SHARED MEMORY ACCESS METHOD FOR A λ COMPUTING ENVIRONMENT

Hirohisa Nakamoto,[1] Ken-ichi Baba,[2] and Masayuki Murata[1]
[1] *Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565-0871, Japan*
[2] *Cybermedia Center, Osaka University, Ibaraki, Osaka 567-0047, Japan*

Abstract: Although optical transmission technology for high-speed broadband networks is being studied actively, the conventional packet-based switching technology cannot assure high-quality communication for each connection. We therefore propose a new computing environment, called λ computing environment, that provides virtual channels utilizing optical wavelength paths connecting computing nodes. These channels provide the reliable high-speed paths needed for recently deployed distributed applications including SAN and Grid computing. In this paper, we propose and evaluate a method for accessing the virtual ring network for realizing a shared memory in a distributed fashion on WDM-based photonic networks.

1. INTRODUCTION

In recent years, as users of networks such as the internet are increasing, the amount of traffic is increasing steadily. Various applications utilizing images become to be used, and the demand on the technology which enables the high speed and large scale transmission in networks is increasing. Research on optical transmission technology has therefore been expanded in efforts to develop WDM technologies, which use light of various wavelengths. Investigators are actually exploring the feasibility of a new WDM technology that can use 1000 wavelengths [1]. IP over WDM networks are being studied in order to provide high-speed Internet transmission based on WDM technology. An Internet routing technology called Generalized Multi-Protocol Label Switching (GMPLS) is also being standardized [2], and research on the optical packet switch has begun [3].

Many of these technologies, however, presuppose the present Internet technology. That is, the granularity of information is assumed to be the IP packet.

And the assumption of an architecture based on packet switching technology makes it very hard to provide high-quality communication to each connection. New application technologies such as SAN and Grid computing need to provide the end user with a high-speed and reliable communication pipe, and that cannot be done without setting up a mass wavelength path between end users. That is, the end users can be provided an ultrahigh-speed and ultrahigh-quality communication pipe by building a photonic network that uses established fibers, or newly laid fibers if needed, and by using the wavelength multiplexed in the fiber as the minimum particle size for information exchanges.

OptIPuter is middleware proposed for the high-speed distributed computation environment provided by an optical network [4]. It was developed for the Grid environment to be established on optical networks. It provides virtual communication paths but is based on the present Internet technology and treats a packet as the particle size of information. As a result, the packet-processing problem mentioned previously arises.

We therefore propose a new architecture, called λ computing environment, that has virtual channels utilizing wavelength paths on WDM-based photonic networks for connecting computing nodes. In the conventional Grid environment, data is exchanged by TCP/IP message-passing. On the other hand, The λ computing environment provides reliable high-speed communication between nodes on the Grid by using established wavelength paths. Distributed computation on high-speed channels is thus provided by making virtual channels in the mesh of a photonic network of optical fibers connecting the network nodes and the computing nodes. Moreover, it is possible to use wavelengths as a shared memory by constituting a virtual ring in the λ computing environment. As a result, it is not necessary to distinguish the shared memory from a communication channel in a wide-area distributed system. We expect this to result in high-speed data exchange between computing nodes (see Fig. 1).

In this paper, we propose and evaluate a method for accessing the virtual ring network utilized as a shared memory. Specifically, we propose that the cache in the CPU and local memory of each computer group are used as a cache of a shared memory. When a virtual ring is used as a shared memory, it is necessary to consider restrictions on the timing and frequency of access because the shared memory is spread out on a long-distance optical fiber and also to take into consideration the coherency between the shared memory in the virtual ring and the cache of each computer group. It is not necessary, however, to distinguish the shared memory in a wide-area distributed system from a communication channel, and this seems to be what makes it possible to exchange data between computing nodes at extremely high speeds. In consideration of such features, we propose a shared memory access method for the λ computing environment and evaluate the method through simulations.

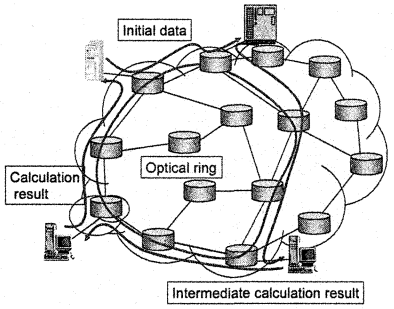

Figure 1. Virtual ring on a photonic network.

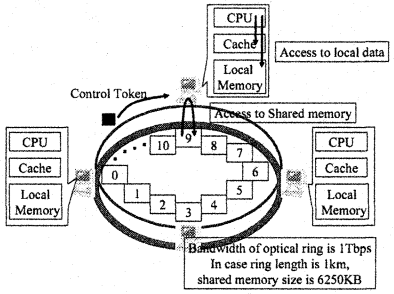

Figure 2. Network model.

2. SHARED MEMORY AND ACCESS METHOD IN A λ COMPUTING ENVIRONMENT

2.1 Network model

In the network model we use (see Fig. 2), the computing nodes constituting the λ computing environment are connected by optical fibers that make a ring network virtually. This paper presupposes that each computing node has one CPU, a level 1 cache, and a local memory. A local memory is used for storing programming codes, and a shared memory is used for storing the shared data used by all the computing nodes.

The optical ring network has a wavelength path for shared memory and a wavelength path for control signals. The bandwidth of the wavelength path for the shared memory is 10 Tbps and the propagation delay is 5 nano-second per meter. The processing delay in the intermediate nodes, such as network devices that are part of the optical ring network, is not taken into consideration here but is simply included in the propagation delay. Therefore, when this optical ring network is used as a shared memory, its capacity is 6250 KBytes per kilometer of fiber length.

2.2 Shared memory access method

In conventional shared memory systems, each processor's access to the shared memory is restricted by contention in a shared bus. Access to the shared memory is therefore usually improved by using a cache to increase the speed of access. Every processor usually has a cache system consisting of a level 1 cache, a level 2 cache, a level 3 cache, and many stages. When each processor has a cache, it is necessary to fully take into consideration the consistency between the data in those caches and the data in the shared memory [5, 6].

In the λ computing environment, the application program calculates intensively within a computing node. And it exchanges data among computing nodes after synchronous process. So it does not have to exchange data during the processing going on within a computing node. We therefore use the write-back invalidation protocol because the number of write-back accesses to the shared memory is smaller for this protocol than for the any of the snoop cache protocols. Considering the restrictions on memory access timing and cache coherency, we propose the protocol described in Sec. 2.2.1. The cache coherency problem arising when two or more computing nodes synchronously update the same local cache data is solved by preparing a token for control messages. Parallel computers also collaborate by using such synchronous operations as atomic operation, caching of synchronous variable, waiting on a shared memory, a memory lock, the barrier synchronization method, and so on [5, 6]. Application programs evaluating a proposed method first calculate locally and then perform synchronous operations. So in the work reported in this paper we used the barrier synchronization method. The barrier synchronization method on an optical ring is explained in Sec. 2.2.2.

2.2.1 Write-back invalidation protocol for an optical ring network.

As mentioned above, we use the write-back invalidation protocol to solve the cache coherency problem. In adapting this protocol to our network model, we must take a care of control messages, the read-miss process, and the write-miss process.

In the conventional method, when a read miss occurs and the other computing node has relevant data, that node will send the data to the demanding computing node. In the shared memory using an optical ring network, however, the demanding computing node can directly access the shared memory. This is because the delay for direct access to the shared memory is shorter than the waiting for transmission of relevant data from another computing node. In the write-back invalidation protocol, the data in the local cache has three possible states: Invalid (I), Clean (C), and Dirty (D). When a copy demand message arrives and the relevant data is in state C, none of the computing nodes returns a response message. When the relevant data is in state D, in contrast, the node returns a response message. In this case, it is not necessary to access a shared memory.

We next consider the processing for the write-out from a processor to a cache. When data in the cache is in state C and a processor writes out the data to local cache, the data will change from state C to state D. Then the invalid demand message for relevant data is added to the control token, and the token is sent out to the wavelength path for control. At this time, other computing nodes with the data of a corresponding address know that the relevant data has been written out, and they change the state of the data that they have in the

self-cache into state I. If two or more computing nodes write out the C state data at the same address simultaneously, two or more computing nodes may hold the data in state D. This problem is solved by having a computing node that needs to update the cache data from state C to state D first catch the control token. After catching the control token, it checks that another computing node has not added the invalidation message to a relevant address. After checking the control token, it adds the invalidation demand message to the control token, and sends the control token to wavelength for control. It can then change the state of the cache of relevant data from state C to state D. Moreover, when the control token is caught and another computing node has already added the invalidation message to a relevant address, the state of the relevant caches is changed to state I state, and the cache update is yielded to another computing node.

2.2.2 Barrier synchronization. Barrier synchronization [5, 6] in the shared memory on the optical ring network is done by first allocating part of the shared memory to a synchronous memory area. When a synchronous memory is accessed, a Fetch&Decrement operation like that in the conventional method is performed indivisibly. That is, it ensures that access to a synchronous memory indivisibly causes subtraction processing to the relevant data. Since only one computing node at a time can access the synchronous memory, the execution of an atomic operation is easy when an optical ring network is used as a synchronous memory. When an application program is establishing synchronization among some computing nodes, each node accesses the synchronous memory. The number of processors is stored there. The value of the synchronous memory will be set to 0 if all nodes access it. If the value of a synchronous memory is set to 0, all nodes will finish their synchronous processing and begin the next processing.

3. PERFORMANCE EVALUATION

In this section, we report our evaluation of the performance of the proposed shared memory access method by using simulation. We show the results of using not only the shared memory in the λ computing environment but also results of using the conventional TCP for the distributed computation. We show them here so they can be easily compared. When coding our simulation program, we referred to the ISIS library [7] currently being developed at the Amano Laboratory at Keio University.

3.1 Simulation model

We used the following network model in our simulation. Each computing node in the λ computing environment is interconnected with optical fiber and

configures the ring network virtually. Each computing node has one CPU, a level 1 cache, and a local memory. The CPU clock frequency is 3 GHz, the capacity of a level 1 cache is 512 KB, and capacity of a local memory is 2G Bytes. The computing nodes are assumed to be equally spaced around the optical ring network. An optical ring network has a wavelength path for shared memory and a wavelength path for control signals. The bandwidth of the wavelength path for a shared memory is 10 Tbps, and propagation delay time is 5 nano-second per meter. The processing delay in the interface of each computing node and the intermediate nodes is not taken into account here but is assumed to be included to the propagation delay. We also assume that the shared memory system exchanges data according to the TCP. This system has one shared memory server and each computing node is connected to that shared memory server by an Ethernet. The performance of a computing node is the same as that in a photonic shared memory model, and the distance from each of the computing nodes to a shared memory server is set to 1 kilometer. The Ethernet transmission speed is set to 1G bps.

We evaluated the performance by using some of the Splash2 benchmark programs, such as the radix sort program that sorts the sequence of an integer value by using the radix-sort algorithm, the matrix-product program that calculates the product of an $n \times n$ matrix, and the queen-problem program that solves the n-queen problem.

3.2 Result of the radix-sort program

We first show in Fig. 3 the number of CPU execution clocks when we apply a radix-sort program to the simulator of the shared memory model in the λ computing environment. The numbers of keys for sorting are 32768, 65536 and 131072. Even if it increases the number of computing nodes in the case of the problem size 32768, parallel processing is not taken advantage of because the number of synchronous operations is a large fraction of the total number of operations. When problem size becomes large, however, like 65536 or 131072, parallel processing is advantageous whenever there are fewer than eight computing nodes. The effect of parallel processing becomes weaker, however, as the number of nodes increases. We show in Fig. 4 the number of CPU execution clocks when we use radix-sort programs in the simulator of the TCP message passing model. Although the same tendency as that seen in the shared memory simulator in the λ computing environment is shown, the number of execution clocks is, on the whole, larger than that in the shared memory simulator in the λ computing environment. From these results we found that the shared memory and the access method for the λ computing environment have advantages when performing distributed processing of a problem of sufficient size.

3.3 Result of the matrix-product program

We show in Fig. 5 the number of CPU execution clocks when we apply a matrix-product program to the simulator of the shared memory model in the λ computing environment. The matrix sizes are from 32×32 to 256×256. When the problem size is small, parallel processing is not taken advantage of. But when problem size becomes large, the effect of parallel processing is evident when the number of computing nodes is less than eight. Though we cannot show the graph of a TCP model because of lack of space, even if problem size becomes large, parallel processing is not taken advantage of. This is unlike what we see in the results we obtained with the radix-sort program. From these results, we can see that when performing distributed processing of a problem of sufficient size, the shared memory method for the λ computing environment performs better than the method for TCP message-passing does.

3.4 Result of the queen-problem program

We show in Fig. 6 the number of CPU execution clocks when we apply a queen-problem program to the simulator of the shared memory model in the λ computing environment. The problem sizes are from 4×4 to 32×32. In the case of a queen problem, there is no advantage of parallel processing even when the problem size is large. This is because the number of synchronous accesses is larger than in the other application programs, though we cannot show the graph because of lack of space.

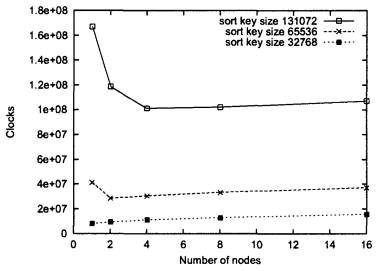

Figure 3. Processing time for a radix-sort program using photonic shared memory.

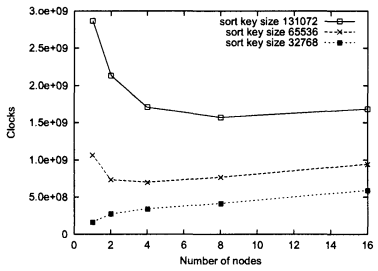

Figure 4. Processing time for a radix-sort program using TCP message passing.

4. CONCLUSION

In this paper, we proposed a method for accessing shared memory on a photonic network. We also evaluated the performance of the proposed method by

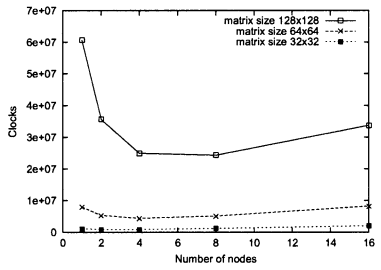

Figure 5. Processing time for a matrix-product program using photonic shared memory.

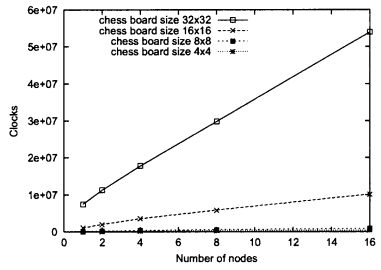

Figure 6. Processing time for a queen problem program using photonic shared memory.

using benchmark programs for parallel computing. As a result, we showed that the effectiveness of using an optical ring as a shared memory and of parallel processing implemented by increasing in the number of nodes is seen when the number of synchronous processing is small. A more efficient method for accessing a shared memory and a practical use of a local memory should be considered. Furthermore, since neither the processing delay due to the interface nor simultaneous access to a photonic ring network from two or more node computers was taken into consideration in the work reported here, these are subjects that should be investigated. We are also going to examine the shared memory system and the memory access technique at the time of using an optical ring network as a high-speed channel.

REFERENCES

[1] M. Murata and K. Kitayama, "Ultrafast photonic label switch for asynchronous packets of variable length," in *IEEE INFOCOM 2002*, June 2002.

[2] E. L. Berger, "Generalized multi-protocol label switching (GMPLS) signaling functional description," in *IETF RFC3471*, Jan. 2003.

[3] T. Yamaguchi, K. Baba, M. Murata, and K. Kitayama, "Scheduling algorithm with consideration to void space reduction in photonic packet switch," *IEICE Transactions on Communications*, vol. E86-B, pp. 2310–2318, Aug. 2003.

[4] T. DeFanti, M. Brown, J. Leigh, O. Yu, E. He, J. Mambretti, D. Lillethun, and J. Weinberger, "Optical Switching Middleware for the OptIPuter," *IEICE Transaction on Communication*, vol. E86-B, Aug. 2003.

[5] H. Amano, *Parallel Computer*. Shoukoudou, June 1996.

[6] N. Suzuki, S. Shimizu, and N. Yamanouchi, *An Implemantation of a Shared Memory Multiprocessor*. Koronasha, Mar. 1993.

[7] M. Wakabayashi and H. Amano, "Environment for multiprocessor simulator development," *I-SPAN 2000*, pp. 64–71, Dec. 2000.