A Message Passing Interface Library for Inhomogeneous Coupled Clusters

Martin Poeppe, Silke Schuch, Thomas Bemmerl
Lehrstuhl für Betriebssysteme,
RWTH Aachen, Kopernikusstr. 16,
52056 Aachen, Germany
E-mail: {martin, silke, thomas}@lfbs.rwth-aachen.de

Abstract

Clusters of PC containing mostly general purpose hardware have become more and more usable for high performance computing tasks in the past few years. Clustering existing systems or extending cluster installations leads to the problem of inhomogeneous cluster installations with a broad variety of hardware and even operating systems. This paper describes the efforts of the MetaMPICH project to deal with multi-platform inhomogeneous clusters, using different networks and processor types. An approach is presented which on one hand hides the heterogeneity of soft- and hardware from the MPI application programmer, but on the other hand supports the exploitation of special abilities of single meta computing resources by providing a new MPI communicator which reflects the real topology of the connected systems.

Keywords: *Grid Computing, Meta Computing, Cluster, MPI, Meta Computer Configuration*

1. Introduction

The use of Clusters of PC for high performance computing tasks has become very popular recently, because they provide a high performance to a much lower price than dedicated multicomputers. New networking techniques have been developed in the past few years to increase performance and to combine the relatively cheap high processor performance with adequate networking performance. The short-time availability of standard PC hardware components of a special type leads to the problem of keeping extensions of existing cluster installations homogeneously in respect to the PC hardware of the single nodes. In addition to this, it is often not possible to keep the network hardware and topology, e.g. because a new network is used or the cluster extensions have to be installed in a remote location. This paper presents the MetaMPICH project, which provides a MPI [12] platform for *meta computers*, as a solution to many of these problems.

Though MetaMPICH is also suitable for large Massively Parallel Systems and dedicated multicomputers as shown in the Gigabit Testbed West project [3], where a predecessor of MetaMPICH has been used to run coupled simulations on WAN-connected supercomputers, this paper focuses on PC-clusters with high numbers of processors because these have become more and more popular in the past years. These systems confront the middleware developer with special problems generated by the lack of a single system image. The resulting difficulties in resource management and process creation are covered in section 4. A lot of research and development has been done to solve these problems for dedicated NORMA-systems[1] in the past years, for example in the Intel Paragon Project [14], [13], [15]. The Paragon software aimed at providing a complete single system image by the operating system layer, which made it difficult to transfer this technology to other hardware and software platforms. Other efforts were made to emulate expensive supercomputers on much cheaper networks of workstations to decrease the costs of software development and testing [16]. With the development of the MetaMPICH library we tried to provide transparent communication using a user-level software, which runs on a variety of hardware platforms and multi-purpose operating systems including Linux and Windows XP.

Other projects like MPICH-G [4] (a MPI implementation for Globus) and PacX [5] made it possible to run MPI applications on distributed Massively Parallel Systems, but relied on existing internet connections, i.e. the normal existing network routes between the systems were used. In addition to this, our goal was to make it possible to use multiple dedicated network connections and to support memory-coupled clusters.

Traditional beowulf-like clusters use a standard network for communication, in most cases this is fast or Gigabit Ethernet. In these environments it is easy to extend an

[1] NORMA = **N**o **R**emote **M**emory **A**ccess

In: International Parallel and Distributed Processing Symposium (IPDPS 2003), Workshop for Communication Architecture in Clusters (CAC 03), Nice, France, April 2003

installation because the used IP-based networks can easily grow with the number of nodes and are able to bridge longer distances to other parts of the cluster in remote facilities. The performance of the network in such systems is usually low, and thus only parallel applications on such systems do not have to communicate very much at all. Because of this, the scalability of the network is not very important.

Other clusters that diverge from this COTS[2] paradigm by using particular networking hardware, which is specially developed for parallel cluster computing like Myrinet and Scalable Coherent Interface (SCI). These are much more sensitive to inhomogeneous extension because the applications they are running can communicate much more in comparison to local computing work.

Applications which scale well on the metacomputing platforms described in this paper are therefore those who don't communicate very much at all. This implies they never benefit from high speed cluster interconnects. But even non-trivial parallel applications with much more communication can make use of this hardware platforms if they take care of the communication topology. As an example, one could distribute a simulation application, which consists of two or more different loosely coupled parallel applications, on different cluster systems. Another possibility is to partition application data in a manner, which minimizes the communication between groups of partitions which then are assigned to different clusters. In this way, internal high speed cluster interconnects could be exploited without the inter-cluster connection causing a serious bottleneck in communication.

This article is structured as follows: In section 2 we give a short description of the hardware architectures that we used for our work and how these systems are connected. We explain our techniques to describe and configure meta computing systems, which are used in section 3, where the MetaMPICH library is presented. Another important topic in this scope is covered in section 4: process management in distributed inhomogeneous computing environments. Section 5 shows some benchmarking results measured with MetaMPICH. The future work which has still to be done is described in the last section in addition to a short summary of this paper.

2. Cluster-based Meta Computing Architectures

Meta computing resources are existing hard- and software installations which are to be combined to a higher level construct which is widely named as *meta computer* or *grid computer*. Because these systems were not installed with the intention to be used together as one big system, a broad variety of hard- and software and even of system architectures is found. Figure 1 shows the processor related view of a sample meta computer configuration consisting of a cluster (meta host A) and a SMP-system (meta host B). Unlike in traditional massively parallel systems, the processors are not replaceable by each other, instead they have to be grouped into a number of levels within which they are exchangeable. The figure shows four level 1 groups in meta host A, which could be dual processor SMP systems. Their grouping in a level 2 processor group could be done by a system area network (SAN) like Scalable Coherent Interface [7], which is supported by MetaMPICH through the SCI-MPICH extension [17] based on the SMI-library[3] developed at the chair for operating systems ([1],[10],[2]). Each SMP node represents a SCI node in a level 2 group. These nodes are partially exchangeable if the nodes run the same operating system and provide the same hardware platform. Only processes that need special networking hardware which is not installed in all nodes have to be executed on the special I/O-node, which is a task for the meta computer configuration described in section 2.1. The network interfaces can be attached to different process group levels, e.g. while meta host A has only interfaces at multiple level 1 groups, all processors of meta host B share the networking facilities at level 2.

The figure also defines the term *meta host* which is often used in this paper: a meta host is a component of a meta computer which is seen as one system from the view of the meta computer.

Our installation at the chair for operating systems consists of a quad Xeon(550MHz) server, a 8-node cluster with dual Pentium-III (800MHz) and a 4-node-Cluster with dual Pentium-II processors. For the testing of processor-heterogeneous configurations we use a SUN Sparc Enterprise 450.

For networking we use the above mentioned SCI, which we use at the same processor level as shared memory. To connect the meta hosts to each other fast and Gigabit Ethernet as well as multiple ATM 155Mbps adapters, which can be used for ATM WAN connections, are installed.

2.1. Meta Computer Configuration

In contrast to SMP-Systems, a Cluster of PC has no single process space, no shared memory and in particular no shared access to networking facilities. Therefore in opposite to SMP-machines it is not possible to just fork a number of symmetrical processes which can all take the same part in the meta computing application. The processors of

[2]COTS - components off the shelf: This means only standard hard- and software components designed for the mass marked

[3]The **S**hared **M**emory **I**nterface is a shared memory communication library which utilizes the SCI network.

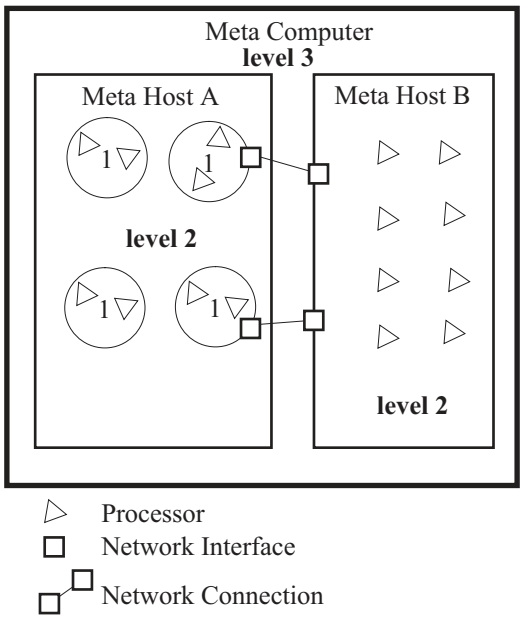

Figure 1. Processor Groups in a Sample Meta Computer Configuration

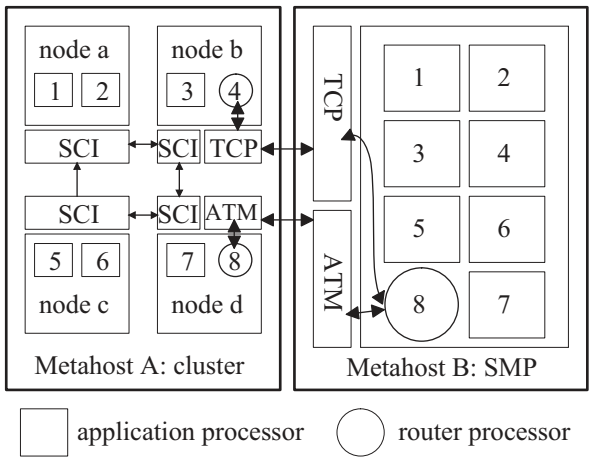

Figure 2. A more concrete Meta Computer Configuration

```
METAHOST metahost_A {
   TYPE=ch_smi;
   NODES= node_a,
          node_b (192.168.0.1),
          node_c,
          node_d (ATM_PVC 0.0.42);
}
```

Figure 3. Definition of a Meta Host

a meta computer can be grouped in two or more levels as shown in figure 1.

To use a cluster, the routing processes (see 3.1) which need access to the network interfaces, have to be initiated on the appropriate nodes of the meta host.

2.2. A Description Language

To allow the user of a meta computer a flexible way to configure his desired computing platform, we developed a simple description language. The MetaMPICH library, which is described in section 3, makes use of this language, both to manage the needed processes and to inform the processes about their part in the meta computing application.

Figure 2 extends the example from figure 1 to a real meta computing platform. Meta host A would be declared as shown in figure 3.

The nodes of a meta host built by a cluster of PC must be enumerated together with the network addresses which can be used to connect to other meta hosts, the native ADI[4] device must also be specified. IP networking is supported but also the AAL-5 layer provided by ATM protocol stacks, in this example an ATM PVC connection is used. In addition it is possible to define the quality of service parameters (QoS) for an ATM connection in the configuration. The advantage of a QoS supporting network is obvious, because one gets a stable networking environment with the desired properties. Using public IP networks like the internet or intranets to couple meta hosts will lead to unpredictable results caused by network traffic of other applications and changing network routes. An ATM connection allocates the resources for the specified QoS parameters at initialization time and guarantees them for the life-time of the connection.

Furthermore, the meta host definition may contain other parameters like environment variables for the processes and the path to the executable. Maximum numbers of application processes per node can also be given to disburden nodes which are supposed to do the external communication for the meta host. For large cluster installations with a big number of nodes, these can be enumerated with node boundaries. Precondition is that the node names follow the pattern `<prefix><number>`, where `<number>` is a serial node number and `<prefix>` is a common name prefix. `p4-00 - p4-63` for example would denominate 64 nodes from the cluster `p4`.

Another important part of the meta computer description is the connection configuration. The example in figure 2 has only two meta hosts connected by one IP connection and one ATM connection. To describe these, each pair of

[4] Abstract Device Interface - software interface for the networking device drivers used in the MPICH-library

meta hosts is declared separately with both connections as figure 4 shows.

```
PAIR metahost_A metahost_B 2 -
192.168.0.1 -> 192.168.0.2
ATM_PVC 0.0.42 -> ATM_PVC 0.0.42

PAIR metahost_B metahost_A 1 -
192.168.0.2 -> 192.168.0.1 \
   ATM_PVC 0.0.42 -> ATM_PVC 0.0.42
```

Figure 4. Definition of the Connections between two Meta Hosts

In this example, we have a two-to-one router configuration, therefore the first PAIR section has two router definitions. In the inverted order only one router is specified, which is made possible by the SMP architecture of meta host B. The first meta host must have two router processes because the network interfaces used are installed on two different cluster nodes.

The complexity of the configuration grows with $O(m^2)$, where m is the number of meta hosts, because the number of host-to-host connection declarations equals to $m * (m - 1)$. More information about the meta computer configuration can be found in the MetaMPICH manual [11].

3. MetaMPICH: A Meta Computing MPI-Implementation

Based on the portable MPI implementation MPICH from Argonne National Labs [6], we developed a metacomputing extension for this software library. The goal was to keep the portability of the MPICH-library and use the variety of networking devices for our purposes, which leads to an architecture that uses two new pseudo devices for the meta computing functionality. These new devices are ADI-2 devices which fit in the communication architecture of MPICH. This clearly layered communication model of MPICH, next to its freely available source code, was the strongest reason to use it for our extension. To make an existing ADI-2 device work with our extension, only a few changes have to be made to its source code. This makes MetaMPICH easy to port and to extend.

MetaMPICH is part of the MP-MPICH[5] project, which also includes SCI-MPICH for Scalable Coherent Interface support.

3.1. Router Processes for Message Transport

The meta computers covered by this paper do not need to provide any all-to-all networking facilities, i.e. nodes may exist which are not able to communicate directly with their primary network interface. Therefore MetaMPICH needs special communication nodes which route outgoing messages from a meta host to the destination meta host. The communication is done by routing processes, which have to be run at application startup. As shown in figure 5, a router process drives connections to one other meta host. To do this we can have one-to-many connections (but all to the same meta host) and multiple connections between two routers. Each router has two parts of communication: first the native network, which connects the local meta host processors, and second the external network interface, which bridges the distance to the remote meta host.

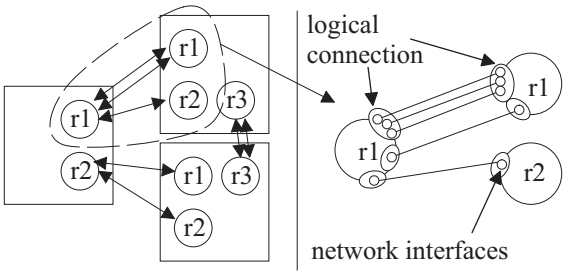

Figure 5. Point-to-point Router Connections between three Meta Hosts

To optimize the use of the networking resources, router-to-router connections can also bundle network interfaces to one logical connection. Large messages are split then into smaller pieces which can be transferred at a larger bandwidth and lower latency.

Figure 6 shows the architecture of the multithreaded router process. It has two communication directions: vertically it receives messages from the local meta host via `MPI_Receive` calls and forwards them horizontally to the appropriate remote router process, which then *tunnels* the message in the remote meta host to the target process. Implemented by now is the use of the TCP protocol and the AAL5 protocol with ATM adapters. As described in section 2.1 it is possible to use QoS parameters which provide network connection with stable properties and performance even for long-distance connections over public ATM networks.

3.2. Meta Computing Devices

The architectural design of MetaMPICH uses three ADI-devices. This solution was chosen to separate the meta computing functionality from the native device, which makes the adaption of new ADI-devices easy because only marginal changes have to be made to the source code of the device. Figure 7 shows the splitted communication layers

[5] `http://www.lfbs.rwth-aachen.de/mp-mpich`

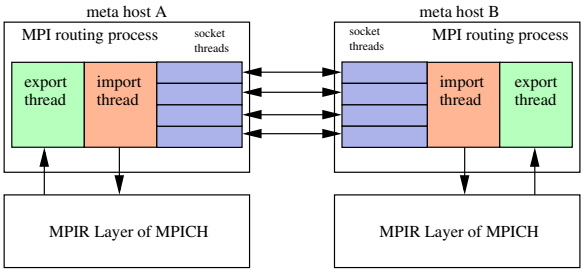

Figure 6. A layered view of horizontal and vertical communication of the MetaMPICH routers

for the meta computing devices: the meta computing functionality is achieved through the ch_gateway and ch_tunnel devices, which allow communication with the router processes we described in section 3.1. They forward messages from the local meta host to another by using other networking facilities than the local application processes. The two new devices are pseudo devices in that they need a native device (a *real* device) to communicate. Their own task is to forward messages to the router processes (gateway) and distribute them into the local meta host (tunnel), pretending to be the real sending process on the remote meta host. This way, the native ADI-device which receives this message on the local meta host does not know that the message came from a remote meta host. Since the meta computing devices use standard calls of the native device, they are device-independent.

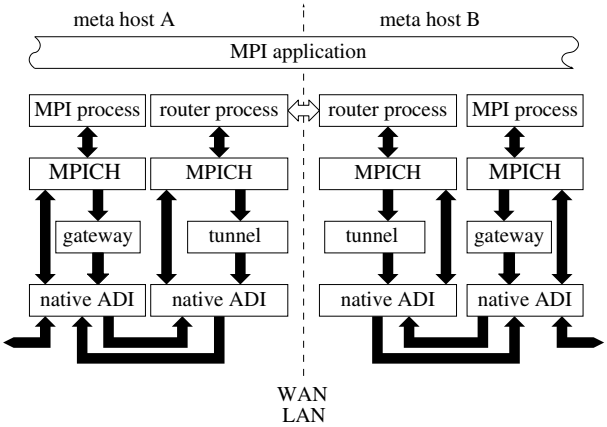

Figure 7. Multi Device Architecture of MetaMPICH

Messages with senders and destinations on the local meta host are sent using the default communication method which is provided by the native device functions. Supported so far are the devices ch_shmem and ch_smi, which is the shared memory device for Scalable Coherent Interface adapters. Therefore the target platforms are symmetrical multi-processor systems and SCI-coupled clusters.

3.3. Exploitation of Meta Host Specific Hardware

MetaMPICH hides the distributed topology of the configured meta computer completely from the message passing application. But although every MPI program runs out of the box, it is clear that the router connections will form communication bottlenecks because of their much lower bandwidth and higher message latencies. So only applications with low communication per calculation relation will scale well on a meta computer consisting of coupled clusters. To use more communication intensive applications on this kind of parallel computing platform, the MPI programmer must take care of the bottlenecks. MetaMPICH supports such efforts in providing a special new communicator, which groups the process groups described in section 2. Figure 8 shows a sample set of application processes, where the subsets located on different meta hosts are available in the local communicator MPI_COMM_LOCAL, all application processes are in MPI_COMM_WORLD as one would expect.

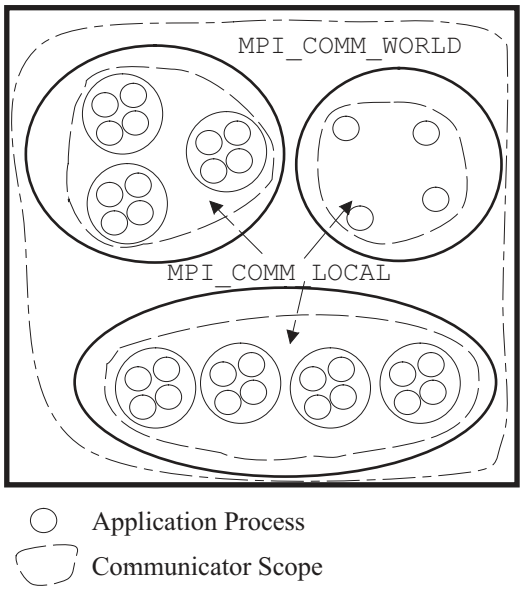

Figure 8. Example with three Meta Hosts: The new Communicator provided by MetaMPICH

4. Process Management

As described in chapter 2.1, it is not sufficient to start similar processes on the nodes of a Meta Computer in order to run a parallel application. The distribution of the processes depends on one hand on the hardware structure of the meta computer and on the other hand on the operating system running on the different nodes as shown in this chapter.

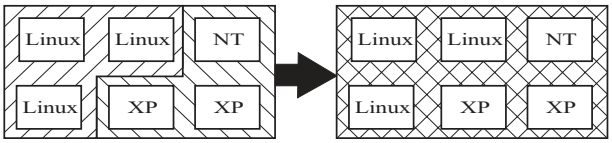

Figure 9. Merging two Homogeneous Clusters to one Heterogeneous

4.1. Process Distribution on a Meta Computer

To run a MPI application on a meta computer consisting of several meta hosts, which might themselves be clusters of computers, it is not sufficient to merely specify the number of application processes that have to be run on each meta host. As shown in section 3.1 there have to be two types of processes, router processes and application processes. The distribution of the different processes depends on the user-specified meta computer configuration (see section 2.1). In the example from figure 2, the two router processes on meta host A must run on the two dedicated nodes node_b and node_d. Therefore, it must be possible for the mpirun command to place the router processes exactly on the nodes where they are needed.

To make good use of the hardware, it must also be possible to specify the number of application processes on each node. In the given example, the nodes of meta host A have two processors each. Because the router processes are running on two of them, it is a good idea to run only one application process on node_b and node_d. The single node forming meta host B has 8 processors which can all access all network interfaces - so no process has to be started on a dedicated processor. In some cases it is profitable to reserve more than one processor for a routing process to take an advantage of its multithreaded architecture as it is described in section 3.1

To provide a simple `mpirun` command to the user, which needs only the configuration description and the program name, the process creation procedure is split into the process groups on the metahosts. For the given example the user would commit the command `mpirun -meta mymetacomp.cfg progname` which causes the execution of two sub-mpirun calls. One sub-call is made for the SCI Cluster on the frontend node of meta host A (`mpirun.smi -np 8 -nodes node_b,node_d,node_a,node_c -metarun mymetacomp.cfg progname`) and one for the SMP system (`mpirun.shmem -np 8 -metarun mymetacomp.cfg progname`). The mpirun command parses the configuration to run the commands of the next stage, the running processes themselves have to parse it again to determine their part in the meta computing application. The routing processes then ”know” they have to execute the router code instead of the application code.

4.2. Process Creation in Inhomogeneous Operating Systems Environments

The MPI-processes located on one meta host use the same communication device, e.g. SCI, but different meta hosts can use different communication devices. In that case it is impossible to use the same commandline parameters for each meta host of one meta computer. Each communication device needs special commandline parameters as described in the MP-MPICH manual [8]. Furthermore, to use different communication devices, different executables respectively dynamic libraries are required. To use the appropriate files and parameters, `mpirun` has to analyze the meta configuration file.

When starting a MPI-application on a meta computer, it should be possible to use machines with different operating systems. Meta computers can not only be a combination of meta hosts with different operating systems, but also each meta host can be a heterogeneous cluster consisting of machines with different operating systems - this is especially true in environments where existing desktop workstations are put together to use them for cluster computing. This demand effects that the process initiation on one meta host also is inhomogeneous. Depending on the operating systems the executable files and commandline parameters differ on the nodes of one meta host.

The creation of MPI-processes in environments with multiple operating systems is laborious. Although it is possible to start remote processes by using rsh directly this is not a comfortable solution, because it is necessary to start each process manually. The creation of many processes should not be more complicated than the start of a single process. To simplify the process creation on a meta computer special remote executions tool are necessary.

4.3. Remote Execution Tools

The MP-MPICH project provides startup tools for homogeneous Windows- and Linux/Unix-clusters, but these tools are not yet suited for process creation in inhomogeneous

clusters. For initiation of MPI-processes on one homogeneous meta host, the command-line tools `mpirun` respectively `mpiexec` can be used.

The call of `mpirun` is similar on Unix/Linux and Windows-systems, however the effect of the call is different on the varying platforms. When calling `mpirun` on a Unix node rsh is used to start remote processes, while the remote processes in a Windows environment are started by using a special RPC-service. NT-MPICH offers the Windows-service *rcluma* for remote execution, which uses Windows-RPC and provides several features for remote execution.

As a workaround for remote execution on heterogeneous meta hosts, it is possible to call `mpirun` for each operating system of this meta host on an appropriate node. Thus, manual analysis of the meta configuration file and comparison with the arrangement of operating systems on the nodes is necessary.

For comfortable cluster management, the exclusive use of `mpirun` is not adequate, therefore a graphical frontend for management of heterogeneous clusters is necessary.

5. Performance of the Router Connections

As we would expect, meta computing environments are not suitable for parallel applications with processes which communicate and synchronize very often all-to-all. The slow inter-meta-host connections (in comparison to todays fast cluster interconnects) are bottlenecks to the communication performance. Only applications which take care of the topology with the use of meta-communicators described in section 3.3 will scale well. These application must diverge from the classical SIMD-scheme and execute application code on the meta hosts which communicates locally much more than to other meta hosts. For example a good idea would be to split a simulation containing two weakly coupled computing models on two meta hosts.

To show some basic performance data, we present two point-to-point communication experiments done with the PingPong test of the Pallas MPI benchmark. Fast and Gigabit Ethernet and ATM at 155 Mbps connections were used directly over one switch as meta-host interconnect and SCI was used as cluster interconnect.

Figure 10 shows the measured round-trip communication latencies for several message sizes. As one would expect, Gigabit Ethernet does not perform best at small message sizes but gets an advantage with large messages because of the high bandwidth. ATM performs nearly equal for small messages while Fast Ethernet has an advantage at sizes below 512 byte.

The bandwidth chart in figure 11 shows that ATM performs slightly better than Fast and Gigabit Ethernet for message sizes below 1024 byte. Although the ATM adapters we used only support 155 MBit/s, they provide a bandwidth

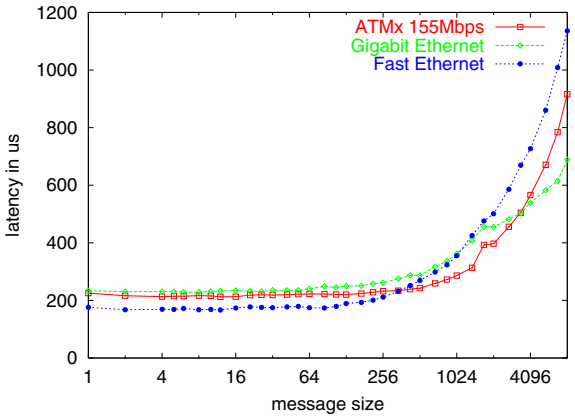

Figure 10. Point-to-Point Latencies with different Router Interconnects

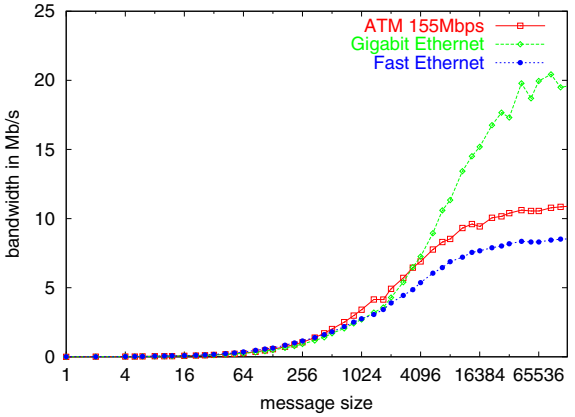

Figure 11. Point-to-Point Bandwidth with different Router Interconnects

similar to GE for smaller message sizes. This effect is conditional on the small packet switching technology of ATM, which transports messages in portions of 48 Bytes. In addition, the ATM adapters used (ForeRunner PCA-200EPC) have a dedicated on-board processor for data processing, which disburdens the system processor. The bandwidth saturation is 11 MByte/s for ATM, 9 MByte/s for Fast Ethernet and 23 MByte/s for Gigabit Ethernet. For Gigabit Ethernet this is a bad result which indicates a problem in the router implementation, which seems to be caused by the thread scheduling in the multithreaded router. The bandwidth of the SCI network is 80 MByte/s, which would be high enough to use the full capacity of the Gigabit Ethernet connection. The latency between application process and router using the SCI network is in the range of $8 - 12\mu s$

and has therefore low influence on the measured latencies.

Both experiments show that the router architecture of MetaMPICH exploits the used networking techniques very good. Especially the latencies for MPI messages are not much higher than the normal latencies of the protocol stacks which have to be used. Thus the overhead of the message forwarding done by the router processes is low.

6. Conclusions and Outlook

In this paper we presented a solution to run parallel applications using the Message Passing Interface on a cluster of clusters system. We described a way to configure this type of *meta computers* with a description language which offers the possiblity to specify multiple dedicated network connections between the parts of the meta computer. The networking facilities of multiple cluster nodes can be used to increase the performance. To use Wide Area Network technologies more efficiently, the utilization of the AAL5 layer of the ATM protocol has been implemented which supports quality of service and avoids the overhead of the IP protocol. We showed that the MetaMPICH architecture gains nearly the maximum performance out of the meta computer connections, with exception of Gigabit Ethernet. This problem has to be investigated and solved. Future work will now focus on suitable meta computing applications to prove the usability of the described computing platforms.

Much work has still to be done in the area of process management on clusters. To provide a comfortable solution to start processes on meta computers, a new startup-mechanism for inhomogeneous clusters has to be implemented. The new remote execution tools, including `mpirun`, must be independent of platforms and operating systems, especially user account management will get separated from the operating systems mechanism. This will avoid the need for normal user accounts on external computing resources, only a meta computing account is needed then which will never allow direct login.

Important will be the choice of the underlying communication mechanism. Unix RPC and Microsoft RPC will not be used because they are not compatible due to authentication issues. As a platform independent solution, JAVA RMI is very attractive but the needed running virtual machine on each system has the disadvantage of a large memory consumption, in addition many of the needed functions will have to be implemented with JAVA native interface.

Due to these difficulties, we decided to choose another middleware for the process communication, which will be a real-time CORBA implementation developed at our institute [9].

References

[1] M. Dormanns. *Parallelisierung gitterbasierter Algorithmen auf NUMA Verbundsystemen mit gemeinsamem Speicher*. PhD thesis, RWTH Aachen, 1999.

[2] M. Dormanns, K. Scholtyssik, and T. Bemmerl. A Shared-Memory Programming Interface for SCI Clusters. In H. Hellwagner and A. R. (eds.), editors, *SCI: Scalable Coherent Interface*, pages 281–290. Springer Verlag, 1999.

[3] Eickermann, Völpel, Wunderling. Gigabit Testbed West Abschlussbericht. Technical report, Forschungszentrum Jülich, March 2000.

[4] I. Foster and N. Karonis. A grid-enabled mpi: Message passing in heterogeneous distributed computing systems. In *Proceedings 1998 SC Conference*, 1998.

[5] E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed Computing in a Heterogeneous Computing Environment. In *Proceedings PVM/MPI 1998*, pages 180–187, 1998.

[6] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, Sept. 1996.

[7] IEEE. ANSI/IEEE Std. 1596-1992, Scalable Coherent Interface (SCI). Technical report, IEEE, 1992.

[8] J. Worringen and K. Scholtyssik. *MP-MPICH, user documentation & technical notes*. Lehrstuhl für Betriebssysteme, RWTH Aachen, 2002.

[9] S. Lankes, M. Pfeiffer, and T. Bemmerl. Design and Implementation of a SCI-based Real-Time CORBA. In *4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001)*, Magdeburg, Germany, May 2001.

[10] M. Dormanns. Shared-Memory Parallelization of the GROMOS96 Molecular Dynamics Code. In H. Hellwagner and A. R. (eds.), editors, *SCI: Scalable Coherent Interface*. Springer Verlag, 1999.

[11] M. Pöppe and J. Worringen. *Meta-MPICH, user documentation & technical notes*. Lehrstuhl für Betriebssysteme, RWTH Aachen, 2002.

[12] MPI Forum. MPI: A message-passing interface standard. *International Journal of Supercomputing Applications*, 1994.

[13] Rüdiger Esser, Renate Knecht. Intel Paragon XP/S - Architecture and Software Environment. In *Supercomputer '93 - Anwendungen, Architekuturen, Trends*, pages 121–141, Mannheim, Juni 1993.

[14] Scalable Systems Division Intel Corporation. *Intel Paragon Supercomputers*. 1993.

[15] Stephan Zeisset, Stefan Tritschner, Martin Mairandres. A new approach to distributed memory management in the mach microkernel. In *USENIX Annual Technical Conference*, San Diego, January 1996.

[16] T. Bemmerl and B. Ries. Programming tools for distributed multiprocessor computing environments. *International Journal of High Speed Computing, Vol.5 No.4*, pages 595–615, January 1992.

[17] J. Worringen. SCI-MPICH: The Second Generation. In *SCI Europe 2000*, pages 10–20, 2000.