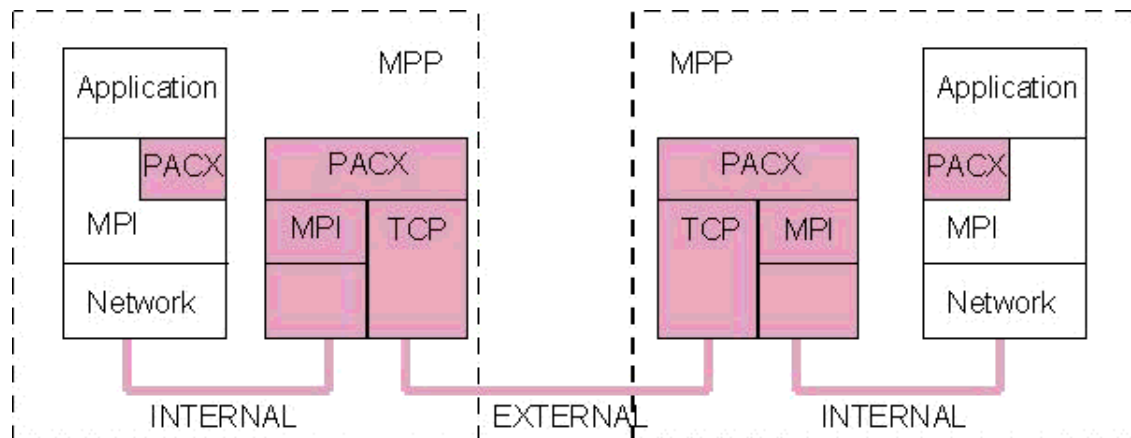
Results with both applications point out, that a loosely coupled application like the latter one seems to be optimal for metacomputing. Nevertheless, it was shown, that even closely coupled applications, like a flow simulation, can be adapted for metacomputing, so that one can achieve satisfying results. [5]

Basic Concept of the Interprocessor Communication Library

To reach the metacomputing goals, a library called PACX (PARallel Computer eXtension) was developed at RUS as an extension of the message passing library MPI. The main goals of this library include

- No need to change the source code
- A single system image to the programmer
- Usage of the vendor implemented fast MPI for internal communication
- Usage of a standard protocol for external communication.

PACX-MPI redirects the MPI-calls to its own library. For applications written in C this is done by using a macro-directive, and for Fortran applications by using the profiling interface of MPI. Thus PACX-MPI is a kind of additional layer between the application and MPI.



The PACX library determines whether there is a need for contacting the second MPP. If not, the library passes the MPI call to the local system unchanged, where it is handled internally. This guarantees usage of highly tuned vendor specific MPI implementations for internal communication. The overhead PACX imposes is very small on the CRAY T3E so that for calls on the local machine hardly any difference in performance can be seen.

Usage concept of PACX-MPI

To use PACX-MPI for an application, the user has to compile and link his application with the PACX-MPI library. No changes in code has to be done. The main difference for the user is the start-up of the application. First, he has to provide two additional nodes on each machine (like mentioned in the next section). An application which needs 1024 nodes on a T3E thus takes $2 \cdot 514$ if running on two separate T3Es.

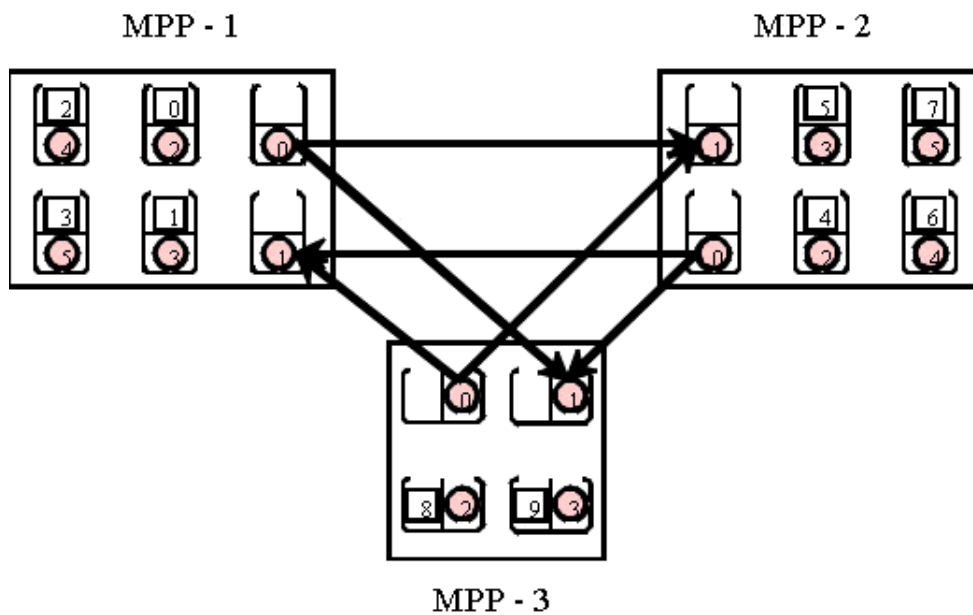
Then he has to configure a hostfile, which has to be identical on each machine. The hostfile contains the name of the machines, the number of application nodes, the used protocol for the communication with this machine and optionally the startup command, if he wants to make use of the automatic startup-facility of the new version PACX-MPI 3.0. Such a hostfile may then look like this:

```
#machine number of nodes protocol start-up command
host1 100 tcp
host2 100 tcp (rsh host2 mpirun -np 102 ./exename)
host3 100 tcp (rsh host3 mpirun -np 102 ./exename)
host4 100 tcp (rsh host4 mpirun -np 102 ./exename)
```

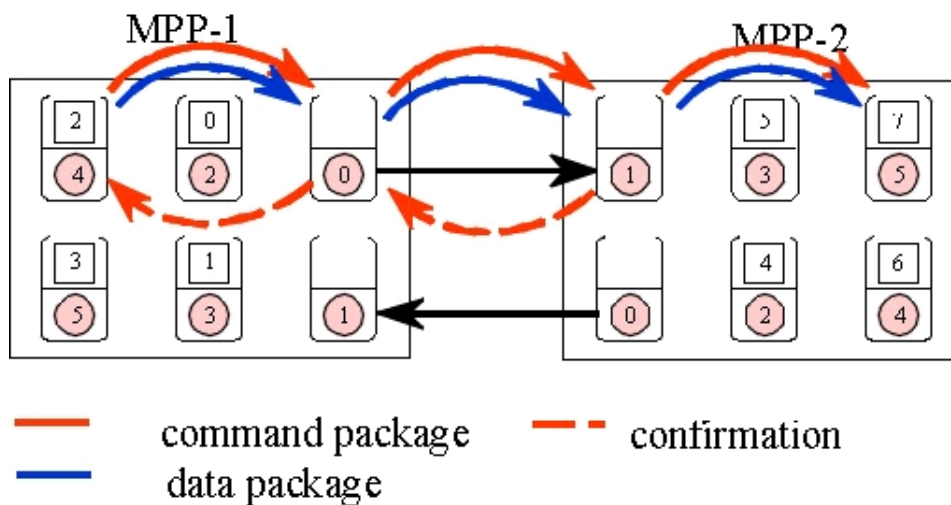
Technical concept of PACX-MPI

For the communication between the two machines, each side has to provide two additional nodes, one for each communication direction. The responsibility of each communication node is to receive data from the compute partition of its machine or from the network respectively, to compress or uncompress the data, and then transfer data to the network or the local compute nodes. Data compression uses the library LIBLZO.

Communication via the network is done with TCP sockets. The concept of using two extra communication nodes (one for handling all outgoing communication and one for the incoming communication) has turned out to be a useful design, because it is easier to handle the complex communications needs of a real world application this way. Additionally it is getting more and more important to have only a limited number of ports for the external communication, because most of the real big machines nowadays are protected by some kind of firewalls. Therefore it makes sense to use only a small number of well known ports that can be easily monitored and controlled.



PACX provides an MPI_COMM_WORLD across all machines. The creation of such a communicator requires two different numberings for each node; a local and a global numbering. In the figure above the local numbers are written in the circles and the global numbers are written in the boxes. The two additional communication nodes are not considered in the global numbering and are therefore transparent for the application. To explain the function of the communication nodes, we describe the sequence of a point-to-point communication. In the example below the global node two sends a message to the global node seven.



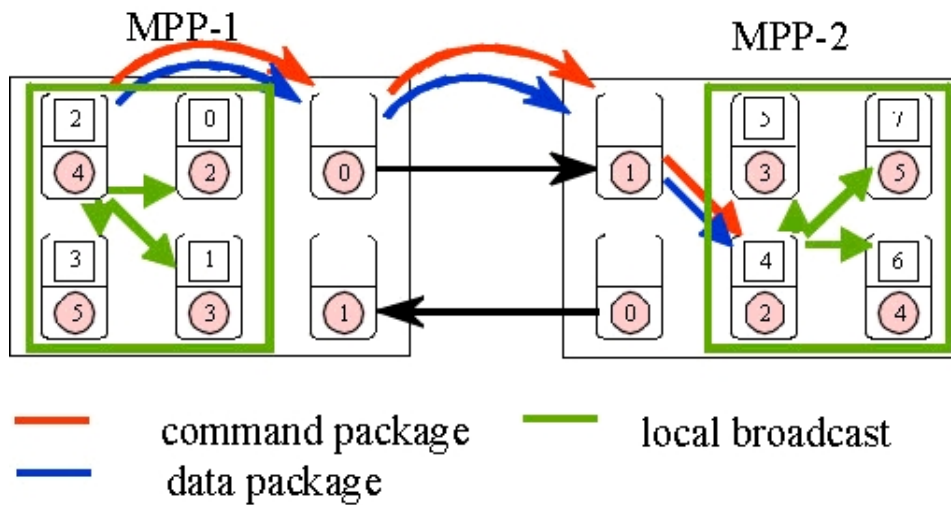
The sending node checks first, whether the receiver is on the same MPP or not. If it is on the same machine, it will do a normal MPI_Send. If not, it creates a command-package, which has the same function as the message-envelope in MPI, and transfers this command-package and the data to a communication node, the so-called MPI-server. The MPI-server compresses the data and transfers the command-package and the data to the destination machine. There the command-package and the data are received by the so-called PACX-server, the second communication node. Data are decompressed and passed on to the destination node.

In the previous PACX-MPI versions the receiver only checked, whether the sending node is on the same machine or not. For most situations this works well, but using more than two machines or different communicators this may lead to race conditions, where you need the possibility to buffer a message. For this, PACX-MPI 3.0 has a buffering system on the receiver node. The decision was to buffer messages without a matching receive on the application nodes rather than on a communication node, because this allows to distribute both memory requirements and working time.

The receiving node checks now first whether the expected message is an internal one. In case it is, it is received by directly using MPI. If it is an external message, the receiver node checks first whether the expected message is already in the buffer. If not, the message is received from the PACX-server.

For global operations, the concept has changed from the older version PACX-MPI 2.0 to the newer one. In the old version, some parts of the global operations were executed by the communication nodes. This could lead under certain circumstances to a blocking of the application and therefore in the new Version PACX-MPI 3.0 the communication nodes are no longer involved directly into global operations.

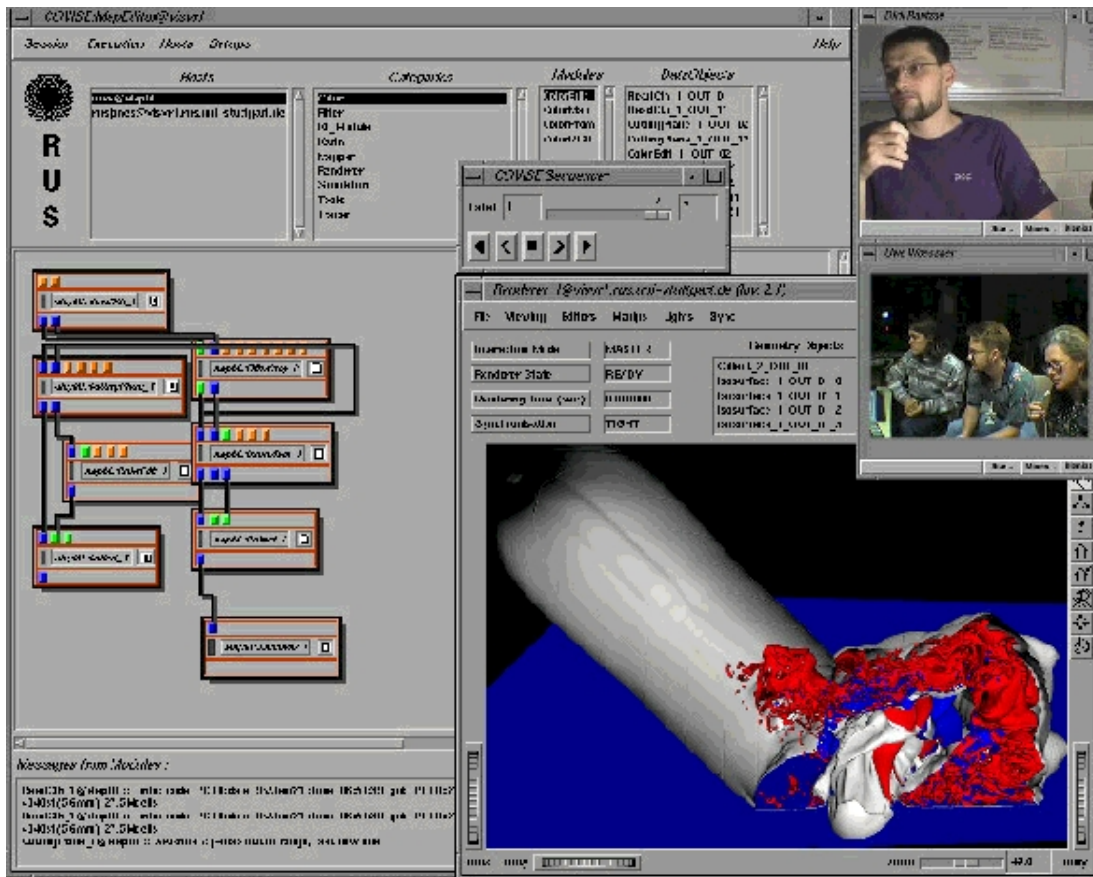
The sequence of a broadcast operation from global node 2 to MPI_COMM_WORLD is shown in the next figure. At first the root with the global number 2 sends a command-package and a data-package to the MPI-server. Then a local MPI_Bcast is executed. On the second machine the PACX-server transfers the command and the data-package to the node with the smallest number in this communicator on this machine. This node then does the local broadcast. This means that global communications are handled locally by nodes from the application part rather than by one of the servers.



So far PACX has been installed and tested on Cray T3E, Intel Paragon and Hitachi SR2201. Actually there is ongoing work to support heterogenous metacomputing, but nevertheless homogenous clustering will always be faster and therefore PACX-MPI will enable the optimization of homogenous metacomputing.

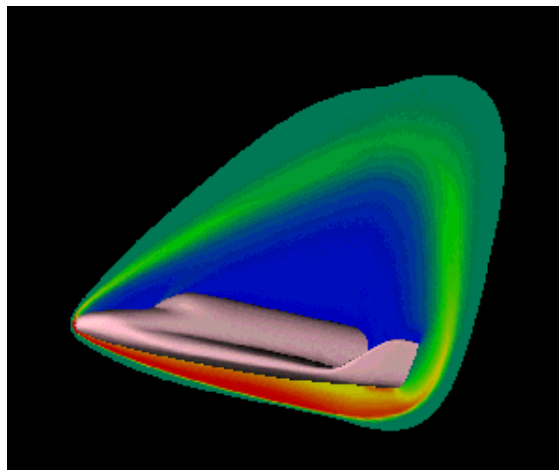
COVISE

To visualize metacomputing results tools for distributed visualization and control are needed in interactive metacomputing-scenarios. The HPC-center Stuttgart has a long tradition in distributed visualization and cooperative working. Within several EC-funded projects the distributed visualization system COVISE (Collaborative VISualization and Simulation Environment) [3] has been developed. COVISE is a distributed multi process environment, that allows to integrate simulation with pre- and post processing steps across heterogeneous computing resources. A visual programming paradigm allows to define the processing chain as depicted in the figure below. Typically the last processing step is a render module, which generates visual representations of a simulation content.



Results

The Navier-Stokes solver URANUS (Upwind Relaxation Algorithm for Nonequilibrium flows of the University of Stuttgart) was developed at the Institute for Space Systems at the University of Stuttgart. It is used for the simulation of nonequilibrium flows around reentry vehicles in a wide altitude-velocity range. URANUS was chosen for our metacomputing experiments because the simulation of reentry vehicles requires an amount of memory that can not be provided by a single machine.



To compute large 3-D problems, the URANUS code was parallelized at RUS. Since the code is based on a regular grid a domain decomposition was chosen. This results in a perfect load balancing and an easy handling of communication topology. An overlap of two cells guarantees numerical stability of the algorithm.

Initially the code was split into three phases (preprocessing, processing and postprocessing). Preprocessing was done sequentially. One node read in all data, did some preprocessing work, and distributed data to the other nodes. For the tests the Cray T3E of the Pittsburgh Supercomputing Center and the Cray T3E of the Stuttgart University were coupled using a dedicated network connection with a bandwidth of 2 Mbits/second and a physical latency of about 70 ms.

In the following we give the overall time it takes to simulate a medium size problem with 880.000 grid cells. For the tests we simulated 10 Iterations. We compared a single machine with 128 nodes and two machines with 2 times 64 nodes.

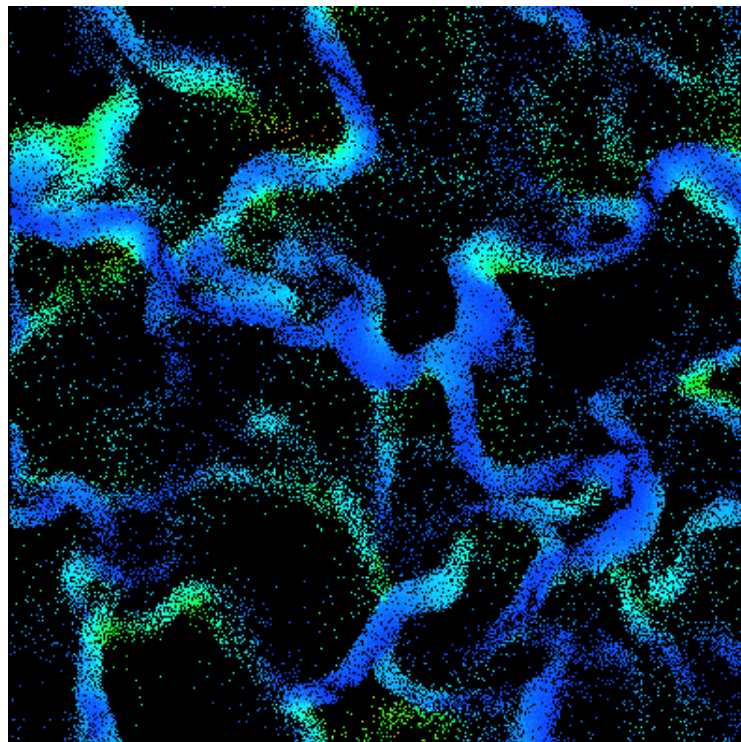
Method	128 Nodes using MPI	2*64 Nodes using PACX-MPI
URANUS unchanged	102.4	156.7
URANUS modified	91.2	150.5
URANUS pipelined	-	116.7

Obviously the unchanged code is much slower on two machines compared with the execution time on a single machine. However, the overhead of 50% is relatively small with respect to the slow network. The modified URANUS version, which uses fully asynchronous message passing improves the execution time for both cases. Using the so called "Message pipelining" [5] messages are only received if available. The receiving node may continue the iteration process without having the most recent data. This helped to reduce the computing time significantly but for convergence the number of iterations had to be increased by about 10%. Using this final version, a full space vehicle configuration using more than 1 million grid cells was run on 760 nodes during Supercomputing '97.

The second application is P3T-DSMC. This is an object-oriented Direct Simulation Monte Carlo Code which was developed at the Institute for Computer Applications (ICA1) of the Stuttgart University for general particle tracking problems. Since Monte Carlo Methods are well suited for metacomputing, this application gives a very good performance on the transatlantic connection.

Particles/CPU	60 Nodes using MPI	2*30 Nodes using PACX-MPI
1935	0.05	0.28
3906	0.1	0.31
7812	0.2	0.31
15625	0.4	0.4
31250	0.81	0.81
125000	3.27	3.3
500000	13.04	13.4

For small numbers of particles the metacomputing shows some overhead. But up to 125.000 particles timings for one time step are identical. During SC'97 this application was able to set a new world record for molecular dynamics simulating a crystal with 1.4 billion particles on two T3E using 1024 processors. This proved that metacomputing can be a tool to further push the frontiers of scientific research.



At the end of April 1998 another test was done with this application coupling a Cray T3E/600 with 512 nodes of the High Performance Computing Center in Juelich, a Cray T3E/900 with 256 nodes of the same institution and the Cray T3E/900 with 512 nodes of the High Performance Computing Center in Stuttgart. P3T-DMSC ran successfully on this cluster. This test pointed out, that the new PACX-MPI version is able to couple more than two MPPs.

Conclusions

We have shown that two T3E's can be successfully coupled even across the Atlantic Ocean and more than two MPPs are able to work on the same problem using the PACX-MPI library. We have shown that PACX-MPI works correctly, and the concept can be extended for heterogenous metacomputing.

URANUS has been shown to scale well on large numbers of nodes. Contemporaneously we've shown, that although one can use your code unchanged to do metacomputing, one can achieve a much better performance if you use asynchronous message passing or do some latency hiding. Especially for such a closely coupled application it makes sense to adapt the code for metacomputing.

P3T-DSMC is a loosely coupled application and therefore optimal for metacomputing. Applications like this, based on Monte Carlo Methods, show a very good performance even for long distance connections.

The ongoing work with PACX-MPI intends to support heterogenous metacomputing, too. The data-conversion routines are implemented and are now in the testing phase. Another main research field tries to improve the external network performance, and therefore in the near future PACX-MPI will support other network protocols in addition to TCP/IP. An other possible direction of PACX-MPI may also be to support a part of the functionality of MPI-2, especially the dynamic process start would be of very big interest for metacomputing.

References

- [1] Thomas Beisel, Edgar Gabriel Michael Resch. **An Extension to MPI for Distributed Computing on MPP's**. in : Marian Bubak, Jack Dongarra, Jerzy Wasniewski (Eds.) 'Recent Advances in Parallel Virtual Machine and Message Passing Interface', Lecture Notes in Computer Science, 75-83, Springer, 1997.
- [2] H.-H. Fruehauf, O. Knab, A. Daiss, U. Gerlinger. **The URANUS code - an advanced tool for reentry nonequilibrium flow simulations**. Journal of Flight Sciences and Space Research (19) 219 - 227. 1995.
- [3] A. Wierse **Performance of the COVISE visualization system under different conditions** in Visual Data Exploration and Analysis II, Georges G. Grinstein, Robert F. Erbacher (Eds.), Proc. SPIE 2410, pp.218-229, San Jose 1995.
- [4] Matthias Mueller, Hans J. Herrmann **DSMC - a stochastic algorithm for granular matter** in: Hans J. Herrmann and J.-P. Hovi and Stefan Luding (Eds.) Physics of dry granular media, Kluwer Academic Publisher, 1998.
- [5] Thomas Boenisch and Roland Ruehle **Adapting a CFD code for metacomputing** to be presented at 10th International Conference on Parallel CFD, Hsinchu, Taiwan, May 11-14, 1998.
- [6] Michael Resch, Thomas Beisel, Thomas Boenisch, Bruce Loftis, Raghu Reddy **Performance Issues of Intercontinental Computing** Cray User Group Meeting, San Jose, May 1997.

Author Biography

MICHAEL M. RESCH received his Diploma degree in Technical Mathematics from the Technical

University of Graz/Austria in 1990. From 1990 to 1993 he was with JOANNEUM RESEARCH - a leading Austrian research company. Since 1993 he is with the High Performance Computing Center Stuttgart. where he was and is involved in the European projects CAESAR and HPS-ICE. Furthermore he is responsible for the development of message-passing software and numerical simulations in international and national metacomputing projects. Since 1998 he is head of the Parallel Computing Group of the High Performance Computing Center Stuttgart. His current research interests include parallel programming models, metacomputing and numerical simulation of viscoelastic fluids.

- [mailto: resch@hls.de](mailto:resch@hls.de)
- URL: <http://www.hls.de/people/resch>

THOMAS BEISEL is a member of the System Administration Group of the High Performance Computing Center at Stuttgart (*HLRS*). He received the diploma degree in Mechanical Engineering from the University of Stuttgart in 1996. His diploma thesis was the initial version of the PACX-MPI library.

- [mailto: beisel@hls.de](mailto:beisel@hls.de)

HOLGER BERGER was a mathematical technical assistant at the University of Stuttgart. Since 1998 he is with NEC focusing on programming models.

- [mailto: berger@rus.uni-stuttgart.de](mailto:berger@rus.uni-stuttgart.de)

KATRIN BIDMON is currently studying Mathematics at the University of Stuttgart and is working now for nearly one year in the software development group of the PACX-MPI project, where she is responsible for data-conversion problems and heterogenous metacomputing.

- [mailto: katrin.bidmon@rus.uni-stuttgart.de](mailto:katrin.bidmon@rus.uni-stuttgart.de)

EDGAR GABRIEL is working since 01.06.98 with the Parallel Computing Group of the High Performance Computing Center in Stuttgart. There he is responsible for the parallel program development in the SFB 259 (URANUS) and part of the software development group of the PACX-MPI project. He got his Dipl.-Ing. for Mechanical Engineering of the University of Stuttgart in 1998.

- [mailto: gabriel@rus.uni-stuttgart.de](mailto:gabriel@rus.uni-stuttgart.de)
- URL: <http://www.hls.de/people/gabriel>

RAINER KELLER is actually studying Computer Science at the University of Stuttgart. He is working since more than one year with the software development group of the PACX-MPI project and is the main programmer of the latest PACX-MPI version.

- [mailto: rainer.keller@rus.uni-stuttgart.de](mailto:rainer.keller@rus.uni-stuttgart.de)

DIRK RANTZAU received the Diploma degree in Mechanical Engineering from the University of Stuttgart in 1993. From 1993 to 1994 he was involved in the RACE Project PAGEIN where he worked in the field of interactive steering of supercomputer simulations and collaborative working at the High Performance Computing Center of the University of Stuttgart. Since 1995 he is working in the Collaborative Research Center Rapid Prototyping established at the University of Stuttgart in the field of VR based virtual prototyping. His current research interests include collaborative virtual environments

for scientific visualization, 3D user interfaces and interaction techniques for computational steering.

- [mailto: rantzau@hls.de](mailto:rantzau@hls.de)
- URL: <http://www.hls.de/people/rantzau>

Acknowledgements

The authors gratefully acknowledge support from their home organizations HLRS and from PSC, as well as supercomputing time provided by Forschungszentrum Jülich GmbH.