European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2000
Barcelona, 11-14 September 2000

PRELIMINARY INVESTIGATION OF DISTRIBUTED SHARED MEMORY SYSTEM ON A CLUSTER OF HIGH PERFORMANCE CLUSTERS

Takeshi Nanri⋆, Yoshitaka Watanabe⋆, Hiyoyuki Sato† and Masaaki Shimasaki‡

⋆ Computer Center, Kyushu University 6-10-1, Hakozaki, Higashi-ku, Fukuoka 812-8581 Japan e-mail: nanri@cc.kyushu-u.ac.jp

† Computer Centre, University of Tokyo, Japan

‡ Faculty of Engineering, Kyoto University, Japan

Key words: distributed shared memory, cluster environments, cluster of clusters

Abstract. *This paper introduces design and basic performance of the runtime system for DSM(distributed shared memory) systems on a cluster of clusters. Networking devices such as Myrinet have improved the performance of cluster systems significantly. However, because current high-performance networking devices have small number of ports, multi-clusters, clusters of high-performance clusters, is needed for larger scale computation. To ease the difficulty of programming with message passing, which is the conventional programming paradigm on cluster systems, many DSM (distributed shared memory) systems have been developed in recent years. However, there have been no DSM systems developed on multi-clusters. The DSM system of this paper provides a basic functions for implementing DSM systems on such environment. The functions are allocation of global data, read and write accesses to global data synchronization of the whole system, and mutual exclusion. The authors have evaluated the performance of the runtime system. In the experiment, the authors have used a multi-cluster which has two clusters of four PCs. The result shows that a read access to remote memory in the same cluster costs about 0.2msec, while a read access to remote cluster costs about 1.3msec. The execution time of LU decomposition on a cluster of 2 clusters each of which has 4 nodes is about 2.8times faster than the time on one node.*

1 INTRODUCTION

Cluster systems of network-connected PCs(personal computers) and WSs(workstations) have become promising platforms for high performance computing not only due to their cost-effectiveness but also because of their flexibility in employing the latest hardware and software components. Especially, networking devices such as Myrinet have improved the performance of such cluster systems significantly. Unfortunately, current high-performance networking devices have small number of ports. Therefore, multi-clusters, clusters of high-performance clusters, is needed for larger scale computation.

On cluster systems, the conventional programming paradigm is message passing. In this paradigm, programmers are responsible for explicitly inserting communication and synchronization codes. Therefore, it is difficult to develop large applications efficiently and correctly. Shared-memory programming paradigm, on the other hand, provides direct read/write accesses to the remote memory, making it easier for programmers to use. Thus, the implementation of an efficient distributed shared memory environment is a key enabling factor for feasible PC/WS cluster computing. On the other hand, a multi-cluster consists of two levels of parallelism: intra-cluster parallelism and inter-cluster parallelism. This complicates the programming on such systems much more. However, there have been no DSM systems developed on such environment.

The runtime system of this paper provides a basic functions for implementing DSM systems on such environment. The functions are allocation of global data, read and write accesses to global data synchronization of the whole system, and mutual exclusion. These functions require a kind of DMA(direct memory access) mechanism. The runtime system assumes that Myrinet is used for intra-cluster interconnect, while Ethernet is used for inter-cluster interconnect. Myrinet supports DMA mechanism by itself. Ethernet, on the other hand, does not support DMA mechanism. Therefore the authors have implemented a DMA mechanism by software. In order to handle inter-cluster request messages, a handler process runs on one node of each cluster. It routes messages of inter-cluster requests appropriately, so that DMA is done correctly.

The rest of the paper is organized as follows. Section 2 briefly discusses related work. Section 3 introduces the structure of a multi-cluster. Section 4 describes the design and the implementation of the software DSM system on a multi-cluster. Section 5 presents the results of experiments. Section 6 gives a brief summary.

2 RELATED WORKS

Recent years, many DSM systems have been developed on PC/WS clusters. Most of them can be characterized by how accesses to the virtual shared memory are handled.

On distributed memory parallel computers, most DSM systems use communication co-processors to perform direct memory accesses (DMAs) to remote memory. Therefore, it is easy to build a DSM mechanism. Special network interfaces, such as Myrinet, also have a type of DMA engine. SHRIMP [1], HPVM [2], RWC [3] and Active Message II [4] use Myrinet to implement high performance DSM systems on clusters of PCs and WSs. However, for the sake of portability, which has become an important factor in recent years, the DSM system should be designed and implemented on more commonly available hardware such as 10Mbps and 100Mbps ethernets. Since such systems do not have any support from special hardware for remote DMA, a communication library to perform asynchronous accesses to remote memory is needed.

The virtual shared memory of TreadMarks [5] is implemented on the communication library which uses the paging mechanism of UNIX. When a page fault occurs on one processor in the TreadMarks system as a result of an access to global memory, it sends a request to the owner of the page. The request is served by the SIGIO handler invoked by the owner. Split-C/PVM [6] is a portable DSM system, which uses PVM as the base communication library since PVM is available on almost every platform. An active message layer is implemented to provide a facility to manage shared memory on the message passing model of PVM. The active message layer, implemented with user-level signals in UNIX, enables asynchronous accesses to remote memory. Adsmith [7] is another DSM system on PVM-based clusters. It supports object-based shared memory accesses. The virtual shared memory of Adsmith is managed by a special daemon running on each machine in the cluster.

Systems in which message handling is accomplished via a dedicated process require context-switching between processes. Generally, this cost is significantly high. The cost of context-switching between threads, on the other hand, is relatively low. Therefore, using a multithread model is an efficient method for implementing software DSM's.

There are some existing multithread-based DSM systems implemented on commodity hardware. A DSM system developed at the University of Tokyo [8] supports replicated shared memory for wide-area networks on ATM-connected SPARC Stations. Munin [9] is a multithread-based DSM system which uses worker threads to handle consistency and synchronization functions. The parallel computer, EM-X [10], has a global shared memory space supported by multithreading. Brazos [15] is a software DSM system implemented using multithreading and selective multicasting on Windows NT. Filaments [11] and CVM [12] also support multithread-based software DSM systems on various UNIX platforms. Because of multithreading, these systems can reduce communication latency by overlapping communication and computation.

As for multi-clusters, there have been no DSM systems developed. A multi-cluster has two levels of parallelism: inter-cluster parallelism and intra-cluster parallelism. The structure is similar to the structure of a cluster of SMP (Symmetric Multi-Processors) machines, which has inter-node parallelism and intra-node parallelism. On SMP clusters, some DSM systems have already been implemented [13, 14].

3 MULTI-CLUSTER

Recently, clusters with high-performance network have come into wide use. Therefore, large-scale computing using more than one cluster systems is becoming available in a laboratory or among neighboring laboratories. However, this kind of multi-clusters mostly use commodity network such as Ethernet to connect each cluster. Moreover, in this kind of multi-clusters, each cluster manages each machine with private addresses and consists only one machine which can communicate with other clusters in the multi-cluster. This machine is called the gateway machine of the cluster. Other machines in a cluster cannot communicate with other clusters directly, but needs communication via the gateway. The DSM system introduced in this paper targets on this kind of multi-clusters.

Another important point for using this kind of multiclusters is heterogeneity; architecture or OS of clusters in a multi-cluster may different each other. We are improving the DSM system so that it can support heterogeneous multi-clusters.

4 DSM SYSTEM

4.1 Structure of the system

The core part of DSM system is the mechanism for remote memory access: reading from and writing into memory on remote machines directly. On a cluster of PCs or WSs, the mechanism is implemented by using multi-threads, signal handlers or polling functions. With intelligent network interfaces like Myrinet, the mechanism can be implemented by running protocols for remote memory access on the network controller of the interface.

DSM systems on multi-clusters require a mechanism which supports remote memory access on two levels of networks: inter-cluster network and intra-cluster network. Therefore, the mechanism is implemented hierarchically. For intra-cluster level, the mechanism uses the network controller of Myrinet to support remote memory access among machines in the cluster. For inter-cluster level, on the other hand, the mechanism runs a request handling process on the gateway machine of each cluster to support remote memory access among clusters.

Intra-cluster level mechanism

The DSM system uses NICAM, a communication layer for Myrinet [16]. NICAM uses Active Message mechanism on the Myrinet network interface to provide operations of direct remote memory access communication in a cluster. With NICAM, data is transferred using the direct memory access engine on the Myrinet network interface.

Inter-cluster level mechanism

Operations of direct remote memory access communication inter clusters is managed by the request handler running on the gateway machine of each cluster. To communicate with each machines in the cluster, the request handler prepares channels on global memory space of the cluster. Requests from each machine and replies to each machine are transferred via these channels. To communicate with request handlers of other clusters, on the other hand, the request handler connects a TCP socket to the request handler of each cluster. Requests, replies and data are transferred from one cluster to another via this socket.

For example, when machine p1 in cluster c1 reads data from memory space in cluster c2, the request handled as follows.

1. p1 writes the request message into the channel.
2. The request handler of c1 reads the request from the channel.
3. The request handler sends the request to the request handler of c2 via the TCP socket.
4. The request handler of c2 receives the request from the socket.
5. The request handler copies the requested data from the memory space of cluster c2 by using remote memory access communication of NICAM.
6. The request handler of c2 sends a reply message with the requested data to c1 via the TCP socket.
7. The request handler of c1 receives the reply message and data from the socket.
8. The request handler writes the data into the address specified in the request message.
9. The request handler writes the reply message into the channel for p1.
10. p1 reads the reply message and continues execution.

The request handler repeats polling channels and sockets. After polling all channels, the request handler gathers requests in groups according to target. Then it sends each group as a message to its destination cluster. After that, the request handler polls TCP sockets and handles requests and replies reached to the cluster.

4.2 Shared-memory interface

Table 1 shows the programming interface provided by the DSM system in this paper.

The DSM system provides two-dimensional address space as the global memory space. Therefore, the global address used in operations `read` and `write` needs two-dimensions. Detail of the two-dimensional address space is described in the following section.

The remote accesses such as `read` and `write` are blocking operations. Therefore, if a machine issues these operations, it waits the reply message before it goes to the next operation.

`read`	blocking read access to global memory
`write`	blocking write access to global memory
`barrier`	synchronize all machines in the multi-cluster
`lock`, `unlock`	handles mutual exclusive locks

Table 1: Programming interface of the DSM

4.3 Two-dimensional address space

The DSM system provides two-dimensional global space: the inter-cluster dimension and the intra-cluster dimension. All machines in the multi-cluster can read from or write to the global address directly.

The inter-cluster dimension is the number which specifies an unique cluster. On the other hand, the intra-cluster dimension is the address of virtual memory space shared by all machines in a cluster. The one-dimensional virtual shared memory in a cluster is supported by NICAM.

Because the shared memory space of the DSM system is two-dimensional, programmers must specify allocation of data explicitly. However, this means that programmers can control data distribution in detail. On multi-clusters, inter-cluster communication is one of the most costly operations, because the gateway machines will be a bottle-neck of such communication. The performance of inter-cluster network may also gain the cost of the communication. Therefore, positions of shared data must be decided carefully to reduce inter-cluster communication. Two-dimensional shared memory space enables such kind of optimization.

5 PERFORMANCE

5.1 Experimental environment

The platform of experiments in this paper is COMPaS, a cluster of SMPs in RWCP(Real World Computing Partnership) in Japan. It has 8 PCs connected by Myrinet switch and Fast Ethernet switch. Each PC has four 200MHz Pentium Pros and 128MB memory and runs Solaris 2.5.1. Though PCs are SMP machines, programs in this experiment runs one process on each PC. Multi-cluster environment is build by dividing PCs in COMPaS into two groups of 4 PCs. PCs in a group use Myrinet to communicate each other. Communication among PCs in a group is done by Myrinet, while communication between groups is done by Ethernet via the gateway machine running on each group. Figure 1 shows the structure of the multi-cluster.

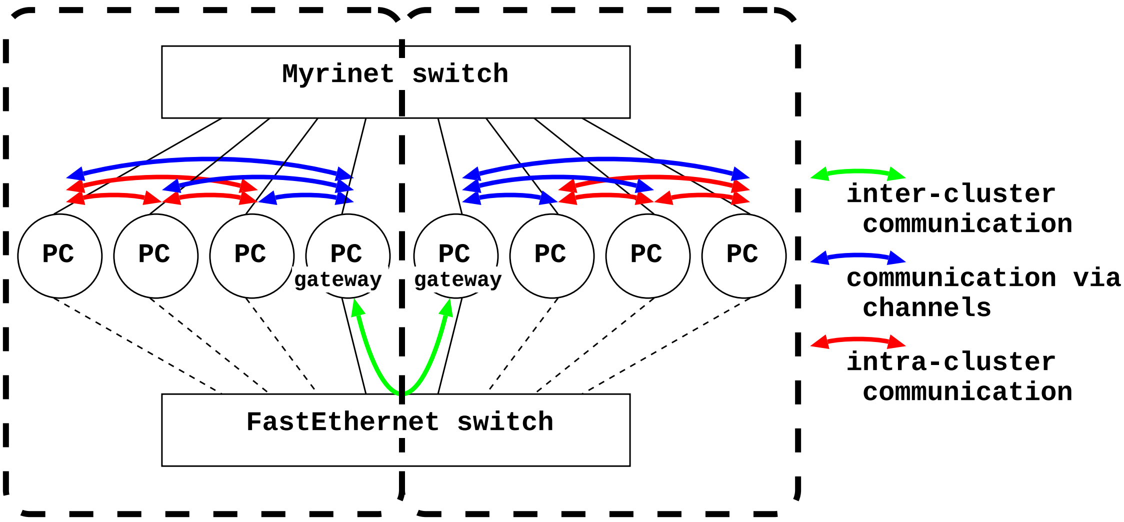

Figure 1: The structure of the multicluster

	read (μsec)	write(μsec)
memory access		
local	8	8
intra-cluster	208	200
inter-cluster	1358	1161
synchronization		
barrier	$969\mu sec$	

Table 2: Basic performance

5.2 Basic performance

The basic performance of the DSM system is shown in Table 2. Read latency (or write latency) are elapsed time required for reading (or writing) 8 byte data. Barrier time, on the other hand, is an elapsed time to synchronize all machines in the multi-cluster.

One inter-cluster read need following operations.

1. NICAM-write of a request to a channel on the gateway
2. local-read of the request from the channel
3. TCP-transfer of the request between clusters
4. NICAM-read of the data specified in the request
5. TCP-transfer of a reply message with the requested data

NICAM read	$200\mu sec$
NICAM write	$188\mu sec$
TCP roundtrip	$440\mu sec$

Table 3: Elapsed time for operations

# of clusters	# of nodes/cluster	elapsed time (sec)
serial		32.878
1	1	42.463
1	2	21.675
1	3	14.721
2	1	23.801
2	2	14.275
2	3	11.707

Table 4: Elapsed time of LU decomposition

6. NICAM-write of the data into the address specified in the request
7. local-write of the reply message to the channel
8. NICAM-read of the reply message from the channel

Table 3 shows the time for each operation on COMPaS. From this table, total time for transferring data is calculated as 1234usec. Remaining 124usec is spent for waiting message or handling data structure of the DSM.

5.3 Performance of LU decomposition

Table 4 shows the elapsed time of computing LU decomposition executed on the DSM system on COMPaS. The size of the matrix used in this experiment is 1200×1200. Because one PC is used as the gateway machine, no more than three PCs are used in each cluster, The program runs 2.8 times faster with two clusters of three PCs, than with one PC.

6 Conclusions

In this paper, design and basic performance of the DSM system on a multicluster is introduced. The result of the experiment done on COMPaS shows that the DSM system on a multicluster can be one of the platforms for high-performance computing. To gain the performance of the system, a kind of cache mechanism is required. In addition to that, we are working on adapting the system to heterogeneous environment.

REFERENCES

References

[1] C. Dubnicki, A. Bilas, K. Li and J.F. Philbin, "Design and Implementation of Virtual Memory-Mapped Communication on Myrinet", *Proceedings of 11th International Parallel Processing Symposium* (1997).

[2] A. Chien, S. Pakin, M. Lauria, M. Bunchanan, K. Hane, L. Giannini and J. Prusakova, "High Performance Virtual Machines (HPVM): Clusters with Supercomputing APIs and Performance", *Eighth SIAM Conference on Parallel Processing for Scientific Computing (PP97)* (1997).

[3] H. Tezuka, A. Hori, Y. Ishikawa and M. Sato, "PM: An Operating System Coordinated High Performance Communication Library", *High-Performance Computing and Networking, volume 1225 of Lecture Notes in Computer Science*, 708-717, Springer-Verlag (1997).

[4] A. Mainwaring and D. Culler, "Active Message Applications Programming Interface and Communication Subsystem Organization", *Technical report, Computer Science Division, University of California at Berkeley.*

[5] C. Amza, A.L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu and W. Zwaenepoel, "TreadMarks: Shared Memory Computing on Networks of Workstations", *IEEE Computer*, 29(2):18–28 (1996).

[6] T. Nanri, H. Sato and M. Shimasaki, "Implementing a Portable SPMD Shared-Memory Model Parallel Language in a Distributed Computing Environment", *Proceedings of International Symposium on Parallel and Distributed Supercomputing*, 243-252 (1995).

[7] W.Y. Liang, , C.T. King and F. Lai, "Adsmith: An Object-based Distributed Shared Memory for Network of Workstations," *IEICE Transaction on Information and Systems*, E80-D(9):899–908 (1997).

[8] M. Oguchi, H. Aica and T. Saito, "A proposition and evaluation of DSM models suitable for a widearea distributed environment realized on high performance networks," *IEICE Transactions on Communication(Japan)*, vol.E79-B, no.2, 153-162, (1996).

[9] J.B. Carter, J.K. Bennett and W. Zwaenepoel, "Implementation and Performance of Munin," *Proceedings of the Thirteenth Symposium on Operating Systems Principles*, 152-164 (1991).

[10] M. Sato, Y. Kodama, H. Sakane, S. Sakai, Y. Yamaguchi and S. Sekiguti, "Programming with Distributed Data Structure for EM-X Multiprocessor," *Theory and*

Practice of Parallel Programming, Ito and Yonezawa (Eds.), Lecture Notes in Computer Science 907, 472-483, Springer-Verlag (1995).

[11] V.W. Freeh, D.K. Lowenthal and G.R. Andrews, "Distributed Filaments: Efficient Fine-Grain Parallelism on a Cluster of Workstations." *First Symposium on Operating Systems Design and Implementation*, 201-212 (1994).

[12] K. Thitikamol and P.J. Keleher, "Per-Node Multi-Threading and Remote Latency," *IEEE Transactions on Computers* (1998).

[13] M. Sato, S. Sato, K. Kusano and Y. Tanaka, "OpenMP Design for an SMP Cluster," `http://pdplab.trc.rwcp.or.jp/pdperf/Omni/Omni-CDSM/home.html`

[14] A.L. Cox, Y.C. Hu, H. Lu, and W. Zwaenepoel, "OpenMP on Networks of SMPs," *Proceedings of the Thirteenth International Parallel Processing Symposium*, 302-310 (1999).

[15] E. Speight and J.K. Bennett, "Using Multicast and Multithreading to Reduce Communication in Software DSM Systems," *Proceedings of the Fourth Symposium on High Performance Architecture (HPCA)*, 312-323 (1998).

[16] Y. Tanaka, M. Matsuda, M. Ando, K. Kazuto and M. Sato, "COMPaS: A Pentium Pro PC-based SMP Cluster and its Experience," *IPPS Workshop on Personal Computer Based Networks of Workstations*, LNCS 1388, 486-497 (1998).