Heterogeneous Computing With MPICH/Madeleine and PACX MPI: a Critical Comparison

Daniel Balkanski, Mario Trams, Wolfgang Rehm

{danib,mtr,rehm}@informatik.tu-chemnitz.de

Technische Universität Chemnitz
Fakultät für Informatik*
Straße der Nationen 62, 09111 Chemnitz

November 27, 2003

Abstract

Today, computational Clusters of Networked Workstations (usually off-the-shelf PCs interconnected by high-speed, low-latency communication networks) are playing a major role in redefining the concept of supercomputing. Sadly, the fierce competition and the lack of common industry standards have lead to wide spread of clusters interconnected with incompatible high-performance System Area Networks (Gigabit Ethernet, GigaNet, Myrinet, SCI, QNet, etc.).

With the growing interest in grid technologies, aggregating such clusters into a bigger heterogeneous Clusters of Clusters (CoC) is becoming a 'hot' issue. Lot of work was done by the cluster community in past few years for developing message passing middleware that effectively supports such configurations, but unfortunately most of this projects are incomplete and/or didn't show signs of any further development. Being interested mainly in solutions providing the programmer with some single, portable and widely accepted parallel programming model like the Message Passing Interface (MPI), we have narrowed down our choice to a few alternatives: PACX-MPI, MPICH-Madeleine III and MPICH-G2.

Besides a brief overview of these three alternatives the paper presents our experiences and benchmark results by using two of them: PACX-MPI and MPICH/Madeleine III in the the heterogeneous CoC setup consisting of two sub-clusters equipped with Myrinet, SCI, and Gigabit Ethernet.

1 Introduction

Huge advances of two modern age technologies - personal computers and high-speed networks have allowed to break-down the price barriers and construct cost effective clusters of PCs which provide comparable performance to super-computers at a fraction of the cost.

*The work presented in this paper is funded within the framework for technology promotion by means of the European Regional Development Fund (ERDF) 2000-2006 as well as by means of the Freistaat Sachsen, Staatsministerium für Wissenschaft und Kunst (SMWK Project-Number 7455/1180).

Paraphrasing the indestructible and ever-working Moore's law, computer's processor power doubles every 18 months. But with this rapid increase of computational power comes the problem that the bottleneck in such cluster system shifts from the nodes to the actual networks which connect them. The industry responded to the problem by developing a wide variety of high-performance interconnection technologies with different price and performance characteristics. Today users have the freedom of choosing the most appropriate for their applications and budget interconnection technology, but it faces them with interoperability problems when clusters with incompatible SANs have to be interconnected into a bigger Cluster of Clusters.

This situation often arises when the users want to expand their clusters and discover that another networking technology better suits their application requirements or budgetary constrains. But sometimes even if the users decide to stay with the same type of interconnection technology they discover that the current generation of network interfaces cannot be connected to the existing networks of older generation or if it's possible it would degrade the performance to an unacceptable level. This is due to the changed mechanical, electrical and protocol specifications of the network interfaces (optical vs. copper media, higher signal clocking, introduction of new packet types, protocol changes, etc.)

The most straightforward way to circumvent the problem with interoperability of the inter-cluster SANs is to introduce one or more gateway nodes at each sub-cluster. Beside the respective intra-cluster SAN's interfaces, these gateway nodes have to be equipped with additional network interfaces connecting them to one or more inter-cluster networks. To avoid becoming a bottleneck, these inter-cluster networks must deliver same or higher network performance than the fastest intra-cluster network. From point of view of minimization of the latencies between the nodes participating in the different sub-clusters, optimal configurations are these with the inter-cluster networks merged with the intra-cluster SANs, or in other words, with gateway nodes directly connected to the SANs of the sub-clusters.

Unfortunately, running distributed message-passing applications on such heterogeneous CoC configurations is still problematic, because these applications have to deal with multiple communication interfaces, low-level protocols, data encodings, data compressions and quality of service choices in order to achieve acceptable performance. One possible approach for developing such applications is the ad hoc method, which is based simply on reuse of multiple, readily available program components, each specialized in different low-level communication protocol. While effective, this approach is tedious, error prone, and leads to non-portable applications intricately bound to the specific configuration on which they were initially designed.

A highly desirable approach would be to supply the programmers with some single, wide accepted parallel programming model, like the Message Passing Interface (MPI) [13], [14] specially designed to support such heterogeneous CoC configurations. This MPI implementation must provide mechanisms that allow the methods used for each communication calls to be determined independently of the application code on the basis of the underlying communication structure. However, the development of such multi-protocol message passing middleware requires solving many challenging problems like: reliable ordered message delivery on top of multiple communication methods, effective message packetization strategies, protocol conversions and addressing issues in incompatible networks, application launch and data distribution across the sub-clusters, runtime determination of the routing policies, routing load balancing in a case of multi-gateway environment, effective topology-aware implementation of collective operations, scalability, manageability, fault tolerance, etc.

Given the volume and complexity of these challenges, it becomes clear why most of the projects targeting development of communication middleware systems for heterogeneous CoC are incomplete or dead. Hence, because in this study we consider only the free open-source software solutions available, we have to restrict our attention to only a few usable possibilities described briefly in the following section.

2 Overview of available communication middleware suitable for building CoC

2.1 PACX-MPI

PACX-MPI (PArallel Computer eXtension) [5], [8], [2] has been primarily designed to enable running MPI applications across several MPP or PVP supercomputers, without having to introduce changes to the source code of this applications and by fully exploiting the communication subsystem of each machine.

To achieve this goal, PACX-MPI is designed as a library siting between the user application and the local intra-machine MPI implementation. When the application calls a MPI function, the call is intercepted by PACX-MPI and decision is made whether there is a need to contact another MPP (or sub-cluster in our case) during the call execution. If not, the library sends the message using the matching MPI call from the underlying local MPI library. This way, the highly tuned vendor's MPI implementations are used for all intra-machine communications.

When the MPI call involves another MPP the communication is forwarded via network by using TCP/IP sockets, but in this act MPI processes do not exchange messages directly. Instead, on each parallel system two special nodes are reserved, one for every (incoming and outgoing) communication direction. On each of these nodes a daemon MPI process is running, which takes care of communication with the local nodes, compression and decompression of data for remote communication and communication with the peer daemons of other parallel machines. This daemon approach bundles the communication and eliminates the need to open connections between each process on every system, which saves resources and permits to handle security issues centrally.

A weak point of this design is that the use of wide-area, heavy-weight protocol stacks like TCP/IP for inter-cluster communication introduces significant latency overheads. The optimal, single-hop route configurations for interconnecting a sub-clusters by providing a gateway nodes equipped with interfaces to the both types incompatible intra-cluster SANs are not supported. Instead the messages exchanged between the sub-clusters have additionally to traverse twice the full protocol stacks of the wide-area protocols used for connection between the daemon processes of each sub-cluster. This results in significant difference between the latencies of the intra- and inter-cluster messages, which becomes bigger when low-latencies, and/or OS-bypassing network interfaces are used for the intra-cluster communication.

Another problem is that direct TCP connections are needed between all nodes on which forwarding daemon processes are running, which is not the case for the Beowulf-like clusters, in which all nodes reside in a separate, insulated from the rest of the world private network address space. This problem can be circumvented by putting the nodes of all clusters to be connected in common physical or virtual network and rearranging all conflicting hostnames and addresses. Better solution would be if the forwarding daemon processes can be 'fixated' to spawn always on the front-end nodes of the sub-clusters which normally have second fixed routable Internet address. But this would be appropriate only to the relatively more rarely used cluster setups, where the frontend have connection to the high-speed intra-cluster SAN.

PACX-MPI doesn't provide convenient application launch on machines where each node has different network address which is exactly the case for the Beowulf-like clusters. The most appropriate startup method for PACX-MPI applications on CoC configurations is so called "server startup". For this method a special server computer with a fixed network address is required to gather and exchange the information from the different partitions/clients for the network addresses of the nodes on which the forwarding daemon processes are running. The main inconvenience come from the fact that after launching own instance of startup server the user have to launch the applications on every sub-cluster manually, typically with the means provided by the sub-cluster vendor MPI, for instance `mpirun`. That's why interfacing an PACX-MPI CoC with batch system would be quite difficult.

We want to mention here that to solve this application startup inconveniences, a tool called the "Configuration Manager" is currently under development in the frame of the DAMIEN project [15]. The primary goal of this Configuration Manager is to ease the handling of resources and applications for PACX-MPI jobs.

2.2 MPICH/Madeleine III

With it's last third major revision MPICH/Madeleine becomes one of the most promising MPI implementations for CoC, because it is specially designed to support natively setups of multiple heterogeneous networks.

MPICH/Madeleine, like many other MPIs for high-performance low-latency intra-cluster SANs is actually a port of MPICH [10]. MPICH is probably the most widely used, free realization of MPI, which design goal was to combine portability with high performance within a single implementation. To achieve these goals MPICH employs multi-layered architecture, defining an intermediate interface called Abstract Device Interface (ADI) [9], which allows to plug modules (a.k.a. ADI devices) to support different communication protocols. While it is theoretically possible in this way to support network heterogeneity in MPICH, it is quite complicated and requires a rather heavy integration work to be done each time when a new device is to be added in order to preserve inter-device co-existence. Therefore the authors choose an alternative approach to add heterogeneous support in MPICH by interfacing it with the already available multi-protocol communication library called Madeleine [4], [3]. This approach makes possible to reuse readily available software components and prevents feature modifications in the MPICH's ADI code to cause incompatibilities, although, maybe, it is not the best performance wise solution.

Madeleine III, which is a part of the PM^2 [18] programming environment, is a native multi-device communication library, which transparently supports most of the important intra-cluster SAN protocols like TCP, VIA, GM, BIP, SISCI, MPI, PVM, SBP. While providing a high-level of abstraction the Madeleine III communication library is able to effectively exploit some low-level characteristics of the underlying network devices (like preallocated buffers, DMA operations, etc.). In addition it has a builtin efficient inter-device forwarding functionality [17].

Because Madeleine III can natively forward messages between all maintained protocols, it fully supports the optimal, single-hop route configurations, and don't need to use intermediate WAN protocols. Furthermore, due to the careful design of the routing functionality double traversal of the data through the full protocol stacks on the gateway nodes is avoided, which minimizes the latency of forwarded messages.

To enable the routing functionalities of Madeleine, merging this way several sub-clusters into a single CoC, one must supply two configuration files. First file, which we would call network configuration file contains the description of the topology of all physical networks available inside and between the sub-clusters. Second file, which we would call channel configuration file contains description of the physical channels, which simply overlap a given physical network or only a segment of it. In addition it can also contain one or more virtual channels which are built out of many physical channels, this way creating a virtual communication channels encompassing different physical heterogeneous networks. Applications can use for communication both physical and virtual channels. The virtual channels are slower than the physical one and they should be used only by applications that are running over the nodes which do not have direct physical network connection in between. In this case Madeleine automatically chooses the best physical network available to route the data.

Unfortunately this method of topology description is quite inflexible and restricts the use of the quite powerful routing functionalities of Madeleine. The problem comes from the fact that in the configuration files networks and channels topologies can be described only by using hostnames, and there is no distinguishing between actual hosts, hostnames and network interfaces. This makes impossible to describe some more complex topologies, especially in case when some of the hosts posses several kernel-resident network interfaces.

An other problem (or rather, inconvenience) is that to launch a MPI application one should use a special loader called Leonie instead the familiar `mpirun` command. During the startup MPI applications are supplied only with a default communication channel but not with something like machine-file. It is simply assumed that instances of the application has to be spawned on all hosts that participate on this default channel. This forces the user to deal with lots of configuration files when they want to change often the number of the hosts involved in the computation, while in practice using always the same channel and restricts them from possibility to unload the gateway nodes from computation for performance reasons.

2.3 MPICH-G2

MPICH-G2 [7], [12] is a grid-enabled MPI implementation, which is also a port of MPICH, built on top of services provided from the Globus Toolkit®. The Globus Toolkit is a collection of software components designed to support development of applications for high-performance distributed computing environments, or “Grids”. The services provided by this toolkit help MPICH-G2 to support efficient transparent execution in these heterogeneous environments, while providing application-level management of heterogeneity.

Such complex issues concerning the application startup, typical for wide-area multi-site Grid environments, like cross-site authentication, the need to deal with multiple schedulers with different characteristics, coordinated process creation, heterogeneous communication structures, executable staging, collation of the standard output are successfully hidden.

By the use of few commands additionally to the the familiar `mpirun` command user gains access to powerful features of Globus Toolkit like *Monitoring and Discovery Service (MDS)* for selective search of computational resources, *Grid Security Infrastructure (GSI)* for obtaining a (public key) proxy credential used to transparently authenticate it to each site, *Global Access to Secondary Storage* to stage executable(s) from remote locations (indicated by URLs), and finally *Dynamically-Updated Request Online Coallocator (DUROC)* in conjunction with *Grid Resource Allocation and Management (GRAM)* to start and subsequently manage the application instances on every site (possibly in assistance with the corresponding batch system).

Once the MPICH-G2 application is started the most efficient communication method possible between two processes is automatically used, selecting vendor supplied MPI if available, or *Globus Communication (Globus IO)* with *Globus Data Conversion (Globus CD)* for intermachine messaging over TCP, otherwise. While in contrast PACX-MPI forwards all off-cluster communication operations to intermediate gateway nodes, here any process may use both local area and wide-area communication protocols. This causes certain problems with configurations where part of the application processes are behind firewalls, which are now solvable [19] by opening several holes and ranges in the firewalls for the ports used by each service of the Globus and by forcing the randomly chosen ports for *Globus IO* to fit into allowed range.

While the firewalls are not so typical for the CoC setups of our interest, the bigger problem comes from the fact that most of the Beowulf-like clusters use private address space for their nodes, which is incompatible with the current versions of the Globus Toolkit, because the security mechanisms employed, requires knowledge of the actual IP of the host that is being connected to. To circumvent this, a common TCP network and a consistent name resolution must be provided between all sub-clusters. In addition establishing a own Certification Authority (CA) would be needed to sign the certification requests of the nodes of all sub-clusters, because they do not have unique public Internet addresses and hostnames.

The distinguishing feature of MPICH-G2 is that in addition to the task to *hide* heterogeneity of the underlying configurations aims to enable the users to *manage* heterogeneity, when is required *within the standard MPI framework*. This is done by associating additional attributes to the MPI communicator, expressed within each process in terms of topology depths and colors. The two processes have same color at a particular level, when they can communicate directly with each other at this level. Currently four levels of topology depths are distinguished — wide, local and system (intra-machine) TCP messaging and vendor MPI. MPICH-G2 applications can query communicators to retrieve attribute values and structure themselves appropriately, for example by creating the new communicators that reflect the underlying network topology.

In addition, the gathered and maintained information about the actual network organization enables very effective multilevel topology-aware implementation of the collective operations in MPICH-G2, which brings substantial improvements relative to the typically used topology-unaware binomial trees approaches.

Despite all these attractive features of MPICH-G2 we do not include it in our performance evaluation because at the time of this study it was not possible to use an other MPICH-based implementation of MPI as an underlying vendor-supplied MPI. This was a very restrictive limitation because most of the commonly used free implementations of MPI and especially these for high-performance SANs (like MPICH-GM for Myrinet, MP-MPICH for SCI, MVICH for VIA over Ethernet and GigaNet, MVAPICH for Infiniband, etc.) are based on MPICH. With the latest versions 1.2.5.1 of MPICH-G2 this limitation is a little bit relaxed, to the the requirement that the underlying MPICH-based implementation is an MPICH version 1.2.5 or later and the mpirun of that underlying MPICH exports environment variables to the application. Unfortunately this still prevents to use as an underlying vendor-supplied MPI, implementations that are forked from some older MPICH versions and don't follow MPICH evolution any more (like MP-MPICH, MVICH, etc.) and implementations which follow close latest MPICH development, but have modified startup scripts that cannot export environment variables (like MPICH-GM). [1]

The most important key features and limitations of the MPI implementations discussed in this section are summarized in table 1.

3 Details about our CoC testbed

Our Cluster of Clusters testbed gives us the choice to experiment with the following cluster interconnection technologies, that are widely used currently for intra-cluster SAN's: SCI [11] [6], Myrinet [16] Gigabit and Fast Ethernet. The setup represents a heterogeneous CoC consisting of two sub-clusters each equipped with several different incompatible SAN's.

[1]This limitation is now gone and we sucessfuly have installed MPICH-G2 using MPICH derived local MPIs like MPICH, MPICH-GM and MPICH-VMI 2.0. Our first impressions of the performance and stability are very positive.

MPICH/Madeleine III	PACX-MPI	MPICH-G2
Supported Network Protocols		
TCP/IP, VIA, GM, BIP, SISCI, MPI, PVM, SBP	Depends on the MPI used for intra-cluster communication; TCP/IP, native ATM or SSL protocols for inter-cluster communication.	Depends on the MPI used for intra-cluster communication; TCP/IP through *Globus IO* for inter-cluster communication.
Routing functionality between heterogeneous networks		
Yes. The decision which nodes would become gateways is taken during the startup of the application on the basis of underlying network topology and nodes involved in computation. Every multihomed node can become gateway using any of supported protocols. Gateways cannot be unloaded from computation for performance reasons and the effect is unspecified when more than one gateway exist between two sub-networks.	Yes, through TCP/IP, native ATM or SSL protocols, and it's possible only between vendor MPIs used for intra-cluster communications. Direct connection is required only between each of the gateway nodes, which are two per sub-cluster/partition. Depending on the implementation of underlying vendor MPI the nodes for the gateway processes can be randomly assigned inside the sub-cluster.	Yes, through TCP/IP using *Globus IO*. Therefore, direct TCP connection and consistent name resolution is required between the nodes of all sub-clusters.
Binary portability of MPI applications across interconnects⁺		
No recompilation of application software is required when underlying networks are switched.	A separate application instance is necessary for each sub-cluster/ partition, linked against the respective MPI library for the interconnect technology used.	A separate application instance is necessary for each sub-cluster/ partition, linked against the respective MPI library for the interconnect technology used.
Shared-memory communication within a SMP hosts		
No	Yes, if it's supported by the underlying MPI.	Yes, if it's supported by the underlying MPI. On hosts without MPI only localhost TCP can be used.

⁺sub-clusters of binary compatible machines are assumed.

Table 1: Comparison of Features and Limitations

3.1 Hardware

First Dual-Athlon MP sub-cluster consists of 8 nodes each equipped with:

- 2 x Athlon MP 1600+ CPU 1.4GHz real core freq., 2x133MHz FSB, 256K L2 Cache
- 1 x Tyan Tiger MPX mainboard (AMD MPX chipset)
- 1 x 512 MB CL2 Unregistered PC2100 DDR SDRAM Module
- 1 x AGP Grafic Card with NVidia GeForce2 MX chipset, 32MB RAM
- 1 x 3COM 3C920 Fast Ethernet on-board NIC
- 1 x Intel Pro/100 S Desktop Fast Ethernet NIC
- 1 x Dolphin's PSB66 SCI-Adapter D33x
- 1 x Myrinet-2000 PCI64C (LANai9.3) Fiber NIC, with 2MB on-board memory

Second Dual-Xeon sub-cluster consists of 8 nodes each equipped with:

- 2 x Intel Xeon CPU, 2.4GHz core freq., 4x100MHz FSB, 512K L2 Cache
- 1 x Super Micro P4DMS-6GM mainboard (Intel E7500 chipset)
- 4 x 512 MB CL2.5 Registered ECC PC2100 DDR SDRAM Module
- 1 x ATI Rage XL integrated PCI Grafic Controller, 8MB RAM
- 1 x Intel 82544GC Gigabit Ethernet on-board NIC
- 1 x Intel 82557 10/100M Fast Ethernet on-board NIC
- 1 x Dolphin's PSB66 SCI-Adapter D331

† In addition one of the Xeon nodes contains:

- 1 x Myrinet-2000 PCI64C (LANai9.3) Fiber NIC, with 2MB on-board memory

3.2 Network Structure

All cluster nodes are diskless and they are booting same NFS-root image from the common ClusterNFS server over dedicated for administration purposes Fast Ethernet network. From the cluster part this network consists of 3COM 3C920 Fast Ethernet NIC's of the Athlon nodes, Intel 82557 10/100M Fast Ethernet NIC's of the Xeon nodes and one 24-port Fast Ethernet switch (3Com SuperStack II 3900).

In addition to the common administrative Ethernet network, there are dedicated interprocess communication Ethernet networks for each of the sub-clusters. Nodes of the Athlon sub-cluster are connected to together through their Intel Pro/100 S Desktop Fast Ethernet NIC's and a 24-port Fast Ethernet switch (3Com Super Stack II 3300). Nodes of the Xeon sub-cluster are connected together by their on-board Gigabit Ethernet NIC's and a 12-port Gigabit Ethernet switch (3Com Super Stack III 4900).

We also connected these two networks together to simulate the most common and affordable in the practice interconnection between two sub-clusters - by common TCP/IP network. This way all the nodes from both sub-clusters are connected together by TCP over Fast Ethernet, which is relatively slower and exposes much higher latencies in comparison with the SAN's like SCI, Myrinet and Gigabit Ethernet. Only the connection between nodes of the Xeon sub-cluster is faster due to the Gigabit Ethernet NIC's.

The SCI adapters of the nodes of each sub-cluster are connected into separate independent SCI rings. Therefore, although all of the cluster nodes have one SCI adapter they cannot communicate directly over SCI network with nodes belonging to a different sub-cluster.

All the (8 from Athlon sub-cluster + 1 from Xeon sub-cluster) Myrinet adapters are connected into a single Myrinet network using 16-port Myrinet switch (M3F-SW16). This way the nodes participating in Xeon sub-cluster can exchange messages with the nodes participating in Myrinet network of Athlon sub-cluster only if they are routed through the single Xeon gateway node equipped with additional Myrinet adapter or through the common TCP network. We also tried to provide for our experiments an additional gateway node equipped with two SCI adapters, connected to both SCI rings. Unfortunately this configuration was incompatible with MP-MPICH.

For some of the PACX experiments we provide an additional Gigabit Ethernet connection between the nodes from the Athlon sub-cluster and the nodes of Xeon sub-cluster which acting like a gateways. Unfortunately due to some hardware restrictions and peculiarities of our hardware (lack of free 64 bit PCI slots, broken PCI BIOS) the throughput of this connection (256 MBit/s) is more closer to Fast Ethernet than to the Gigabit Ethernet. In contrast we measure 895 MBit/s between two integrated Intel/Pro 1000 GBE NICs. That's why, in the following text we would refer to this additional connection not like a GBE, but like a 256Mbit/s connection, just to emphasize to the readers that in typical cases better results should be expected, when a normally-functioning GBE connection is used.

3.3 System Software

For the purpose of this study Red Hat 7.3 Linux was used with updated Red Hat 2.4.18-27.7.x kernels. The latest available versions of the following drivers and message-passing middleware were installed:

- MPICH 1.2.5
- Dolphin SCI Cluster Software Source package (DISsp) release 1.34
- Shared Memory Interface (SMI) release from October 26-th 2002

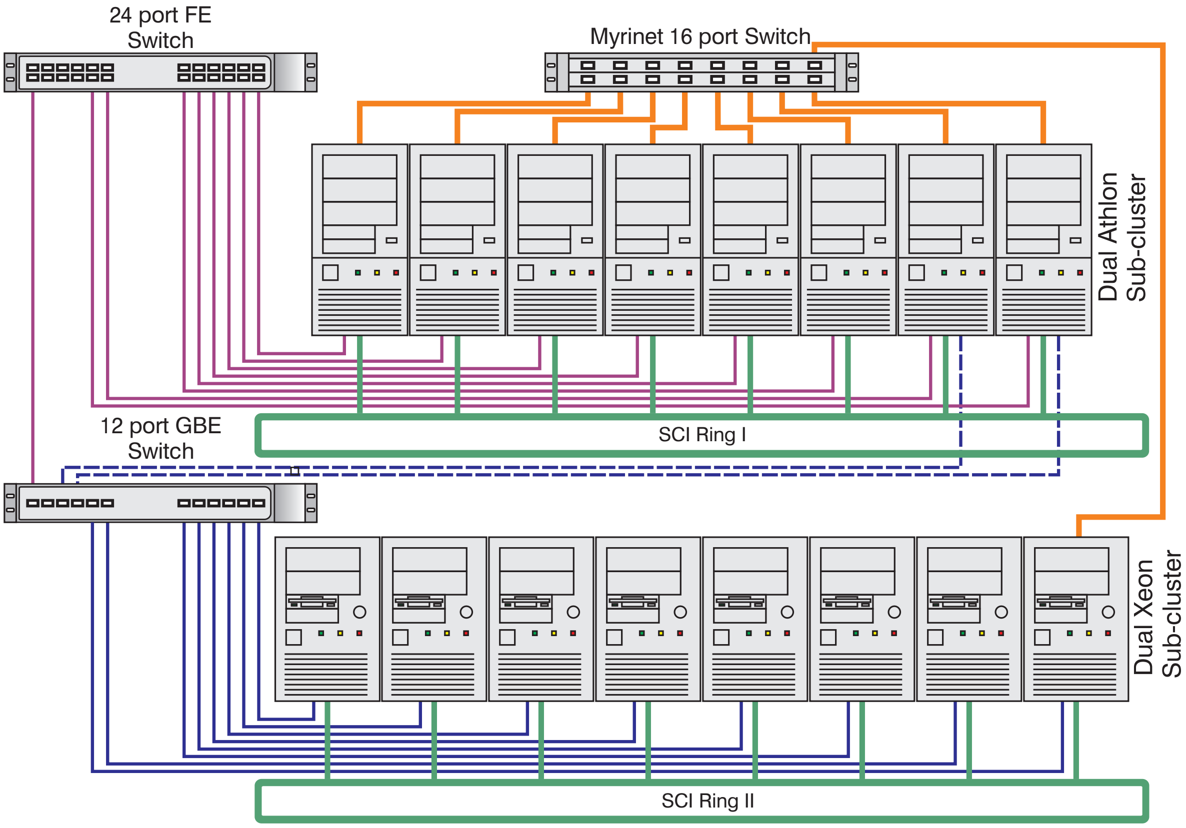

Figure 1: Structure of the dedicated interprocess communication networks of our CoC testbed

- MP-MPICH release October 31-th 2002
- Myricoms's GM release 1.6.4
- Myricoms's MPICH-GM release 1.2.5..9
- PM2 release from March 05-th 2003
- MPICH-Madeleine III release from March 10-th 2003
- PACX-MPI version 4.1.4

4 Benchmark Results

In order to obtain complete and well-balanced view of evaluated MPI implementations we performed series of widely recognized synthetic and application benchmarks. For estimating the low-level performance characteristic of the MPI implementations we use the the industry standard Pallas MPI Benchmark (PMB-MPI1). PMB is built with objectives to provide a concise set of benchmarks targeted at measuring important MPI functions (point-to-point message-passing, global data movement, etc.), while enforcing strict requirements for run rules, set of required results, repetition factors and message lengths and at the same time don't impose an interpretation on the measured results (execution time, throughput and global operations performance).

In addition to the the synthetic benchmarks we performed two application benchmarks, the results of which are much more likely to reflect real-world performance thanks to their use of real application code.

Due to the size limitation of this document, only the most representative results obtained that validate the conclusions would be shown here. The rest of the results can be found at [1].

4.1 Synthetic Benchmark Results

Using the Pallas MPI Benchmark (PMB-MPI1) we measure the performance of all single protocol MPI implementations for every available in our sub-clusters interconnect technology, namely MPICH-GM over Myrinet; MP-MPICH over SCI; MPICH over Fast and Gigabit Ethernet; MPICH/Madeleine III over Myrinet (GM), SCI and Fast and Gigabit Ethernet. In addition, to estimate the overhead of running PACX-MPI over the native intra-cluster implementations we benchmarked the following combinations: PACX-MPI over MPICH-GM, PACX-MPI over MP-MPICH, and PACX-MPI over MPICH (Fast and Gigabit Ethernet). Due to the already mentioned space limitations we are presenting in fig. 2, 3, 4 and 5, only the bandwidth and latency results of the most important PingPong sub-test between two nodes of the same sub-cluster.

As one can see from the obtained results, inside the sub-clusters PACX-MPI introduces very small overheads in the latencies of slow interconnects like Fast Ethernet, ranging from 6% for short messages to insignificant for a messages above 512 Bytes. In opposite, for the fast interconnects like SCI, Myrinet, GBE, the overhead of PACX-MPI over message latencies is significant, starting from 20-90% for short messages and becoming insignificant only for messages above 4-16KB. In similar way the impact over bandwidth is negligible for slow interconnects like FE but it can reduce the bandwidth with 35-45% for the short messages on the fast interconnects. The 95% of underlying MPI bandwidth is achieved only for messages above 2-16k, depending from the fast interconnection type.

The obtained results also make it possible to compare the multiprotocol MPICH/Madeleine III against the optimized only for one protocol MPI implementations. MPICH/Madeleine III have significant advantages over original MPICH on FE and especially on GBE on both bandwidth (+20%) and latency (-17%). For SCI MPICH/Madeleine III is significantly behind MP-MPICH for messages under 128 bytes (-45% in bandwidth and +77% in latency), but for longer messages it keeps close and even outperforms significantly MP-MPICH for messages above 512k. Only for Myrinet using GM library the performance is lagging behind the competition in face of MPICH-GM. For messages above 32k MPICH/Madeleine III is in pair with MPICH-GM but for shorter ones it's far behind with 85% lower bandwidth and 500% higher latency for certain message sizes. The explanation for this is that GM is the newest protocol supported by Madeleine. Before the only available in Madeleine protocol for Myrinet was BIP. We where reluctant to use BIP in our tests, first because is difficult to be supported in one cluster together with GM, and second because when MPICH/Madeleine uses BIP it rely on the external BIP's loader to spawn the application instances on the nodes equipped with Myrinet. This BIP loader was not working in our setup where every node have more than one network interface (because of the diskless operation).

To measure the effectiveness of the evaluated middleware for building aggregate CoC we ran PMB-MPI1 test on all possible combinations of CoC achievable with the PACX-MPI and MPICH/Madeleine III and the available interconnects in our testbed:

- MPICH over both sub-clusters (FE in Athlon sub-cluster with GBE in Xeon sub-clusters). The results of these tests will serve as reference for the usefulness of the evaluated middleware for building CoC.

- MPICH/Madeleine III, TCP over both sub-clusters. The use of this configuration shouldn't bring some special advantages over plain MPICH except that one can easily switch the underlying network configuration without need of rebuilding it's applications.

- MPICH/Madeleine III, GM-SCI. In this case Myrinet (GM) in Athlon sub-cluster and SCI in Xeon sub-cluster is used. The gateway becomes the Xeon host equipped with both SCI and Myrinet interfaces;

- MPICH/Madeleine III, GM-GBE. In this case Myrinet (GM) in Athlon sub-cluster and GBE in Xeon sub-cluster is used. The gateway becomes the Xeon host equipped with both GBE and Myrinet interfaces;
- MPICH/Madeleine III, SCI-GM-SCI. In this case SCI is used in both Athlon and Xeon sub-clusters. The gateways are two - one Athlon and one Xeon hosts equipped with both SCI and Myrinet (GM) interfaces;
- PACX-MPI over MPICH-GM (Myrinet) in the Athlon sub-cluster with PACX-MPI over MP-MPICH (SCI) in the Xeon sub-cluster. Two separate experiments are performed with different speeds of the link between the routing daemons — FE, and sub- GBE (256Mbit/s).
- PACX-MPI over MPICH-GM (Myrinet) in the Athlon sub-cluster with PACX-MPI over MPICH (GBE) in the Xeon sub-cluster. Two separate experiments are performed with different speeds of the link between the routing daemons — FE, and sub- GBE (256Mbit/s).
- PACX-MPI over MP-MPICH (SCI) in the Athlon sub-cluster with PACX-MPI over MP-MPICH (SCI) in the Xeon sub-cluster. Two separate experiments are performed with different speeds of the link between the routing daemons — FE, and sub- GBE (256Mbit/s).
- PACX-MPI over MP-MPICH (SCI) in the Athlon sub-cluster with PACX-MPI over MPICH (GBE) in the Xeon sub-cluster. Two separate experiments are performed with different speeds of the link between the routing daemons — FE, and sub- GBE (256Mbit/s).

Again, we are presenting in fig. 6, 7 and 8 only the bandwidth and latency results for the PingPong sub-tests. The results indicate that in comparison to plain MPICH over both sub-clusters PACX-MPI have significantly higher latencies and reduced bandwidth for messages exchanged between the nodes from different sub-clusters. When we use FE links between the forwarding daemon processes for short messages PACX-MPI have 500-800% higher latencies and 80-90% lower bandwidth. For long messages these values are 10-15% increased latency and 10-13% reduced bandwidth.

When higher throughput, 256 Mbit/s links are used between the forwarding daemon processes in comparison with MPICH for very short messages we achieve 180-480% higher latencies and 70-80% lower bandwidth. Due to the higher then FE bandwidth of the links for messages above 1-2k we start seeing an upturn over MPICH reaching 90-115% higher bandwidth and 30-50% lower latencies depending from the underlying MPI implementation. This is a significant improvement achieved in relatively economical way, because by increasing only the throughput of the links between the forwarding processes we improve the throughput between each of the nodes from the separate sub-clusters.

In point of view of the achievable throughput between the nodes of the separate sub-clusters the clear winner is MPICH/Madeleine III. The explanation for this is in the ability of Madeleine to natively forward messages between all maintained protocols. Unfortunately, due to the already-discussed, low-performance implementation of Madeleine over the Myrinet's GM protocol, we were unable to achieve better results for our best-case CoC scenarios consisting of only high-performance low-latencies, OS-bypassing network interfaces. That's why even for the best case single-hop route GM-SCI configuration MPICH/Madeleine is behind the plain MPICH in the quite important range of message sizes from 16 to 256 Bytes with up to 35% lower bandwidth and up to 50% higher latencies. Nevertheless, for message sizes above 512-1024 Bytes we are seeing an significant upturn over MPICH reaching 120-970% higher bandwidth and 50-90% lower latencies depending from the underlying interconnects.

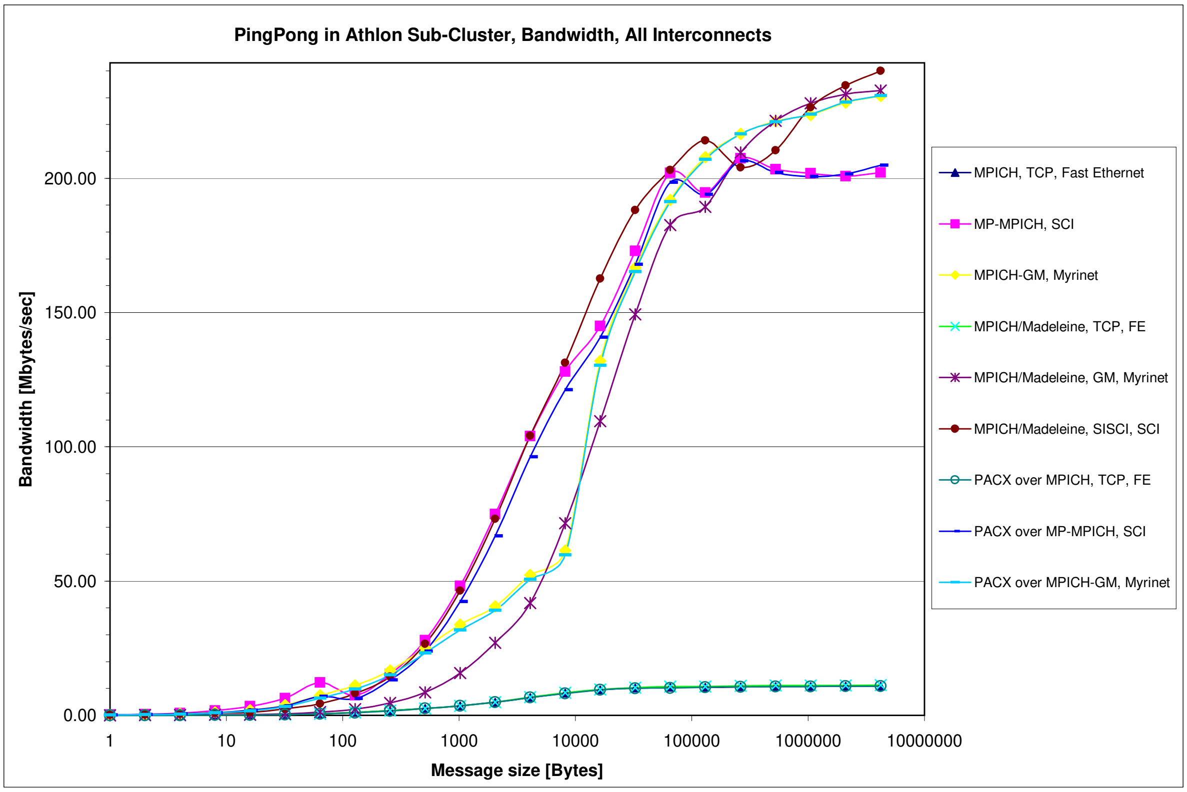

Figure 2: Athlon Sub-cluster, PingPong, Bandwidth, All Interconnects, All Message Sizes

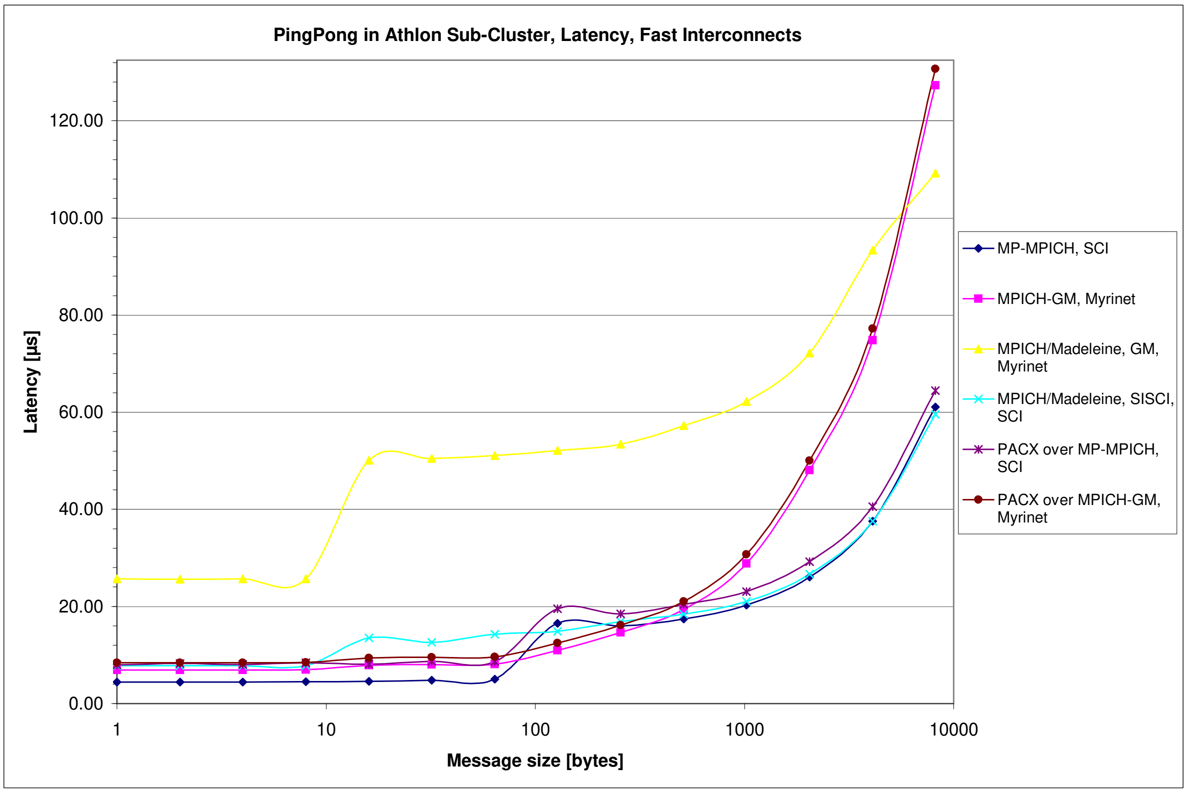

Figure 3: Athlon Sub-cluster, PingPong, Latency, Fast Interconnects, Messages up to 512 bytes

4.2 Stability Issues and Problems

During our experiments with MPICH/Madeleine III and PACX-MPI we discovered several stability issues and problems. We want to discuss them here, first in order to inform the the readers

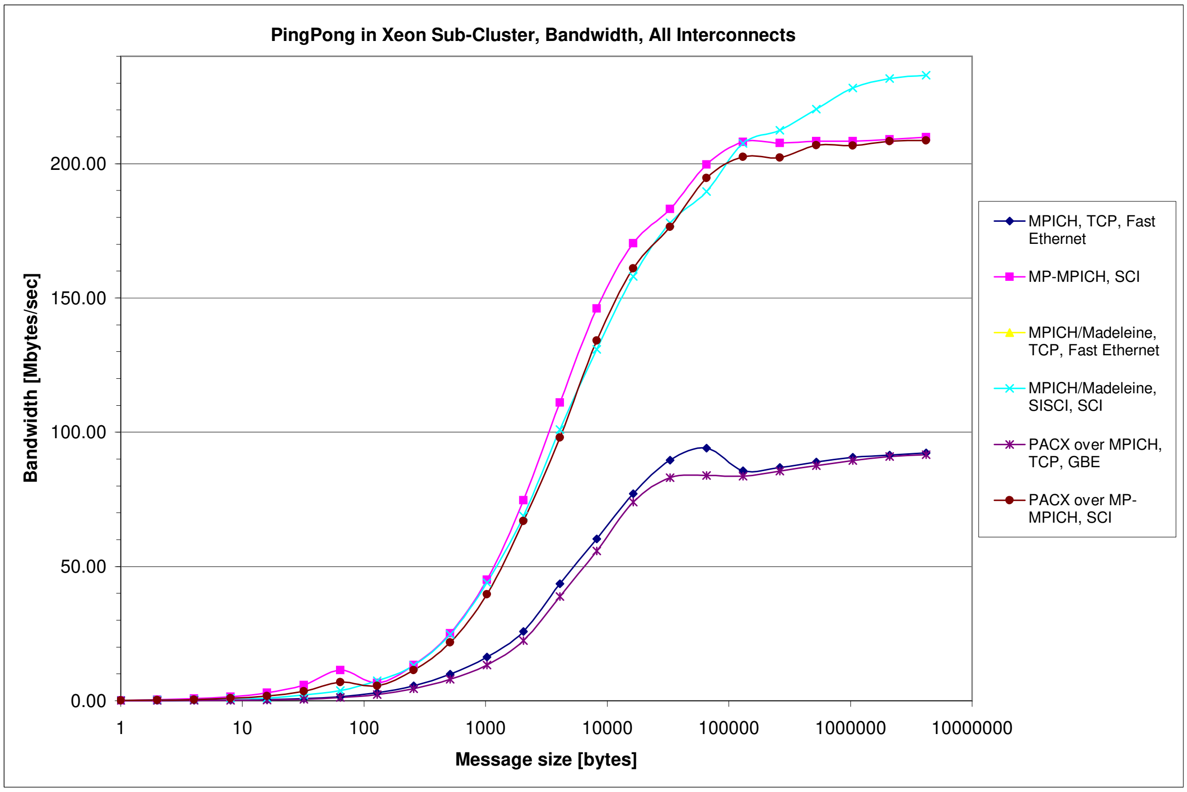

Figure 4: Xeon Sub-cluster, PingPong, Bandwidth, All Interconnects, All Message Sizes

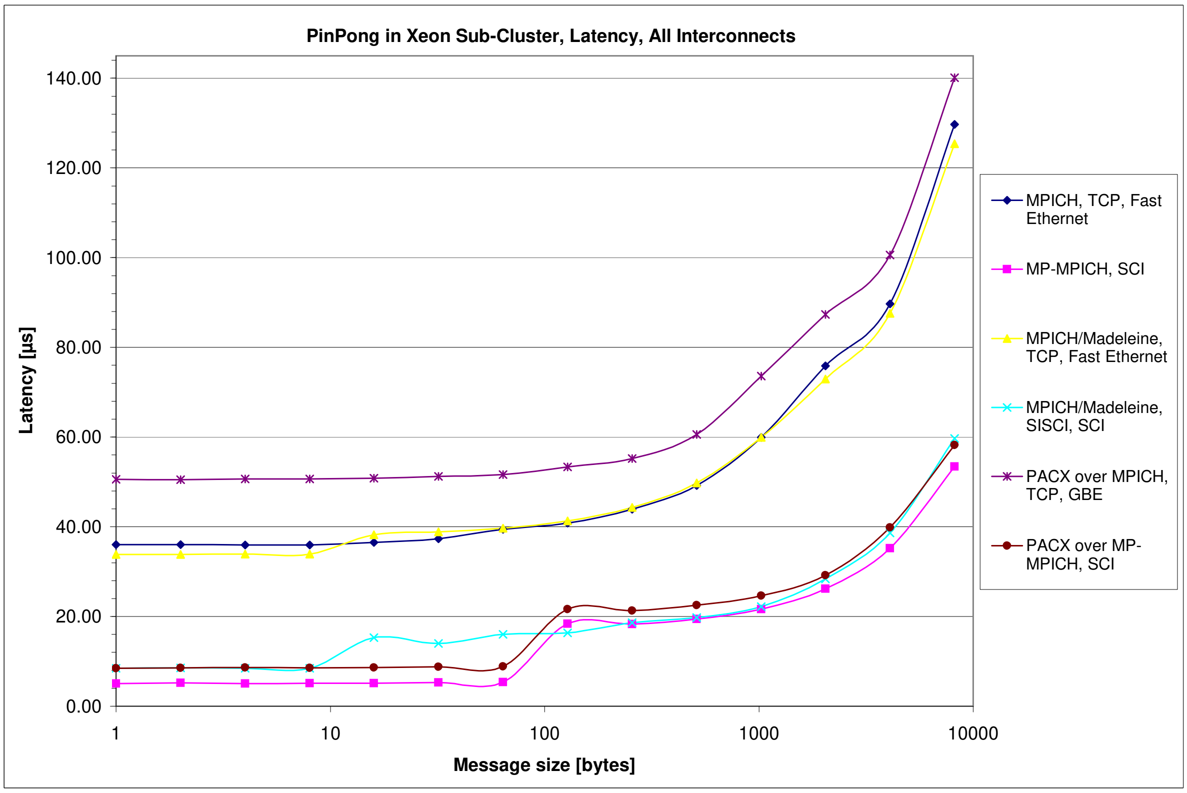

Figure 5: Xeon Sub-cluster, PingPong, Latency, All Interconnects, All Message Sizes

and second to explain why certain of our results are incomplete or missing.

First MP-MPICH, the free implementation of MPI for SCI is not completely stable in SMP mode for Alltoall communication and with for big number of processes. This automatically

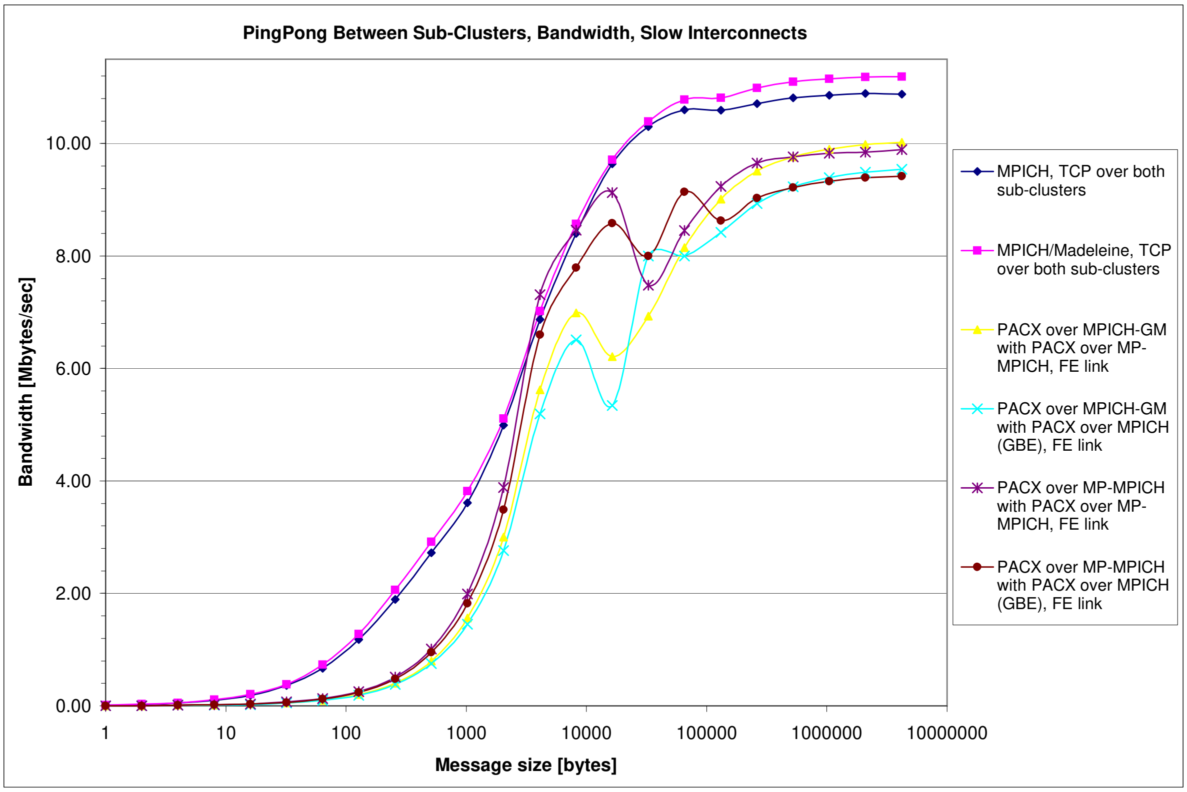

Figure 6: Between Sub-clusters, PingPong, Bandwidth, Slow Interconnects, All Message Sizes

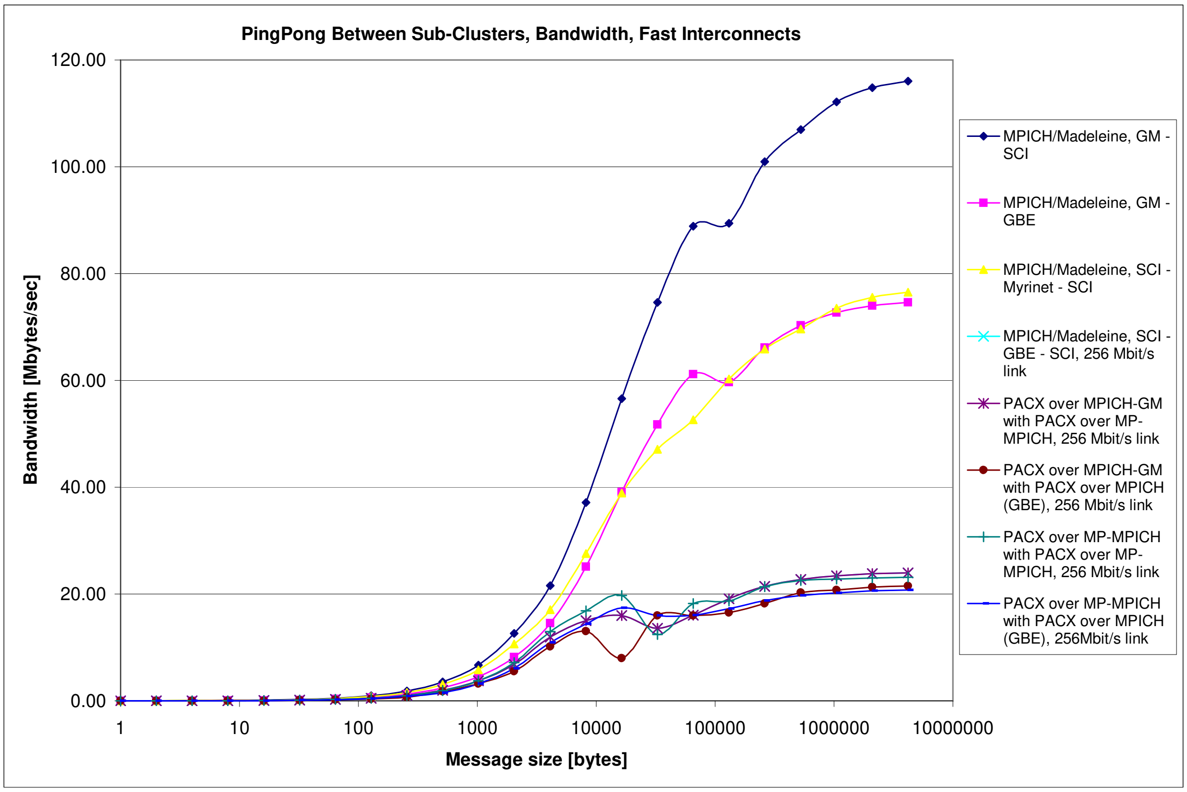

Figure 7: Between Sub-clusters, PingPong, Bandwidth, Fast Interconnects, All Message Sizes

means that the PACX-MPI on top of MP-MPICH possess the same problem. In addition, although when PACX-MPI is used with completely stable version of MPI, there are still defects introduced in some of the collective operations.

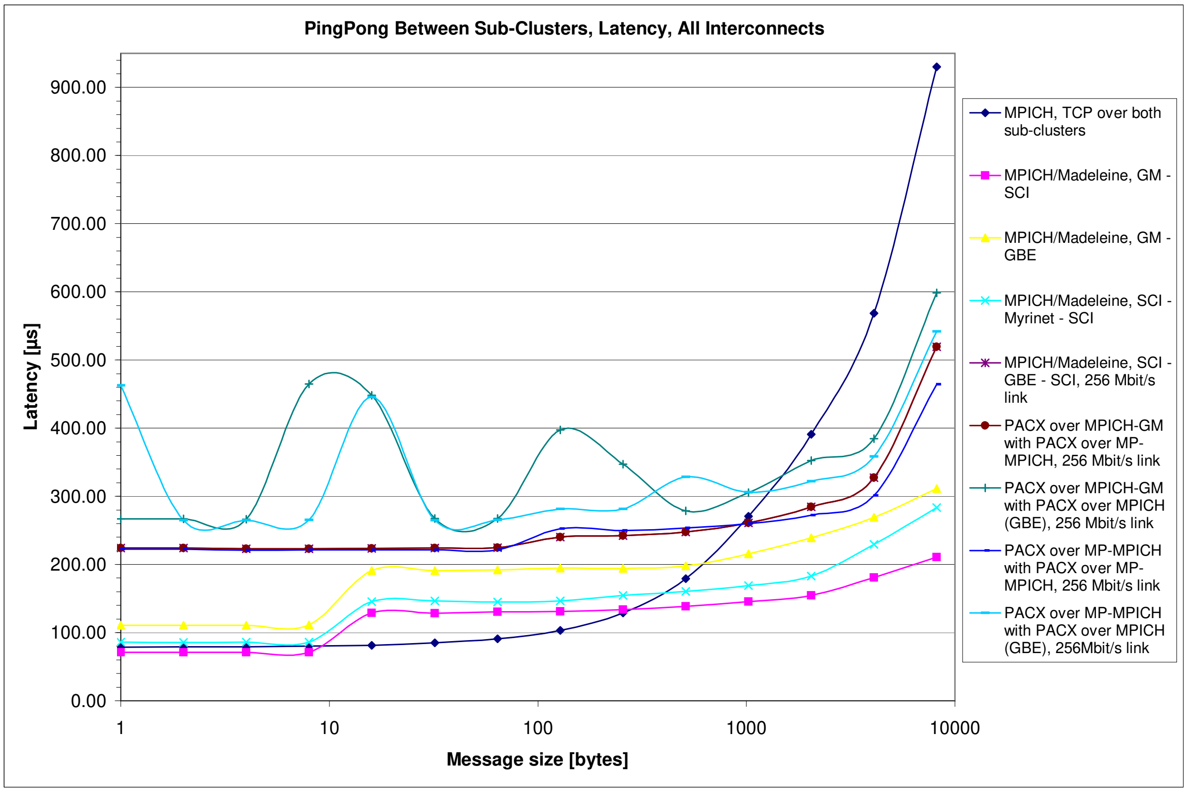

Figure 8: Between Sub-clusters, PingPong, Latency, All Interconnects, All Message Sizes

Second, when MPICH derived MPI is used as the local MPI-environment with PACX-MPI, Fortran applications don't work correctly. This is due to the fact, that PACX-MPI doesn't recover the calling arguments, and doesn't pass them correctly to the C-version of `MPI_Init`. The 4.1.4 version of PACX-MPI we used here seems to overcome this problem, but this workaround works only for the MPICH-based MPIs that use latest MPICH code base. And because MP-MPICH is forked from MPICH 1.2 we were unable to run Fortran programs like NAS Parallel Benchmarks with the combination PACX-MPI on top of MP-MPICH.

Third, we discovered also significant problems inside the implementation of the collective operations and user data types of MPICH/Madeleine III which made impossible to run HPL test with it. In addition, some other problems (probably data loss in some of the channels) caused the incorrect NAS Parallel Benchmarks results for all underlying networks different from TCP to be incorrect and we have to disqualify them. Also, beacuse the current implementation of MPICH/Madeleine MPI is unstable in SMP mode we where restricted only to the tests with single process per machine when it was used.

Despite all these problems, we think that we have collected enough experimental material that would alow us to make interesting conclusions about the relative merits and demerits associated with the different approaches used for implementing the eavaluated communication middleware and about the feasibility of the CoC computing.

4.3 Application Benchmark Results

As we have seen in section 4.1, we have received quite contradictive results from the synthetic benchmarks. The heterogeneous CoC systems tend to expose very asymmetric communication properties — inside the sub-clusters the communication performance is more than order of magnitude higher then between the sub-clusters.

It is becoming increasingly more confusing and difficult for one to judge only from the results of the synthetic benchmarks which CoC configuration is better suitable for the given class of applications. Is it better to have a system that has a lower communication performance inside the sub-clusters, but higher between the sub-clusters? Is the system with the highest communication performance rates able to overlap effectively communications and computation and to deliver also the highest real-world performance?

Trying to answer questions like these we decided to perform additional application tests with HPL Version 1.0 and and MPI enabled version of POVRAY image rendering software on every of the possible combinations of CoC achievable with the PACX-MPI and MPICH/Madeleine III and the available interconnects in our testbed.

HPL is a portable implementation of the High-Performance LINPACK Benchmark for distributed-memory computers. It attempts to measure the best performance of a machine in solving a dense system of linear equations. The problem size and software can be chosen to produce the best performance. The best performance on the Linpack benchmark is used as performance measure for ranking the computer systems in the widely recognized Top 500 computers list available at URL http://www.top500.org.

Because the goal of our tests is not to find the maximal achievable peak performance of our system but rather to estimate the qualities of the evaluated communication middleware for CoC we didn't search the optimal problem size N. Instead we choose a big enough fixed problem size N, and for all examined CoC configurations we tried to tune the the rest configurable parameters (like process grids, broadcast methods, block sizes) of the HPL benchmark to the specific characteristics of the underlying communication middleware in order to achieve maximal performance for this configuration. The results from the benchmarks are presented in fig. 9 and 10.

MPI-Povray is the parallel version of the famous ray-tracing program Povray. It employs the master-slaves parallel programming paradigm to distribute the work amongst a number of processing elements. Communication between the elements is achieved with MPI message passing. We choose a certain complex scene and render it using using all examined CoC configurations. The results are presented in fig. 11.

As one can see from the obtained results, the use of PACX-MPI makes possible to extract more performance than it would be achievable if a plain MPICH is used over the same CoC setup. Even for small 8 node configurations like ours, where the 'waste' of two CPUs per sub-cluster for the forwarding daemons reduces by significant amount the available resources for computation we have significant performance gain over the alternative to run MPICH across both sub-clusters. In addition by affordable improvement of the bandwidth of the link only between the forwarding daemons we where able to double the performance gain. In case of bigger configurations the performance gained over MPICH is even higher because the relative cost of the 'wasting' CPUs for forwarding processes decreases.

One can argue that the performance increase in the case of using faster links between the forwarding processes can also help to increase the performance of MPICH. It is true but some additional tests demonstrated that when only two of the nodes of the Athlon sub-cluster equipped with FE are upgraded with GBE, the performance MPICH over both sub-clusters increases only slightly, although in our case these two nodes are 25% of the nodes with FE. In the case of the tests with 16 processes the improvement of the MPICH performance was about 17% and in the case of 32 processes (SMP mode) it was only about 3%.

When the application doesn't stress the communication subsystem which is exactly the case with the MPI Povray (because master exchange data only with one of the slaves at a given time) there is not any advantages of using PACX-MPI or MPICH/Madeleine III. More important in this case is to have optimally fast connection between the node of the master and the rest of the nodes. We see that in this case MPICH/Madeleine III over TCP is slightly faster, due to the advantages in both bandwidth and latency over MPICH over TCP. But when a virtual channel is used for communication consisting of several different networks MPICH/Madeleine lags behind the MPICH, especially in the case SCI-GM-SCI where we have two routing processes working on two of the nodes which steal the CPU from the slaves. (We mentioned before that one of the drawbacks of the current design of MPICH/Madeleine is that the gateway nodes cannot be explicitly dedicated only for communication and it's not currently SMP enabled.) We don't see lower performance in the case of PACX-MPI because in this case we use separate CPU's for the daemon processes.

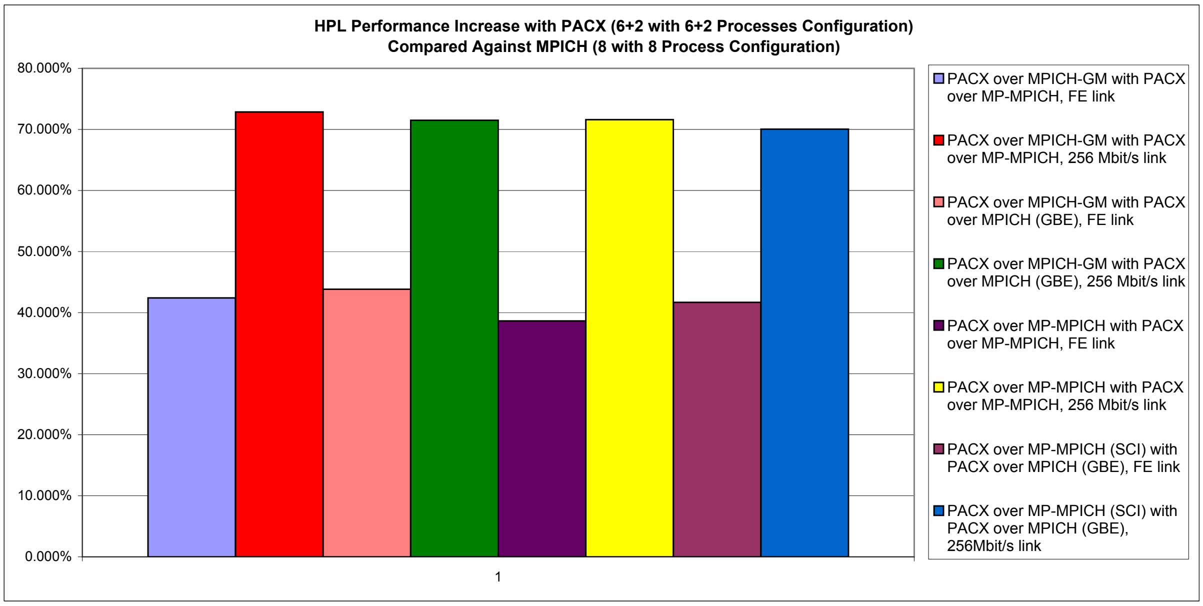

Figure 9: HPL 1.0 Application Benchmark on the agregate CoC, 8+8 processes

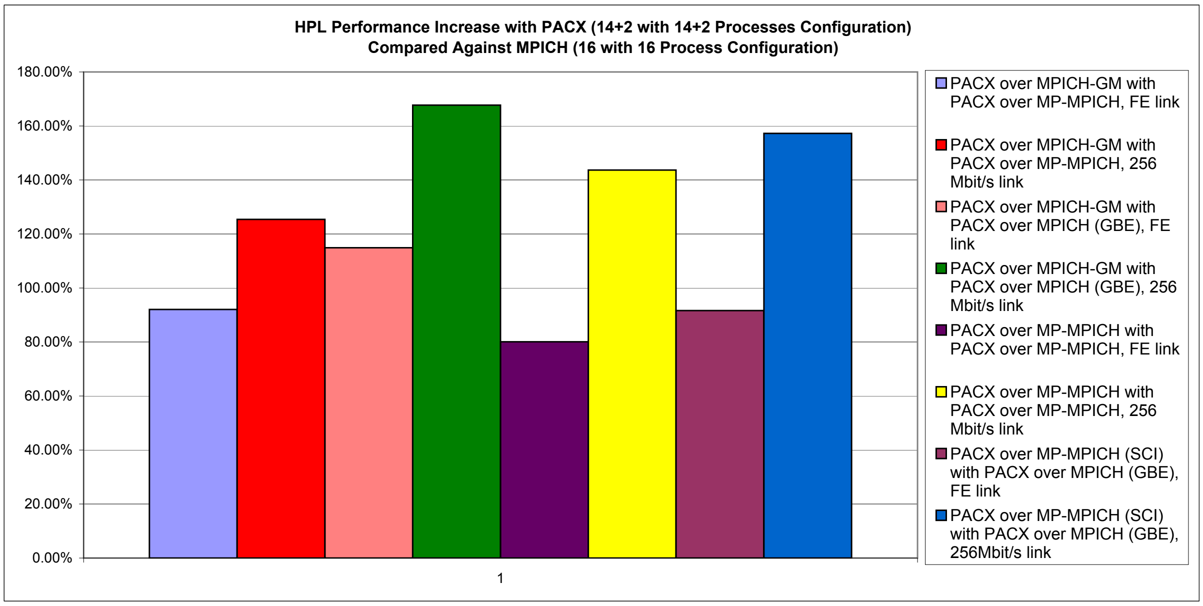

Figure 10: HPL 1.0 Application Benchmark on the agregate CoC, 16+16 processes

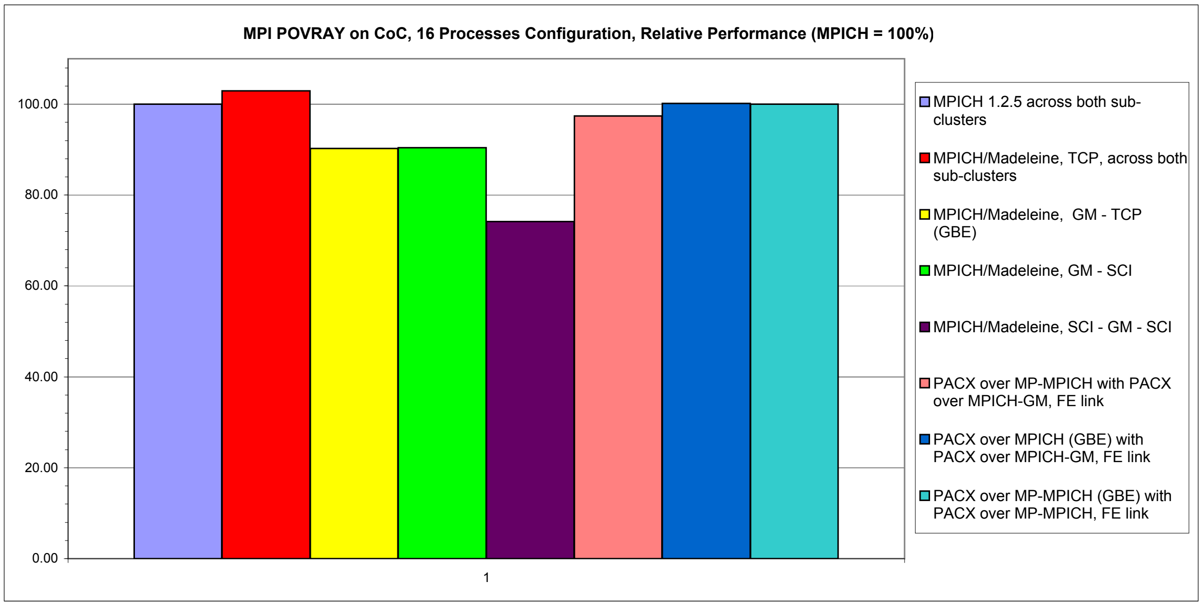

Figure 11: MPI-POVRAY Application Benchmark on the agregate CoC, 8+8 processes

5 Conclusions

Many technological, economical and even political reasons force us to deal with clusters with incompatible SANs even within a single organization. There is a growing interest to connect in effective way all these computational resources into a single-image, aggregate Cluster of Clusters, which would permit to face a more demanding, challenging, scientifical and engineering tasks. Unfortunately, due to the complexity of the problems that have to be addressed for building efficient, flexible, robust CoC, there is a scarce availability of such middleware systems.

We evaluated the two most-usable, at the time of this study, middleware systems, suitable for building CoC — PACX-MPI and MPICH/Madeleine III. We also demonstrated that are able to deliver significantly higher performance in such heterogeneous environments compared to the traditional implementations for homogeneous systems.

The design approaches of both PACX-MPI and MPICH/Madeleine III have their strong and week sides. PACX-MPI's design approach is able to deliver a significant part of the communication performance of the local MPIs inside each sub-cluster in a portable way. These local MPIs are typically very stable and highly-tuned MPI implementations providing, like a rule, the best achievable performance for a given intra-cluster SAN type.

Unfortunately, the faster the underlying local MPIs, the higher is the the overhead introduced of the PACX-MPI inside the each sub-cluster. Besides, the use of heavy wide-area protocols for inter-cluster communication, although appropriate for a Grid configurations, proves to be not very effective in CoC environments. The resulting CoC systems exhibit very high asymmetry of the communication performance between the nodes of one sub-cluster and between the nodes of different sub-clusters.

In the future implementations of PACX-MPI we would like to see a possibility to use a low-overhead SAN protocols for the communication between the forwarding daemons. Other interesting alternatives are shared memory or at least local-host TCP, like in MPICH-G2. The use of shared memory communication between the routing daemons would allow PACX-MPI to support also the most desirable single-hop route CoC configurations where the gateway nodes are equipped with interfaces to SANs of more than one sub-cluster.

The design approach of MPICH/Madeleine III makes it possible to achieve better communication performance with less asymmetry between the nodes of one sub-cluster and between the nodes of different sub-clusters. Unfortunately, it also requires significantly more work to be done to reach the stability and performance of the specialized MPIs on each type of interconnect technology.

One common drawback for both designs is that they are not very scalable. The gateway nodes are regular cluster nodes (albeit equipped with more than one SAN interfaces) and, hence, with standard I/O capabilities. Therefore, they quickly become the bottleneck when have they to forward the traffic between big sub-clusters. The problem is not so acute in the case of PACX-MPI, once because of the use of two (incoming and outgoing) forwarding daemons which can be placed on different nodes, and second, because these nodes can be unloaded from computation. To overcome issue, an introduction of multiple gateways is required. However, this would raise many additional complex problems like a routing load balancing, routing policies, etc.

The approach of routing the whole traffic of big sub-clusters through one or two gateway nodes can show significant disadvantage in setups where all sub-clusters are connected to a common Ethernet switch with enough backplane capacity, because the throughput of the gateway nodes cannot be compared with this of the backplane of a powerful switch. In such cases, the lower but guaranteed throughput of the switch would be preferable over the faster, but shared between all nodes throughput of the gateways. Therefore, in such configurations, the designs like MPICH-G2 where every process can use both intra- and inter-cluster protocols can achieve better results. We would like to mention here that MPICH/Madeleine III can also be configured to work in this way. The additional advantage is that the inter-cluster protocol can be not only TCP, but any of the supported protocols.

Finally, we would like to say that although PACX-MPI and MPICH/Madeleine III are currently unable to address all issues arising in heterogeneous CoC, they at least implement the minimal basic set of functionality required for such an application. The only major obstacle that still prevents their wide adoption is the insufficient degree of stability and ease of use, but PACX-MPI is apparently getting closer.

References

[1] PACX-MPI and MPICH-Madeleine III Benchmark Results. Available at: `http://www.tu-chemnitz.de/ ˜danib/cluster-benchmarks/`.

[2] PACX-MPI Documentation. Available online at URL: `http://www.hlrs.de/organization/pds/projects/pacx-mpi/doc/`.

[3] O. Aumage. Heterogeneous multi-cluster networking with the madeleine III communication library. In *16th International Parallel and Distributed Processing Symposium (IPDPS '02 (IPPS & SPDP))*, page 85, Washington - Brussels - Tokyo, Apr 2002. IEEE.

[4] O. Aumage, L. Bougé, A. Denis, J.-F. Méhaut, G. Mercier, R. Namyst, and L. Prylli. A portable and efficient communication library for high-performance cluster computing. In *IEEE Intl Conf. on Cluster Computing (Cluster 2000)*, pages 78–87, Technische Universitt Chemnitz, Saxony, Germany, Nov. 2000.

[5] T. Beisel, E. Gabriel, and M. Resch. An extension to MPI for distributed computing on MPPs. *Lecture Notes in Computer Science*, 1332:75–82, 1997.

[6] Dolphin Interconnect Solutions Inc. PSB-64/66, Features and Benefits. Available at: http://www.dolphinics.no.

[7] I. Foster and N. T. Karonis. A grid-enabled MPI: Message passing in heterogeneous distributed computing systems. In *Proceedings of SC'98*. ACM Press, 1998.

[8] E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed computing in a heterogeneous computing environment. *Lecture Notes in Computer Science*, 1497:180–??, 1998.

[9] W. Gropp and E. Lusk. Mpich working note: The second-generation adi for mpich implementation of mpi, 1996.

[10] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. High-performance, portable implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22(6):789–828, Sept. 1996.

[11] IEEE. *IEEE Standard for the Scalable Coherent Interface (SCI)*. IEEE Std 1596-1992. IEEE Computer Society, 1993.

[12] N. T. Karonis, B. Toonen, and I. Foster. MPICH-G2: A grid-enabled implementation of the message passing interface. June 25 2002.

[13] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, 1994. http://www.mpi-forum.org/docs/.

[14] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface, 1997.
http://www.mpi-forum.org/docs/mpi-20.ps.

[15] M. Müller, E. Gabriel, and M. Resch. A software development environment for grid-computing. *Concurrency and Computation: Practice and Experience*, 14:1543, 2002.

[16] Myricom Inc. Myrinet documentation. Available online at: http://www.myri.com.

[17] R. Namyst, O. Aumage, and L. Eyraud. Efficient Inter-Device Data-Forwarding in the madeleine communication library. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium (IPDPS-01)*, pages 86–86, Los Alamitos, CA, Apr 2001. IEEE Computer Society.

[18] R. Namyst and J.-F. Méhaut. PM^2: Parallel multithreaded machine. A computing environment for distributed architectures. In *Parallel Computing Conference (ParCo'95): Proceedings of the Conference*, volume 11 of *Advances in Parallel Computing*, pages 279–285. Elsevier, Feb. 1996.

[19] Von Welch. Globus firewall requirements. Available online at:
http://www.globus.org/security/v2.0/firewalls.html.