Abstract

Most interconnections and communication software of recent clusters are LAN or WAN-based which include large communication overheads, and are not optimal for parallel computing. This paper presents a design and an implementation of message passing library called MMP. MMP is designed so that the potential performance of Maestro cluster can be extracted efficiently. Maestro has a dedicated network that consists of high performance network interfaces (NI) and a switch. In this paper, we propose two optimization techniques: optimizing pipelined transfer and autonomous memory management by NI. Evaluation has been done using round-trip communication and application programs. Through the evaluation, we have confirmed that MMP utilized dedicated hardware of Maestro efficiently and achieved high performance.

1 Introduction

As the cost/performance ratio of microprocessor has been improved, PC clusters are gaining popularity in the field of parallel processing[1]. When executing parallel programs on a PC cluster, TCP/IP over Ethernet is used conventionally as their communication infrastructure. However, to guarantee the reliability of communication in WAN, TCP/IP and Ethernet have miscellaneous overheads that are not acceptable for parallel computing[2]. That is, although PC clusters are organized in LAN or SAN environments, communication is performed by using wide-area-based network hardware and protocol software, and thus, are not able to achieve low-latency/high-throughput communication needed for parallel computing.

Parallel applications require low latency communication for fast synchronizations among PCs, as well as high throughput communication for data transfer. However, the conventional protocol software includes the following overheads that deteriorate communication performance:

(1) Redundant data replication

One of the primary factors of the overhead is in data replication on PC. In the conventional protocol stack, data is copied to be passed from one protocol layer to another for several times. The protocol stack for ensuring reliability in WAN is not optimal for cluster computing.

(2) Serialized communication

Most network interface of Ethernet can not complete a transfer until entire data is migrated to the receiver’s network interface, and thus, send and receive operations are serialized. This situation prevents a network from achieving high throughput.

Thus, to realize high performance communication, it is important to reduce the number of redundant copying operations in the protocol software. Extracting parallelism in the communication path is also important. The communication path can be partitioned into multiple stages to transfer data in a pipelined manner.

In this paper, we will focus on those overheads that reside in the conventional communication protocol software, and propose new optimization methods to remedy the problems: optimizing pipelined transfer and autonomous memory management by network interface. Then, we will present a design and an implementation of message passing library, called MMP, on PC cluster Maestro. Moreover, we will evaluate the performance of MMP to show the effectiveness of the optimization techniques.

2 Overview of Maestro Network

We have developed a network hardware, called Maestro network[3], for high performance PC cluster. As illustrated in Figure 1, the Maestro network is composed of network interface cards, each adapted to a PC via its PCI bus, IEEE1394 cables, each emanating from a network interface card, and a central switch box that terminates all the cables to it and that routes messages from the source to destination network interface cards. The NI is composed of an MLC (Maestro Link Controller) for controlling physical layer, a PowerPC, DRAMs and network FIFO buffers. On the other hand, SB has eight ports connected with each NI, and is composed of a PowerPC, DRAMs, for SB interfaces and a switch controller.
Each message analyzer consists of two MLCs and an IEEE1394 physical layer and network buffers. The message analyzer passes the message header that is extracted from a message received from each port to PowerPC, and then, PowerPC analyzes it, and requests the switch controller a DMA transfer from the source to the destination network buffer by using the switching bus.

We have applied two new optimization techniques to its link layer of Maestro: packet aggregation and pipelined transfer. Partitioning each message in fine-grained packets, the former technique aggregates multiple packets best fit to the space currently made available to the receiver's buffer, and transfers them in burst to the receiver. The burst transfer of aggregated packets reduces the number of physical link arbitrations, each followed by miscellaneous device set-up operations. To avoid the serialization of communication operations on network hardware, the latter technique is partitioning each message into fine-grained packets, and has each device component pass one packet to another in a pipelined manner.

3 Message passing library: MMP

3.1 Key techniques

To reduce the overhead of conventional protocol, we have designed and implemented a message passing library, called MMP, over Maestro network. In designing MMP, two optimizations have been introduced as follows:

(1) Optimizing pipelined transfer

Copy operations among different hardware components along the communication path are overlapped to improve overall throughput. For example, (i) the copy operations from sender’s PC to its NI, (ii) the one from sender’s NI to receiver’s NI, and (iii) the one from receiver’s NI to its PC can be overlapped. In organizing the above pipeline, we have optimized the grain size of data transferred in each copy operation.

(2) Autonomous memory management by NI

This optimization allows the NI to manage communication buffer area on PC’s memory. Usually, allocating and freeing receiving buffer is performed by the PC. This memory management disturbs computation and increases latency. By leaving it to NI, a part of the receiving procedure can be overlapped with computation.

3.2 Communication flow

This section describes a communication flow example of MMP. It is assumed that a message is transferred between application programs on PC0 (sender) and PC1 (receiver) as shown in Figure 2.

First, with splitting a message into fragments, the sender allocates areas for each fragment in reserved memory(Figure 2(a)). Reserved memory is reserved on starting up the OS[4] to be accessed by NI directly. Then, after copying all fragments of the message into the allocated area, the sender writes send requests for fragments into the send request buffer (Figure 2(b)). Here, the grain size of the fragments has been determined experimentally. When the NI receives the send request, it transfers each fragment from the reserved memory to the network buffer(Figure 2(c)).

On receiver side, when the NI detects arrival of a fragment in the network buffer, it allocates an area for the fragment in the reserved memory and transfers the fragment into the allocated area(Figure 2(d)). Then, the NI writes the receive request with respect to the fragment(Figure 2(e)). The receiver’s application program receives the request and copies the fragment from the reserved memory(Figure 2(f)). Repeating these receive operations, the receiver reconstructs the whole message.

The above transfers ((a), (c), (d) and (f)) are performed in a pipelined manner by adjusting the fragment size optimally.
Moreover, on the receiver side, the autonomous allocation of reserved memory by NI can avoid serializing computation and communication.

4 Performance evaluation

Performance evaluation of MMP has been done using round-trip communication and application programs. The basic performance, the effectiveness of pipelined transfer, and the effectiveness of overlapped communication with computation will be shown and discussed. In this evaluation, eight PCs (Celeron 400MHz) that are connected with Maestro network are used.

4.1 Basic Performance

According to the results of round-trip communication shown in Figure 3, the minimal latency is 63 usec in case of 4byte transfer. The maximal throughput is 96% of the peak performance of Maestro network. This indicates that the overhead of MMP is very small and MMP can extract the maximum performance available in Maestro network.

4.2 Effects of fragment size

To evaluate the effects of optimizing pipelined transfer, we measured the performance with varying the size of fragment.

Figure 4 shows the comparison of round-trip communication performance for the fragment size of 223bytes, 4048bytes, and 104852bytes.

As seen in the figure, for the fragment size is 223bytes and 104852bytes, the throughput is 65% and 80% respectively. The throughput reaches up to 96% when the fragment size is 4048bytes. Therefore, we found the optimal fragment size in our environment is 4048byte.
4.3 Evaluation using applications

Using IS and CG from NAS parallel benchmarks[5], we evaluate the effect of autonomous memory management by NI and the overall performance of MMP. The problem size of benchmark is class A. We have developed MPI library using MMP since these benchmarks are written using MPI.

We compare the performance of MMP over Maestro with TCP/IP over 100BASE-TX Ethernet.

The left-hand side of Figure 5 shows the result of IS. IS is an application that performs radix sorting of integers. Using different data for each iteration, the sorting procedure is repeated 10 times. Each iteration includes two all-to-all communication that exchange large data. Because the memory bandwidth is larger than the network bandwidth, the single processor executes faster than two processors. However, when the number of processor is eight, the performance of MMP over Maestro is 1.4 times better than the one of TCP/IP over Ethernet. This is due to the optimized pipelined transfer, which is efficient in transferring large data.

The right-hand side of Figure 5 shows the result of CG. CG solves an unstructured sparse linear system by conjugate gradient method that includes a large number of butterfly communications. Using non-blocking communication to transfer small data, this application program tries to hide communication overhead. This implies the performance of CG is very sensitive to communication overhead. From the results, MMP achieves better speedup than the one of TCP/IP over Ethernet. This performance gain results from the autonomous memory management by NI, and thus, the proposed technique hide the overhead in transferring small data efficiently.

5 Conclusion

In this paper, design and implementation of message passing library, called MMP, has been described. MMP is so designed as to extract the potential performance of Maestro cluster. We pointed out the overheads in the conventional message passing library, and propose two optimization techniques: optimizing pipelined transfer and autonomous memory management by NI.

This paper also shows performance evaluation of the proposed techniques using round-trip messaging and application programs. From the evaluation, we confirmed that MMP could provide 96% of available throughput of Maestro network, and exhibited excellent communication efficiency compared to the conventional TCP/IP over Ethernet.

For the future works, we plan to implement and evaluate MMP over Myrinet, Gigabit Ethernet, and other high performance networks. In addition, we will develop a system that automatically decides the suitable grain size for optimized pipelined transfer.

References