Evaluating Applications Performance in a Multi-networked Cluster

Albano Alves
ESTiG-IPB
albano@ipb.pt

António Pina
DI-UMinho
pina@uminho.pt

José Exposto
ESTiG-IPB
exp@ipb.pt

José Rufino
ESTiG-IPB
rufino@ipb.pt

Abstract

Traditionally, a cluster is defined as a collection of homogeneous nodes interconnected by a single high performance communication technology. However, in some cases, cluster nodes may be organized into several partitions – sub-clusters – internally interconnected by one or more selected SAN technologies. In order to constitute a multi-networked cluster, sub-clusters must share a common SAN technology or a bridge facility must be used.
In this paper we show how RoCL – a lightweight user-level communication library designed to support multi-threading in a multi-networked environment – manages to exploit such cluster organization. Performance evaluation results obtained by using two partitions of Myrinet and Gigabit SMP nodes demonstrate the usefulness of our approach both for low-level and high-level operation.

Keywords: cluster computing, SANs, multi-networked clusters, message-passing, multi-threading, performance.

1 Introduction

Low cost parallel computing is possible by using commodity SMP workstations and high performance SANs. However, researchers haven't yet proposed a programming model that easily combines multi-threading and message-passing to fully exploit such powerful hardware, particularly when multiple SAN technologies are present.

1.1 Multi-networked Clusters

Clusters are usually composed of homogeneous nodes interconnected by one only high performance communication technology. However, the process of building a computational platform may result in a mix of computation and communication hardware.
Our platform consists of three main node sets (fig. 1): an initial set composed by dual Pentium III 733MHz workstations interconnected by Myrinet – PIII sub-cluster – a second one, also interconnected by Myrinet, comprising Quad Xeon 700MHz workstations – Xeon sub-cluster – and a last one composed by dual Athlon 1.8GHz workstations equipped with Gigabit technology (SysKonnect 9821 NICs) – Athlon sub-cluster. Quad workstations had also been upgraded to include Gigabit. On-board FastEthernet adapters provide a common communication medium.

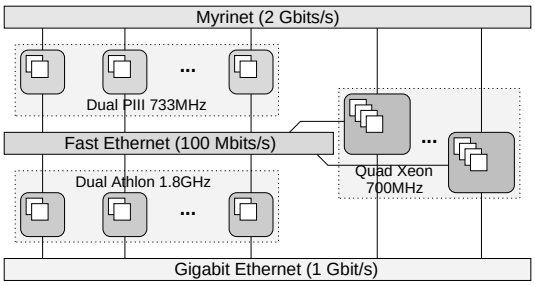

Figure 1. Multi-networked SMP cluster.

The goal is to run applications across all cluster nodes disregarding the communication technology available at each node. Common approaches, like MPI, use some kind of protocol stacking (TCP/IP for example) to guarantee the usability of heterogeneous communication hardware. However, to fully exploit high performance communication hardware it is mandatory to use lightweight protocols. The DECK environment [3], for example, uses low-level communication APIs to provide a single parallel machine by interconnecting a Myrinet and a SCI cluster.
The topology in figure 1 highlights three important issues:

- cluster nodes attached to the same high performance network (Myrinet or Gigabit) must achieve the maximum performance attainable with that technology;
- cluster nodes attached to both Myrinet and Gigabit must be able to outperform the performance achievable by using only one of these technologies;
- cluster nodes that don't share any high performance technology must be able to exchange messages not only by using the FastEthernet medium but also by taking advantage of multi-networked workstations.

1.2 Transient End-points

Traditionally, parallel programming has been used to develop efficient solutions for scientific computational problems. In this class of problems the programmer typically exploits cluster resources by creating a fixed number of communication end-points at start-up.

Highly dynamic parallel applications demand more flexible platforms because some of their components may enter/leave the system at any time thus requiring mechanisms to locate related entities at run-time.
A high performance crawling/indexing/querying system, for example, will be difficult to program using MPI or even PVM, due to the many component modules that would be used and the way they would interact each other. For instance, the system may run under multiple users, requiring advanced communication facilities, and some of its modules may be started/stopped according to external events.

2 RoCL Fundamentals

RoCL is a communication library that combines Linux multi-threading capabilities – NPTL[7] – with existing low-level communication libraries – MVIA[6] and GM[5] – in order to efficiently exploit SMP nodes and take advantage from Gigabit and Myrinet hardware[1].

2.1 Resources

RoCL introduces the resource concept as the main entity for the modeling of parallel/distributed applications. Multi-threaded applications register resources which may be used to address messages, that is, resources are animated by threads through the exchanging of messages. Multiple threads may concurrently send and receive messages using resource identifiers as message origins and destinations.
A global directory service is used to support resource registering across cluster nodes. Registered resources may be located using well known attributes.
A RoCL resource is defined by a set of attributes, provided by the user, and an identifier returned by the directory service. An attribute comprises a name and a value. That way, different kinds of application entities may be registered in a flexible manner.

2.2 Message-passing

RoCL is an intermediate-level communication library – an approach already used by Madeleine[4] – that uses low-level communication ports, available from GM, MVIA or UDP[1], to multiplex messages from several resources in a multi-threaded environment.
The main goal is to not sacrifice low-level communication performance while providing a more convenient communication model. RoCL provides buffer handling primitives to guarantee zero-copy communication and does not add any protocol layers to maintain the low-level lightweight approach[2]. Of course there are scheduling overheads because of our dispatching mechanism (the use of NPTL made possible to reduce significantly those overheads) but this is the price to pay to jump from basic node-to-node communication to a much flexible addressing scheme.

[1] RoCL over UDP was developed mainly for test purposes.

Note that the directory is also a fundamental component for the messaging mechanism since resource identifiers must be mapped into low-level communication addresses. A cache is used in order to reduce directory service usage.

3 RoCL Performance

The performance of RoCL applications depends on the operation of the directory service and the interface to the low-level communication sub-systems.

3.1 Directory Service

RoCL directory service is based on a collection of collaborative servers (one server per cluster node).

3.1.1 Local Operation

RoCL resources are always registered at the local server (at each cluster node). Resource location also uses that server to interface the entire global service.
The performance of the directory service in what it concerns local operations basically depends on the IPC mechanism, the thread/process scheduling and the search mechanism used to access the resource database.

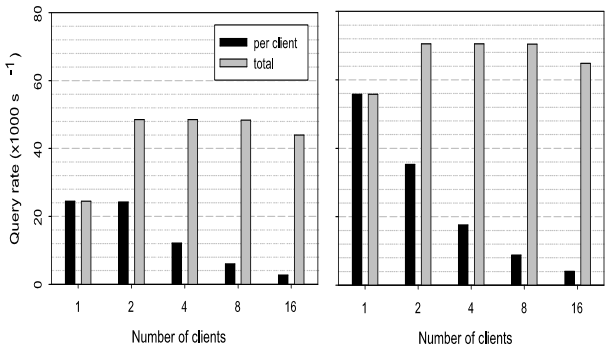

Figure 2. Maximum query rates for local operation.

Figure 2 presents the maximum querying rates possible to achieve using 1 to 16 clients to query the local server. Each query comprises two messages – a request and a reply – each one containing 256 bytes of data. Two scenarios are presented, corresponding to two different workstations: a dual Pentium III 733MHz (fig. 2:left) and a dual Athlon 1.8GHz (fig. 2:right). It is important to note that at least two clients are required to achieve maximum performance (we are using dual-processor machines) and it is also important to emphasize that above 16 clients total querying rates decrease because we are using a single thread server.

3.1.2 Global Operation

Those queries that cannot be satisfied by the local server require a global search. Servers use the Ethernet medium

to broadcast unresolved queries using UDP.

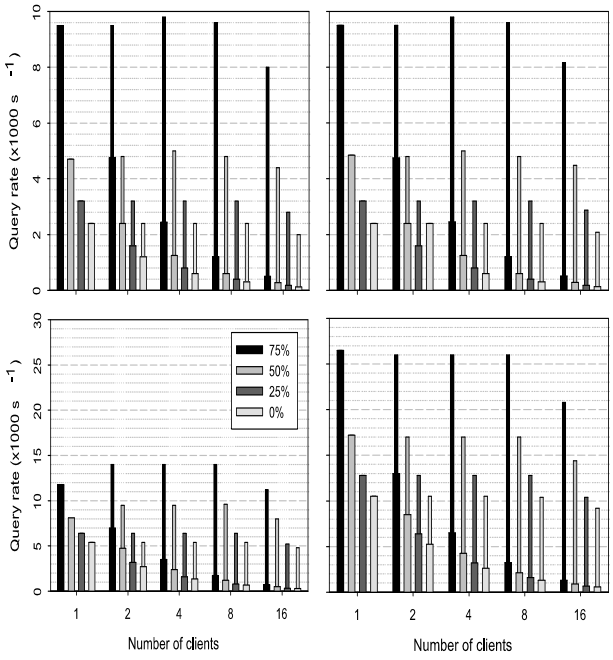

Figure 3. Maximum rates for global queries (4 nodes).

Figure 3 presents maximum querying rates obtained using two different sets of workstations – four dual Pentium III 733MHz (left-side graphics) and four dual Athlon 1.8GHz (right-side graphics) – and two different Ethernet technologies – FastEthernet (top-side graphics) and Gigabit Ethernet (bottom-side graphics). The benchmarks were carried out varying the number of clients per node – 1 to 16 – and varying the number of queries satisfied by the local server (not requiring broadcast) – 0% to 75%. Clients generate queries using total knowledge about the location of resources in order to force the uniform distribution of the queries that have to be broadcasted.

As expected, maximum per client query rates (thick bars) decrease when local servers cannot answer directly. Total querying rates (thin bars), as we have observed for the local operation, drop when we use 16 clients. Surprisingly, the Athlon sub-cluster could only outperform marginally the Pentium sub-cluster when FastEthernet was used; we don't need high performance nodes to exhaust FastEthernet. A more surprising conclusion is that Gigabit Ethernet produces a performance gain of 150% when we use Athlon nodes but the same is not true for Pentium nodes; we observed a gain lower than 50%. It seems that the impact of handling broadcast messages produced at a higher rate is too heavy for Pentium 733MHz nodes.

Figure 4 presents the results obtained using eight nodes – four Athlons and four Pentiums – connected by FastEthernet (left-side graphic) or Gigabit Ethernet (right-side graphic). It is important to note that moving from 4 to 8 nodes, using FastEthernet, drops performance about 60% while using Gigabit Ethernet produces a decrease of only

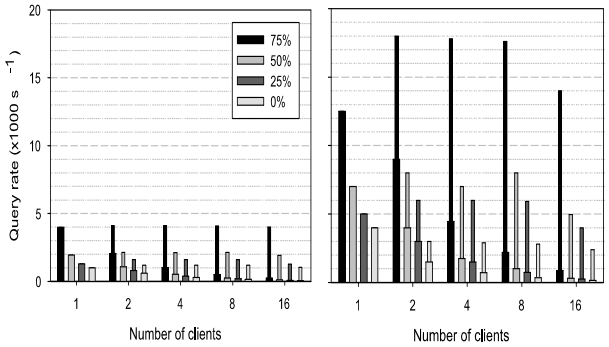

Figure 4. Maximum rates for global queries (8 nodes).

30% when compared to the performance achieved using only four Athlon workstations, leading to the conclusion that FastEthernet does not scale properly.

3.2 Communication

The evaluation of RoCL communication performance took into account resource locality, multi-thread operation, connectivity and the usage of multiple interfaces per node.

3.2.1 Intra-node Communication

Message-passing between entities in the same node should exploit intra/inter-process communication techniques.

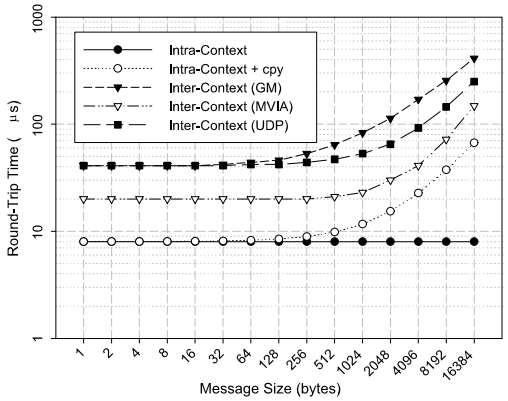

Figure 5. Intra-node round-trip times.

In RoCL, resources are created inside contexts and a context executes in a single address space. So, the sending of a message to a resource in the same context corresponds to a simple memory copy operation and a synchronization event. The memory copy is avoided if the sender instructs the library to discard the message buffer after completion. The sending of a message to a resource in another context (but in the same node) exploits the loopback facilities provided by MVIA and UDP[2]. That way, messages are sent disregarding the destination node.

[2]GM will include a loopback facility on a future release.

Figure 5 presents round-trip times for messages exchanged between resources in the same node. Tests were conducted using a Pentium III 733MHz workstation.

3.2.2 Concurrent Messaging

One of the most interesting features of RoCL is its suitability for the operation in a multi-threaded environment. Because GM and MVIA are not thread safe, RoCL introduces the essential synchronization to overcome that limitation.

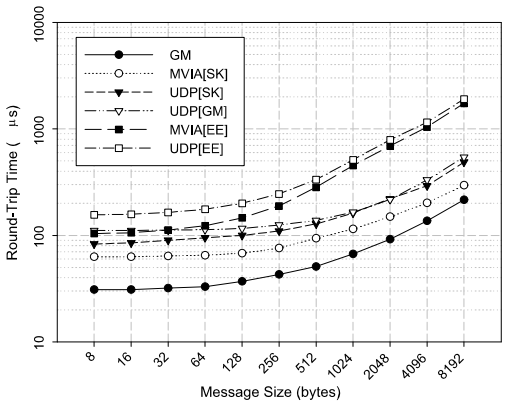

Figure 6. Simple round-trip times.

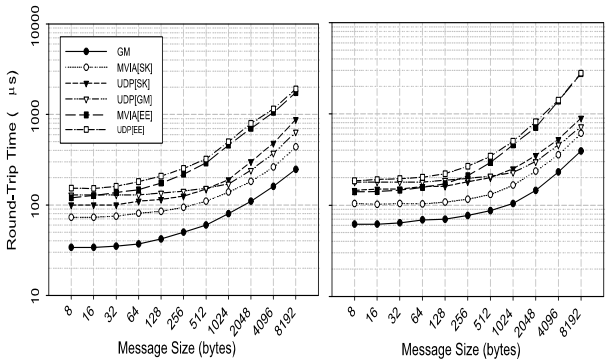

Figure 7. Round-trip times for concurrent messaging.

Figures 6 and 7 present round-trip times for messages exchanged between 1 (fig. 6) and 2 or 4 (fig. 7) pairs of resources, using 3 different low-level communication subsystems supported by RoCL (GM, MVIA and UDP) and 3 different network interfaces (Myrinet LANai9, SysKonnect 9821 and Intel EtherExpress Pro100). GM and UDP (over GM) are used to exploit Myrinet, while MVIA and conventional UDP are used to exploit the other technologies.
The results show that doubling or quadrupling the number of resource pairs engaged on communication does not cause the same effect on round-trip times. In fact, for long messages, the average round-trip times obtained using 4 pairs of communicating resources (fig. 7:right) only double the values from figure 6. For short messages even better results were obtained. This means RoCL is able to overlap communication from multiple simultaneous sources.

3.2.3 Bridging

In section 1.1 it was stated that multi-networked workstations should be used to interconnect nodes that don't share a common high-performance communication technology. RoCL includes a basic bridging functionality that allows for a resource in a Myrinet networked node to send messages to a resource in a Gigabit networked node, and vice-versa, through an intermediate multi-networked node.

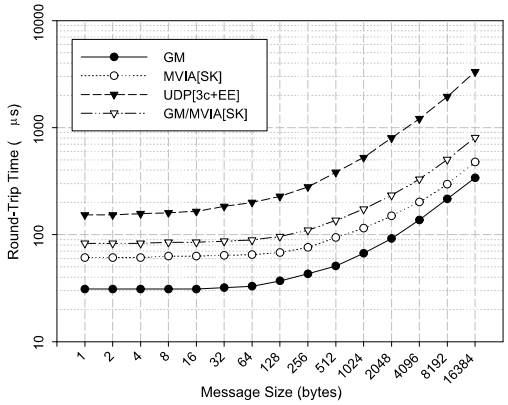

Figure 8. Round-trip times for bridged messages.

Figure 8 presents round-trip times for messages exchanged between resources from two different nodes: a dual Pentium 733MHz with a Myrinet LANai9 interface and a dual Athlon 1.8GHz with a SysKonnect 9821 interface. Messages are routed through a Quad Xeon 700MHz workstation equipped with both communication technologies. Round-trip times for messages exchanged through RoCL over UDP (using the EtherExpress interface available in the Pentium node and the 3Com 3c905C Tornado interface available in the Athlon node[3]) are also presented along with those of RoCL over GM (using 2 Myrinet nodes) and RoCL over MVIA (using 2 SysKonnect nodes).
It is important to note that the bridge mechanism is by far better than the obvious UDP approach.

3.2.4 Aggregation

Multi-networked nodes should be able to take advantage from multiple communication paths to increase throughput.
Figure 9 presents throughput values obtained by sending messages from one resource to another one, located on an second node, using RoCL over GM, RoCL over MVIA (SysKonnect interfaces) and RoCL over GM/MVIA, which combines both low-level communication subsystems.
Experimentation proved that sending 3 messages using MVIA for each 5 messages sent through GM produces the best results (for messages above 4096 bytes).

[3]Note that MVIA doesn't support 3c905C Tornado network interfaces.

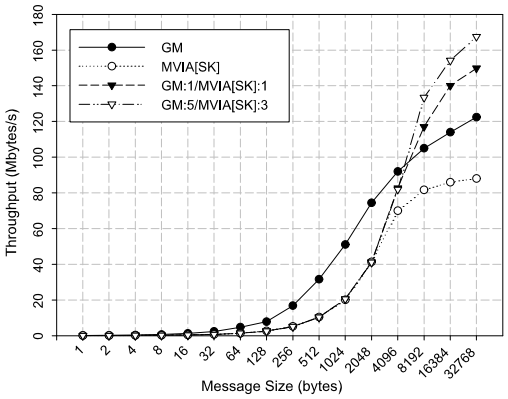

Figure 9. Throughput for aggregation schemes.

4 Performance of a RoCL Application

The usefulness and performance of RoCL has been evaluated through case studies run on the cluster depicted in 1.1.

4.1 A Case Study

The case study we here present tries to capture the basic functioning of a cluster oriented distributed hash tables platform.

4.1.1 Basic Operation

The distribution of partitions among cluster nodes is a dynamic task that may be briefly described as follows:

- the first entity to enter the system takes charge of the super-partition (the overall interval of hash indexes);
- a subsequent entity takes a random hash index and asks the owner of the partition containing that index to give it half of that partition;
- when an entity receives a sub-partition it stores the identification of the original owner and vice-versa;
- to reach the owner of a specific partition, a request is sent to a random selected partition owner which forwards it, if necessary, according to the information mentioned in the above item.

To store or retrieve data records, a client needs to reach the owner of a target partition as illustrated above.

4.1.2 Implementation

We developed two simple programs – a server, that will run on those cluster nodes that support the distributed hash table, and a client that may be used at any cluster node.

Server operation. The server starts its operation by registering itself as a resource with attributes $\{\langle"name", dhash\rangle, \langle"type", "server"\rangle\}$, where $dhash$ is the distributed hash table name provided by the user. Next it will query the directory service to find any resource with the attributes $\{\langle"name", dhash\rangle, \langle"type", "partition"\rangle\}$. If a resource is found then it will be used as the system entry point otherwise the server assumes it owns the super-partition, registers it and uses it as the entry point.
The entry point is used to request a sub-partition; a message with a random selected hash index is sent to the entry-point resource. Each server will acquire several partitions using one or more entry points.
After requesting a sub-partition the server waits for a reply message. The origin of that message is used to integrate the current sub-partition in the entire system (path to the ascendant partition). For each partition, after registering it, the server sends a message back to the origin of the reply to announce the identifier of the new registered partition.
The server creates a thread for each partition to handle four types of messages (4 different message tags): split messages, identification messages, store messages and query messages. A split message may either cause the division of the partition or it may be routed to another partition, if the hash index is outside the partition limits. An identification message is used to integrate a previous splited partition into the entire system (path to a descendant partition). Store and query messages (used to store and retrieve data) are routed just like split messages, towards the right destination.
Administrators may launch servers any time and anywhere (cluster node) without any synchronization or location concerns. It is also possible to run many servers per node. The handling of partitions, at each node, is a concurrent/parallel task directly supported by the RoCL communication model.

Client operation. Clients simply address store and query messages to a random selected entry point (a partition from a server) and wait for replies.

4.2 Performance Evaluation

In a first stage, we used two different sub-clusters – four dual Pentium III 733MHz workstations interconnected by Myrinet (LANai9), Gigabit (SysKonnect) and FastEthernet (Ether Express) and four dual Athlon 1.8GHz workstations interconnected by Gigabit (SysKonnect) and FastEthernet (3Com). At each node a server was launched to manage 32 partitions and 1 to 8 clients were executed to evaluate the maximum data retrieval rate. Each client query comprises a message request (16 bytes), that may be routed through up to 14 partitions, and a reply message (8 kbytes) received directly from the final partition.
Figure 10 presents the operation rates (number of queries per second) achieved per client. Naturally, for the same technology (MVIA or UDP over SysKonnect), clients running on Athlons achieve the higher operation rate. How-

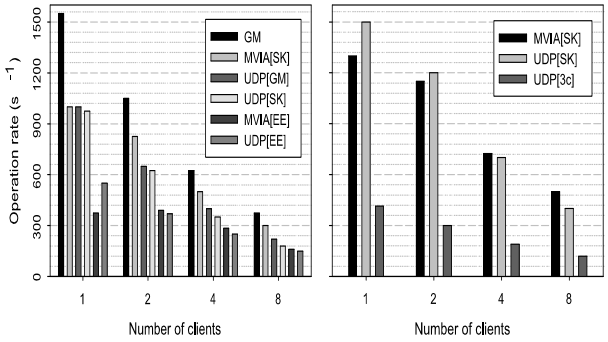

Figure 10. Operation rates (4 nodes).

ever, using one only client per node and RoCL over GM, Pentiums reach maximum performance. Note that fewer clients per node allow for clients using RoCL over UDP to achieve the same or better performance than clients using RoCL over MVIA.

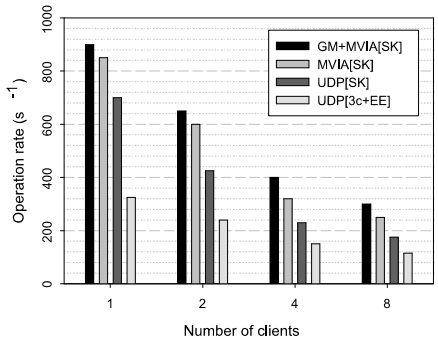

Figure 11. Operation rates (8 nodes).

In a second stage, we used an eight node cluster – four Pentiums and four Athlons interconnected by FastEthernet (Ether Express and 3Com), Gigabit (SysKonnect) and through a RoCL bridge (a Quad Xeon 700MHz workstation). The bridge allows for the operation of the cluster mixing two distinct networks: Myrinet (for Pentiums) and Gigabit (for Athlons).
Figure 11 illustrates that the operation rates per client drop when compared to those obtained using only 4 nodes. However, total operation rate achieved by clients from all nodes increase; for MVIA over Gigabit, for example, using 4 nodes, the system reaches 9600 queries per second, while using 8 nodes it is possible to achieve 16000 queries per second. This means the system scales satisfactorily.
Another important result is that the bridged operation permits to achieve better results than those obtained using only MVIA over Gigabit.
Finally, we forced RoCL to use the directory service 50% of the times in order to map resource identifiers into network addresses, by manipulating the library cache system. Figure 12 shows that, for a 4 node cluster, operation rates drop significantly when we use multiple clients. This means that the directory constitutes an important bottleneck when caches cannot satisfy all mapping requests.

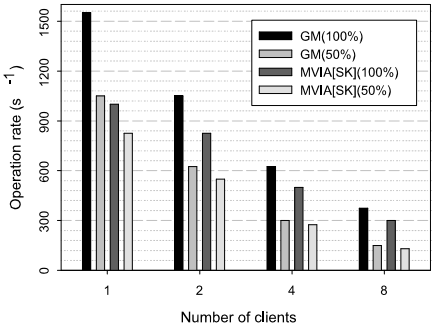

Figure 12. Operation rates (50% cache misses).

5 Conclusions

The major features and components of RoCL were submitted to exhaust testing. Multiple low-level communication protocols and high performance communication technologies were evaluated. Operation of clusters partitioned into sub-clusters were also examined. Experiments show the usefulness of RoCL for exploiting a multi-networked cluster, especially in what it concerns to performance.
A simple case study was used to emphasize how RoCL applications can be developed to dynamically run on a multi-networked cluster, by means of the resource concept, preserving low-level performance. The experiments allow to conclude that it is possible to scale applications by using RoCL to simultaneously exploit two sub-clusters.
In a near future we intend to provide higher level abstractions over RoCL to facilitate applications development.

References

[1] A. Alves, A. Pina, J. Exposto, and J. Rufino. RoCL: A Resource oriented Communication Library. In *EuroPar'03*, 2003.

[2] A. Alves, A. Pina, J. Exposto, and J. Rufino. ToCL: a thread oriented communication library to interface VIA and GM low-level protocols. In *ICCS'03*, 2003.

[3] M. Barreto, R. Ávila, and P. Navaux. The MultiCluster Model to the Integrated Use of Multiple Workstation Clusters. In *PC-NOW'00*, 2000.

[4] L. Bougé, J. Méhaut, and R. Namyst. Madeleine: Efficient and Portable Communication Interface for RPC-based Multithread Environments. In *PACT'98*, 1998.

[5] Myricom. *The GM Message Passing System*, 2000.

[6] National Energy Research Scientific Comp. Center. M-VIA: A High Performance Modular VIA for Linux. http://www.nersc.gov/research/FTG/via/index, 2002.

[7] Ulrich Drepper and Ingo Molnar. The Native POSIX Thread Library for Linux, 2003.