RWC PC Cluster II and SCore Cluster System Software – High Performance Linux Cluster –

Yutaka Ishikawa Hiroshi Tezuka Atsushi Hori Shinji Sumimoto
Toshiyuki Takahashi Francis O'Carroll Hiroshi Harada

Real World Computing Partnership
{ishikawa, tezuka, hori, s-sumi, tosiyuki, ocarroll, h-harada}@rwcp.or.jp
http://www.rwcp.or.jp/lab/pdslab/

ABSTRACT

The RWC PC Cluster II, consisting of 128 Intel Pentium Pro microprocessors connected by a Myricom Myrinet giga-bit network, achieves the comparable speed of a super computer such as Cray T3E. Its system software called the SCore cluster system software is built on top of Linux without any kernel modifications but adding a driver to the kernel. The SCore Cluster system software consists of the user-level communication facility called PM using a Myricom Myrinet giga-bit network, a communication library MPI on top of PM, a parallel programming language system called MPC++, and a global operating system called SCore-D.

This paper shows that a compact and well maintainable PC cluster using Linux can be built and its performance is the comparable super computer power. The key technology to realize high performance communication is introduced, i.e., so-called zero copy message transfer between nodes and one copy message transfer within a node realized in the PM kernel-level driver.

1 Introduction

Many high performance clustering research projects, using commodity hardware with high-speed networks, have been widely investigated. Our distinguished approach to realizing such a cluster system is to design and develop i) a compact and well maintainable PC-based cluster and ii) a total system software architecture on top of a commodity operating system, Linux, without any kernel modifications but adding a driver to the kernel.

The RWC PC Cluster II is the second generation of our PC cluster, which consists of 128 Intel Pentium Pro 200 MHz microprocessors connected by a Myricom Myrinet giga-bit network[7]. To make the system compact and well maintainable, we employ the PICMG PCI-ISA passive backplane standard[1].

The SCore cluster system software is our cluster system software running on top of Linux. To realize the high performance system using commodity hardware and software, the following key technologies have been employed:

- a user-level zero-copy message transfer mechanism between nodes and one copy message transfer mechanism within a node by a high performance communication facility called PM,

- a high performance MPI implementation called MPICH-PM/CLUMP that integrates both zero-copy message transfer and message passing facilities in order to maximize performance, and

- a multi-user environment using gang scheduling without degrading the communication performance realized by an operating system daemon called SCore-D.

In this paper, first, RWC PC Cluster II and the SCore cluster system are introduced to show that a compact and well maintainable PC cluster using commodity hardware can be built on top of Linux. Since the PM kernel-level driver's functionality is significant to realize high performance on top of Linux, section 3 introduces the PM kernel-level driver. The driver supports so-called zero copy message transfer between nodes and one copy message transfer within a node. Then, in section 4, the basic performance of the communication facility is evaluated. Related works are presented in section 5.

2 An Overview of RWC PC Cluster II

2.1 Hardware

The easiest way to build a PC cluster is to stack off-the-shelf PCs into a rack and connect them. This method makes possible a cheap parallel machine composed of commodity hardware components. It, however, requires a lot of space and suffers from maintenance problems.

The RWC PC Cluster II was designed to make the system compact and easily maintainable unlike the example above. We use the PCI-ISA passive backplane standard specified by the PICMG (PCI Industrial Computer Manufacturers Group) [1].

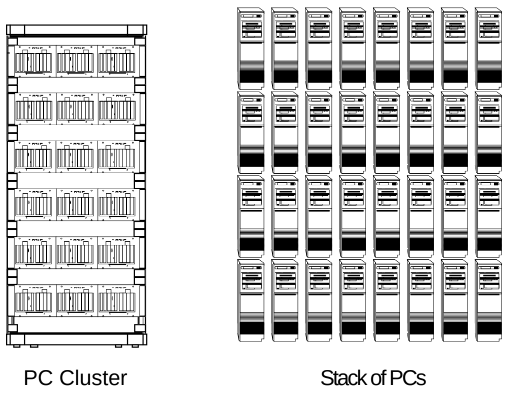


Figure 1: PC Cluster and a Stack of PCs

The PCI-ISA passive backplane does not include any active devices which are normally located on a motherboard. Instead, a PICMG processor board contains all active devices. The advantages of the passive backplane are, i) its more maintainable than a motherboard system and has a much lower mean time to repair, and ii) it is easy to upgrade to the latest processor technology. Those advantages are very useful in constructing a cluster of PCs. Figure 1 depicts comparison with our system and a stack of PCs.

Figure 2 shows node components: a PICMG Pentium Pro 200 MHz processor card, a 4 GBytes local disk, a 100 Base-T network card, and a Myrinet[7] giga-bit network card. Those cards are plugged into the PCI-ISA passive backplane. Two nodes are packed into one module. As shown in Figure 3, the RWC PC cluster II has four cabinets each of which contains 16 modules and one monitor PC. The 32 processor's serial lines are connected to one moni-

[1]PICMG is a consortium of over 350 industrial computer product vendors who collaboratively develop specifications for PCI-based systems and boards for use in industrial and telecommunications computing applications[1].

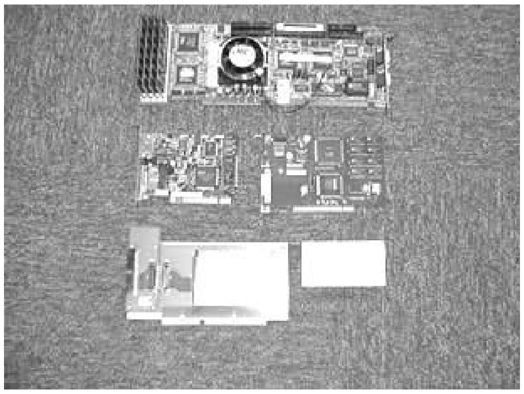
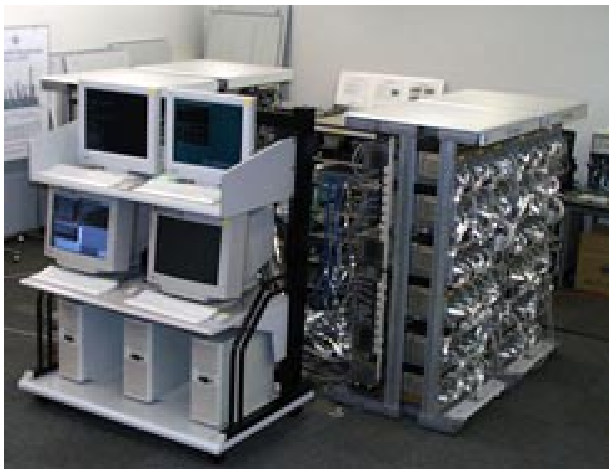


Figure 2: Components of Each Node



There are three PCs and four displays in the front of the cluster. You will imagine how RWC PC Cluster II is small.

Figure 3: RWC PC Cluster II

Table 1: RWC PC Cluster II Specification

Number of Processors	128
Processor	Pentium Pro
Clock [MHz]	200
Cache [KB]	512
Memory [MB]	256
I/O Bus	PCI
Local Disk	4GB IDE
Network	Myrinet

tor PC. Thus, the system has 128 Intel Pentium Pro processors. Table 1 summarizes the system specifications.

2.2 An Overview of SCore Cluster System Software

The high performance computing environment on the RWC PC Cluster II cannot be realized without the SCore cluster system software on top of Linux. The SCore System software consists of a communication facility called PM, MPI implemented on PM called MPICH-PM/CLUMP, a global operating system called SCore-D, and a multi-threaded programming language called MPC++.

2.2.1 PM

To achieve a low latency and high bandwidth communication, PM directly accesses the Myrinet network interface to eliminate kernel traps and data copies between kernel and user spaces[12]. PM consists of a user-level library, a kernel-level driver, and a communication protocol handler on the Myrinet network hardware.

The PM kernel-level driver initializes the Myrinet network interface and maps the SRAM of the interface to the user address space so that the PM user-level library directly accesses it. Since the PM communication facility is realized without involving a kernel routine, such a facility is called a user-level communication facility.

PM realizes not only message passing but also a remote memory write facility or so-called zero copy message transfer to provide high bandwidth communication. In zero copy message transfer, a message is transferred using the DMA facility of the Myrinet network without any memory copy operation by the host processor. Since the DMA facility accesses the physical memory address space, user virtual memory must be pinned down to a physical memory location before the message is transferred. If each message transfer involves pin-down and release kernel primitives, message transfer bandwidth will decrease since those primitives are quite expensive. We have proposed and implemented a zero copy message transfer with a *pin-down cache* technique which reuses the pinned-down area to decrease the number of calls to pin-down and release primitives[13].

The PM kernel-level driver implements pin-down and release primitives since the `mlock` and `munlock` primitives are only available under the super user mode. The PM kernel-level driver will be discussed in section 3.

2.2.2 MPICH-PM/CLUMP

When a message passing library such as MPI is implemented on top of a lower level communication facility that supports the zero copy message transfer primitive, the message passing library must handle the pinned-down memory area which is a restricted quantity resource under a paging memory system. Allocation of pinned-down memory by multiple simultaneous requests for sending and receiving without a control can cause deadlock. MPICH-PM, based on the MPICH implementation, has overcome this issue and achieves good performance[8].

MPICH-PM/CLUMP, the successor of MPICH-PM, supports a cluster of multiprocessors or called CLUMP. Using MPICH-PM/CLUMP, The MPI legacy programs run on CLUMP without any modifications. For example, suppose a CLUMP consisting of 16 nodes each of which contains dual processors. MPICH-PM/CLUMP provides an MPI application with 32 processors or 32 processes. Communication between two processes on different nodes is realized by the PM communication facility using a Myrinet network. Message transfer between two processes on one node is handled by the PM kernel-level driver so that one copy message transfer is realized. The PM kernel-level driver will be discussed in section 3.

2.2.3 SCore-D

The SCore-D global operating system is implemented as a set of daemon processes on top of a Unix operating system without any kernel modification[5]. To utilize processor resources and to enable an interactive programming environment, parallel processes are multiplexed in processors' space and time domains simultaneously under SCore-D. Parallel processes are gang-scheduled when multiplexed in the time domain. It has been proved that the SCore-D gang scheduler overhead is less than 4 % of the total application execution time[5].

To realize gang scheduling under a communication layer which accesses the network hardware directly, the network hardware status and messages inflight on the network must be saved and restored when switching to another parallel process. This mechanism is called network preemption. By co-designing PM and SCore-D, the network preemption technique has been developed.

2.2.4 MPC++

MPC++ Version 2 is designed in two levels, level 0 and level 1. Level 0, called Multi-Thread Template Library (MTTL), specifies parallel description primitives realized by the C++ template feature without any language extensions. It provides remote function invocation, synchronization structure, and remote memory access facilities[6] .

Level 1 specifies the MPC++ meta-level architecture which enables library designers to provide an optimizer specific to their class/template library in the library header file[11]. The library user may use such a high performance library by including the header file.

3 PM Kernel-level Driver

The PM kernel-level driver initializes the Myrinet network interface and maps the SRAM area of the interface to the user address space. Moreover, the driver supports pin-down and release primitives for a zero copy message transfer using the Myrinet interface and one copy message transfer mechanism for communication within a node. Those functionalities are provided by the `ioctl` command of the driver.

3.1 Pin-down/release operation

As described in section 2.2.1, a zero copy message transfer is realized by using the DMA facility of the Myrinet interface which requires the physical memory address. Linux kernel primitive `mlock` provides that user virtual memory is pinned down to a physical memory location while the `munlock` primitive is to release the pinned down area. However, those are only available under the super user mode.

The PM kernel-level driver implements the same functionality of `mlock` and `munlock` operations. The amount of total pinned-down area is restricted to each process so that the physical memory is not exhausted.

The pindown/release operations are not special requirements of PM. Rather, those are required if a zero copy message transfer is implemented using some I/O card which has a DMA facility.

3.2 1-copy message transfer support

One copy message transfer between processes on a multiprocessor might be realized using the shared memory facility. This is true if the processes explicitly allocate the shared region and accesses the region. However, a communication library such as MPI using the shared memory facility involves two times message copies since the user-level communication area is used without respect to the shared memory region. Following is the typical scenario:

1. The communication library copies the sender's data to the shared area in the sender process.

2. After the receive primitive is posted, the communication library copies data from the shared area to the receiver's memory area in the receiver process.

To realize one copy message transfer, memory write to another process's memory area or memory read from another process's memory area must be required. In the PM kernel-level driver, the process's memory read function is supported because it has less risk than the process's memory write function. The process's memory read is only available for processes that have the parent/child relation. The following is the typical scenario of one copy message transfer:

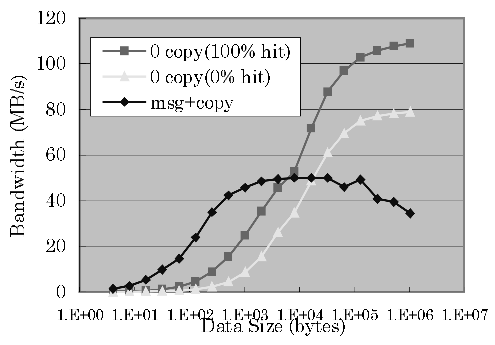


Figure 4: PM Bandwidth between Nodes

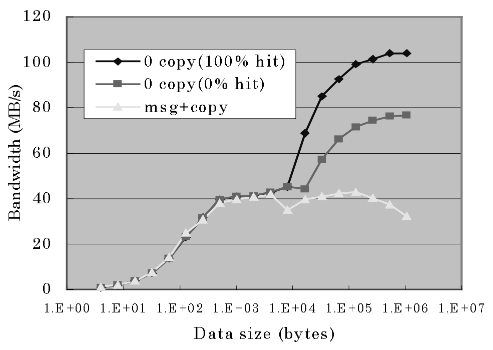


Figure 5: Bandwidth between Nodes using MPICH-PM/CLUMP

1. The communication library in the sender process informs the receiver that the send data is ready.

2. After the receive primitive is posted in the receiver process and the sender informs the data ready, the communication library on the receiver reads the send data and stores it to the receiver's memory area.

4 Evaluation

4.1 PM performance between nodes

Figure 4 shows that the PM data transfer bandwidth using the RWC PC cluster II. The maximum bandwidth is 113.5 MBytes/sec for 256 KBytes in the case of 100 % pin-down cache hit ratio. 100 % pin-down cache hit ratio means that all data area is pinned down and never calling the PM kernel-level driver to pindown an area.

In the case where the pin-down cache is always miss, the maximum bandwidth is still 78.7 MBytes/sec which is still higher than the bandwidth of data transfer with data copy, i.e., using the PM message passing facility. PM also achieves a low latency, 7.5 μ seconds one-way latency.

Figure 5 shows bandwidth between nodes using MPICH-PM/CLUMP. Since the PM message passing facility is better performance than the PM zero copy facility in case of less than 8 Kbytes message, a message of less than 8 Kbytes uses the message passing facility while a message greater than 8 Kbytes uses the zero copy facility. The performance results show that 13.16 micro seconds latency and 104 MBytes/sec maximum bandwidth is achieved.

4.2 PM performance within a SMP node

Figure 6 shows bandwidth within a node using MPICH-PM/CLUMP. This is the result on cluster of 32 Dual Intel Pentium II 330 MHz processors since a RWC PC Cluster II node is a single processor.

The figure contains the result of i) two MPICH-PM/CLUMP implmentations, 2-copies using the shared memory and 1-copy using the PM driver, and ii) MPICH 1.1 shared memory implementation called lfshmem which is included in the MPICH 1.1 distribution. The figure also includes the performance of the PM 1-copy facility.

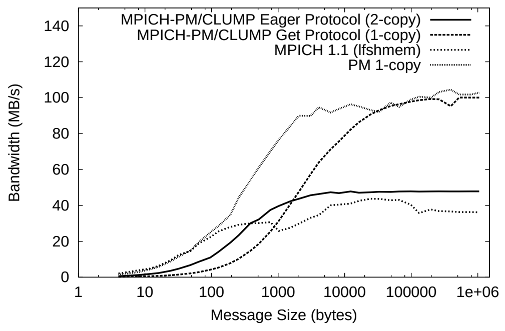


Figure 6: Bandwidth within Node using MPICH-PM/CLUMP

The actual MPICH-PM/CLUMP incorporates with two implementations, 1-copy and 2-copies, to achieve the maximum performance. Thus, the performance results show that 8.62 micro seconds latency and 100 MBytes/sec maximum bandwidth is achieved. MPICH-PM/CLUMP is better performance than the MPICH 1.1 lfshmem implementation when the message size is larger than 400 Bytes.

4.3 NAS parallel benchmarks result

NAS Parallel benchmarks have been used to evaluate the RWC PC Cluster II and compare it with other systems including Cray T3E and SGI O2K. The results show that it is the comparable performance of such supercomputers.

Due to the paper length limitation, the results cannot be presented in this paper. Visit the following URL: http://www.rwcp.or.jp/lab/pdslab/benchmarks/npb2.3/980903/

5 Related Works

Beowulf[10] is a collection of device drivers and tools for parallel programming on top of the TCP/IP protocol. Unlike the SCore cluster system software, it does not support a high performance communication library such as PM.

There are several user-level communication implementations such as Active Messages (AM)[2], Fast Messages (FM)[9], BIP[3], and U-Net[14]. There is a paper evaluating and comparing AM, FM, BIP, and PM[4]. According to the paper, PM realizes better communication functionality with good performance.

AM, FM, and BIP supports MPI implementations. MPI-AM is running on cluster of Sun machines. Both MPI-FM and MPI-BIP are running on a Linux-based cluster. As long as we know, MPICH-PM/CLUMP is only utilizing a cluster of multiprocessors without changing a legacy MPI program.

6 Concluding Remarks

This paper contributes to the Linux users to demonstrate that a high performance parallel system can be built using PCs with a Myricom myrinet network. The RWC PC Cluster II and its software environment, called the SCore Cluster system software, are an example of a compact and well maintainable PC cluster using Linux. The SCore cluster system software is also running on a Compaq Alpha 21164 processor-based cluster.

The SCore cluster system software on top of Redhat 5.1 and Redhat 5.2 is currently distributed freely via the following URL: http://www.rwcp.or.jp/lab/pdslab/dist/

The distribution includes a cookbook for building your own PC cluster, which describes an instruction to order machines and configure the system.

References

[1] http://www.picmg.com.

[2] http://now.cs.berkeley.edu/AM/lam_release.html.

[3] http://lhpca.univ-lyon1.fr/bip.html.

[4] Soichiro Araki, Angelos Bilas, Cezary Dubnicki, Jan Edler, Koichi Konishi, and James Philbin. User-space communication: A quantitative study. In *SC98: High Performance Networking and Computing Conference*, 1998.

[5] Atsushi Hori, Hiroshi Tezuka, and Yutaka Ishikawa. Highly Efficient Gang Scheduling Implementation. In *SC'98*, November 1998.

[6] Yutaka Ishikawa. Multi Thread Template Library – MPC++ Version 2.0 Level 0 Document –. Technical Report TR–96012, RWC, September 1996. *This technial report is obtained via http://www.rwcp.or.jp/lab/mpslab/mpc++/mpc++.html.*

[7] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic and Wen-King Su. "Myrinet – A Gigabit-per-Second Local-Area Network". *IEEE MICRO*, 15(1):29–36, February 1995.

[8] Francis O'Carroll, Hiroshi Tezuka, Atsushi Hori, and Yutaka Ishikawa. The Design and Implementation of Zero Copy MPI Using Commodity Hardware with a High Performance Network. In *International Conference on Supercomputing '98*, pages 243–250, July 1998.

[9] Scott Pakin, Mario Lauria and Andrew Chein. "High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet". In *Proceedings of Supercomputing '95, San Diego, California*, 1995.

[10] T. Sterling, D. J. Becker, D. Savarese, M. R. Berry, C Reschke. "Achieving a Balanced Low-Cost Architecture for Mass Strage Management through Multiple Fast Ehternet Channels on the Beowulf Parallel Workstation". In *Proceedings of the 10th International Parallel Processing Symposium*, April 1996.

[11] Tosiyuki Takahashi, Yutaka Ishikawa, Mitsuhisa Sato, and Akinori Yonezawa. A compile-time meta-level architecture supporting class specific optimization. In *Scientific Computing in Object-Oriented Parallel Environment, ISCOPE'97*, volume 1343 of *Lecture Notes in Computer Science*, pages 89–96, 1997.

[12] Hiroshi Tezuka, Atsushi Hori, Yutaka Ishikawa, and Mitsuhisa Sato. PM: An Operating System Coordinated High Performance Communication Library. In *High-Performance Computing and Networking '97*, 1997.

[13] Hiroshi Tezuka, Francis O'Carroll, Atsushi Hori, and Yutaka Ishikawa. Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication. In *IPPS/SPDP'98*, pages 308–314. IEEE, April 1998.

[14] Thorston von Eicken, Anindya Basu, and Werner Vogels. U-Net: A User Level Network Interface for Parallel and Distributed Computing. In *Fifteenth ACM Sumposium on Operating Systems Principles*, pages 40–53, 1995.