Implementation and Evaluation of MPI Library with Globus Toolkit for Establishing λ Computing Environment

Mai IMOTO†, Eiji TANIGUCHI†, Ken-ichi BABA††, and Masayuki MURATA†

†Graduate School of Information Science and Technology, Osaka University 1-5 Yamadaoka, Suita, Osaka, Japan
††Cybermedia Center, Osaka University 5-1 Mihogaoka, Ibaraki, Osaka, Japan
E-mail: †{m-imoto, e-tanigu, murata}@ist.osaka-u.ac.jp, ††baba@cmc.osaka-u.ac.jp

abstract We propose a new architecture, the λ computing environment, that accomplishes high speed data transmission over optical fibers. In the λ computing environment, we can achieve highly reliable high-speed communication by establishing wavelength paths. In this paper, we use the Globus Toolkit to build a Grid environment and establish the λ computing environment using the AWG–STAR system. Moreover, we implement the MPI library using services of the Globus Toolkit and the shared memory of the AWG-STAR system to exchange data, and evaluate performance of distributed computation in a λ computing environment. Our results show that the performance depends on the number of accesses to the shared memory and the size of exchanged data.

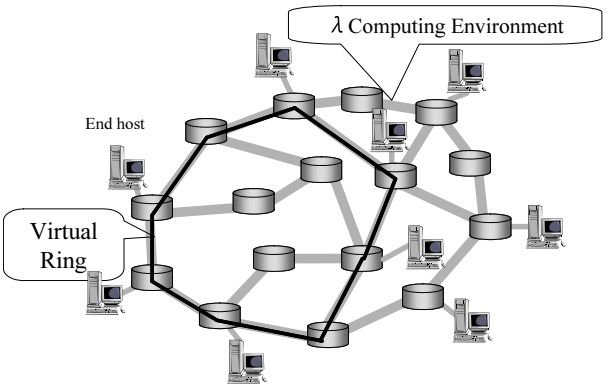

Figure 1: λ computing environment.

1 Introduction

The Grid technology has been studied by many researchers to perform large-scale scientific computation. The Grid technology virtually enables large scale computing with connecting distributed computing nodes, storage and all kinds of devices. This virtual computer is expected to have high performance computation which single computer cannot achieve, to perform large-scale scientific computation, to push huge data made by high performance observation equipments, and to visualize large amount of data by rapid real-time calculation in parallel.

In order to realize the Grid computing, one of the most important issues is how to share and cooperate geographically/systematically distributed resources. To solve that problem, some middlewares are necessary. Nowadays, Globus Toolkit developed by Globus Alliance is a de-facto standard for the middleware of the Grid computing [1].

At the same time, researches into the optical transmission technology as a lower layer of the Grid have been also actively pursued. Researches into WDM technologies that use multiplexed light wavelength have been the main target for development, and then technologies from new WDM research that can use 1000 wavelength has also been advanced [2]. In recent years, IP over WDM network has been studied and developed to provide high-speed transmission on the Internet based on WDM technology. Moreover, standardization of the routing technology of the Internet, called GMPLS [3], which is the communication technology that uses various optical technologies for a lower layer, has also been advanced in IEIF. Further, aiming to realize the true IP communication of a photonic network, researches into optical packet switches based on the optical technology have also begun [4-8]. However, many such technologies presuppose the existing Internet technology. That is, an IP packet is treated as a information units, and the target for research and development is how to carry IP packets at high speed on a network. Therefore we cannot realize high-speed and high-reliable communication as long as we use the architecture based on a packet exchange technology.

Then we propose a new architecture which we call the λ computing environment that has virtual channels utilizing optical fibers connecting computing nodes [9, 10]. In the conventional Grid environment, data are exchanged with the message passing using TCP/IP. In the λ computing environment, communication between nodes on the Grid not by conventional TCP/IP but by established wavelength paths is realized, so that we can achieve highly reliable high-speed communication (see Figure 1).

Our target in this paper is to implement high-speed distributed computation environment by adopting the Globus Toolkit into the λ computing environment. That is to say, when the λ computing environment is adopted in a lower layer of the Globus

Toolkit, users can utilize the high-speed distributed computation environment without changing their conventional programs. In concrete terms, users execute MPI Applications on the Globus Toolkit, which are actually communicated in the λ computing environment. In addition, we propose and implement methods of access to the shared memory which each node has. As an instance of the λ computing environment, we utilize the AWG–STAR system developed by NTT Photonics Laboratory [11, 12]. AWG–STAR system is one of instances that realize High-Speed Channel Architecture, and is the information sharing network system realized by data transmission using WDM technology and wavelength routing using AWG (Array Waveguide Grating) router.

In this paper, we establish the λ computing environment utilizing AWG–STAR system, implement an MPI library on the Globus Toolkit, and evaluate the performances of the distributed computation.

The rest of the paper is organized as follows. In section 2, we describe the overview of the AWG–STAR system. In section 3, we explain the implementation of MPI library in the λ computing environment with the AWG–STAR system, and in section 4, we evaluate our approach by using MPI benchmark program. Finally, we conclude the paper and describe future work in section 5.

2 AWG–STAR system

In this paper, we utilize the AWG–STAR system as an instance of λ computing environment. In this section, we explain that system.

2.1 Brief overview of the AWG–STAR system

The AWG–STAR system is an information sharing network platform realized by the WDM technology and wavelength routing using the AWG routers. Computing nodes connected to the AWG router configure a star topology in physical, but does a ring in logical (see Figure 2). The AWG router processes optical signal without transforming into electrical signal, which provides high-speed transmission. Each node is equipped with a shared memory board (SMB). It has shared memory that can contain the identical data at the same address over all nodes by the AWG–STAR system. While conventional systems need apparent instructions to transmit, data in this system are automatically sent to the optical ring network when they are written on the shared memory, and the data on the other SMB of all computing nodes are updated in real time. Furthermore, to read the data from the shared memory only needs to access their own SMB. This system achieves high-speed data sharing because it runs in the background at hardware level.

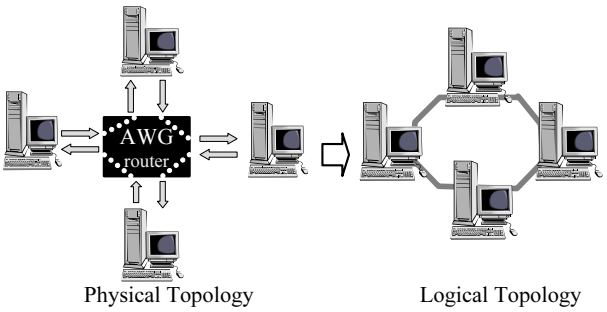

Figure 2: Topology of the AWG–STAR system.

Table 1: Specification of the SMB.

Transmission speed of the optical ring	2.152Gbps
Every transmission data size	1KByte
Processing time of frame transmission	500ns
Transmission rate to SMB	64MBytes/s
Transmission rate from SMB	80MBytes/s

2.2 Access Delay Time of AWG–STAR system

An access delay time from CPU to the shared memory is slower than that to the local memory because SMB is equipped with computing nodes via a PCI bus. It may largely affect the entire performance. Also, data have to go around the optical ring to be updated on all SMB. Table 1 shows the specification of the SMB and the speed of access via PCI bus.

2.3 Application of AWG–STAR system to the λ computing environment

There may be two models to realize the λ computing environment. We utilize the optical ring network as a shared memory, or as a high-speed channel. In the first case, shared variables in a distributed-parallel program are stored on the optical ring as the shared memory, and each node reads/writes the shared data from the shared memory. Since the data are shared by all computing nodes, it takes one-round time to update the data and the data sharing delay is inevitable. In another case, variables in a program are stored not in the shared memory but in the local memory of each computing node. The computing nodes calculate separately, and they send the data to the other by writing on the shared memory.

In this paper we adopt the latter model with the AWG–STAR system, because it has higher affinity with MPI and can shorten the delay.

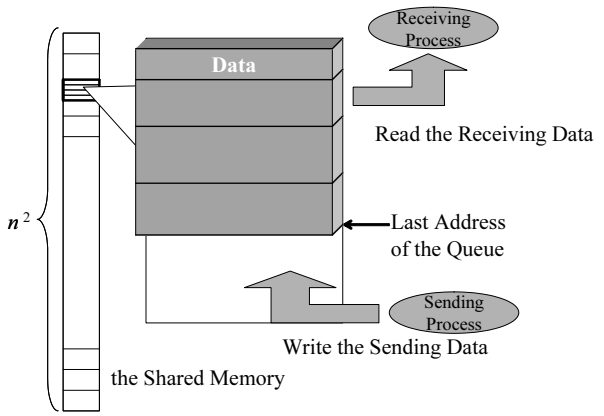

Figure 3: Utilization of the shared memory.

3 Proposed method of Message-Passing utilizing the Shared Memory

3.1 MPI Library: MPICH–G2

MPI (Message-Passing Interface) is a library specification for message-passing via network, sharing data and synchronization of processes. There are many implementations of MPI, which one of those is MPICH–G2, grid-enabled MPI [13]. MPICH–G2 works on the Globus Toolkit and uses API for communication and job management. To execute MPI application in the λ computing environment, we implement an MPI library using the Globus Toolkit for job management and the AWG–STAR system for message-passing, drawing upon MPICH–G2. We use the term "process" to refer to a process run by MPI application, and "sending (receiving) process" to a MPI process which sends (receives) data.

3.2 Proposed method for implementation of MPI library utilizing the AWG–STAR system

To realize message-passing using the AWG–STAR system, queues to send and receive data are prepared on the shared memory. To be specified, let the number of processes be n, the shared memory is divided into n^2 areas and each process utilizes each area as a queue, which is for transmission between a pair of sending and receiving processes. That is to say, when process rank i sends data to process rank j, the sending process writes the data on the area $[i, j]$. When transmitting, sending data is written on the next address where previous data is located, and the last address of every queue is stored in the local memory of each computing node. On the other hand, the receiving process reads new data from the shared memory to the local memory. At this time, to notify the receiving process that the sending process wrote data on the shared memory, we use a signal function provided by the AWG–STAR system. It means that after writing the data on the shared memory, the sending process sends a signal to the receiving process. The process which received the signal reads the data from the shared memory. Figure 3 shows the utilization of the shared memory. By implementing queues on the shared memory, we can enable to combine the operations of both the local memory and the network socket which are apart in MPICH–G2.

Also, drawing upon MPICH–G2, we implement two queues called "posted queue" and "unexpected queue" on the local memory of each process, because the timing when the process receives the data and when the receiving function is called may be different. That is to say, in the case that the process calls receiving function, but that the data to be received are not on the shared memory yet, it has to buffer the reception request until the data are actually received. Where the reception request is stored is "posted queue". In a similar way, in the case that the process receives the data from the shared memory, but that the function to receive the data is not yet called, it has to buffer the data until the function is called, so the data go to "Unexpected queue". Consistency between receiving data and requests are determined with the header of them.

3.3 Comparison between MPICH–G2 and proposed method

Figure 4 shows MPI application execution methods with both MPICH–G2 and AWG–STAR system. In both methods, the authentication of the computing nodes and the job execution request for processes are communicated with API of the Globus Toolkit. Message-passing during MPI application of MPICH–G2 also utilize API of the Globus Toolkit, that is, data are copied from the local memory to the socket buffer by system call of OS, then transmitted from the socket buffer to the memory on NIC, next divided into MAC frames, and lastly transmitted through Ethernet. On the other hand, in the method of the AWG–STAR system, data are copied from the local memory to the shared memory via PCI bus, transmitted to the network. We consider that communication by the AWG–STAR system has less overhead of the memory copy and the packet generation.

4 Evaluation

In this section, we evaluate the proposed method to access the shared memory system by executing Himeno Benchmark program, one of the MPI application benchmarks [14]. We assume that every

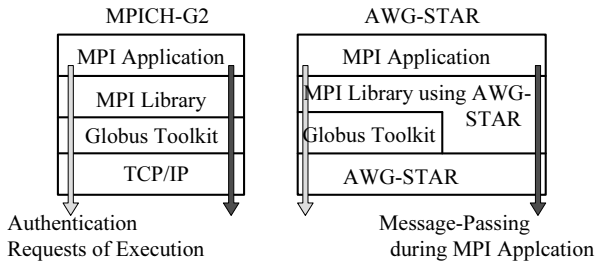

Figure 4: Comparison between MPICH–G2 and our method.

Table 2: The Specification of computing nodes.

CPU	Xeon 2.80 GHz
Main memory	SDRAM 512MB
Primary cache	512KB
Secondary cache	512KB
NIC	Intel PRO/100MT
PCI bus	64 bit/66MHz
PCI transmission speed	533MBytes/sec
OS	Redhat Linux 7.3
Compiler	gcc 2.96
Middleware of the Grid environment	Globus Toolkit 2.4

computing node executes one MPI process, that is because the SMB of the current AWG–STAR system can not be shared with the multiple processes or applications. For comparison purpose, we also show the cases for MPICH–G2.

4.1 Environment for Evaluation

Table 2 shows the specifications of the computing node for evaluation. The numbers of the computing nodes are from one to four, and all nodes are the same specifications. In this evaluation, we change the length of the optical ring depending on the number of the computing nodes. That is to say, let the number of the computing nodes be N, the length of the optical ring is $10Nm$. In the case for MPICH-G2, we use 1Gbps Ethernet and the length of each Ethernet cable is 10 m.

4.2 Application program

We use a benchmark program to evaluate the performance of our method. One of such programs is Himeno Benchmark program. Himeno Benchmark program was developed by Dr. Ryutaro Himeno in the Institute of Physical and Chemical Research to evaluate the performance of incompressible fluid analysis code on various computers. The code solves Poisson's equation by the Jacobi iterative method. The benchmark measures the computational time of the main loop in the code and reports the corresponding performance in MFLOPS. The Jacobi method can be parallelized by domain decomposition which partitions a three dimensional array into smaller arrays and assigns every smaller array to a processor. Parallel version of Himeno Benchmark program with MPI is also provided. The problem size of original Himeno Benchmark program is fixed to small, medium, large, or extra large. However, we change the problem size to some other size.

4.3 Evaluation by Comparison to MPICH–G2

We compare the performance of computation utilizing the AWG–STAR system with MPICH–G2. Figure 5 shows the performance comparison executed with two processes and figure 6 shows with four. The Y axis represents performance in MFLOPS and the X axis represents the size of 3-dimensional array. The data size of one message-passing is proportional to the array size, and the number of message-passings is in inverse proportion. Also, figure 7 shows the ratio of the calculating-time and the communicating-time among whole the processing-time. The Y axis represents the ratio of time, and X axis represents the array size.

As shown in figures 5 and 6, when the array size is small, the performance is rather low. The reason is due to the delay time of the shared memory, because the small size problem causes many accesses to the shared memory and that is the bottleneck. This is also shown in figure 7, that the communicating-time takes a fair percentage of the processing-time. As the array size increases and the number of accesses decreases, the delay time becomes shorter and the performances become better. Compared to the MPICH–G2, however, the performance of the AWG–STAR system is much lower, so the improvement of the access speed from CPU to the SMB is needed.

4.4 Evaluation by the number of processes

Figure 8 shows the performance of the same array size executed by the different numbers of processes. At first we compare with one process and two processes. Performance is higher by one processes when array size is smaller than 129×129×257. This is because no message-passing occurs by one process, so there is no overhead of message-passing. But as the array size becomes larger, differences get smaller, and when the size is 161×161×321, two processes compute faster than one process. This is because as the size becomes larger, the number of message-passing and its overhead decrease. Secondary we compare with two processes and four processes. Per-

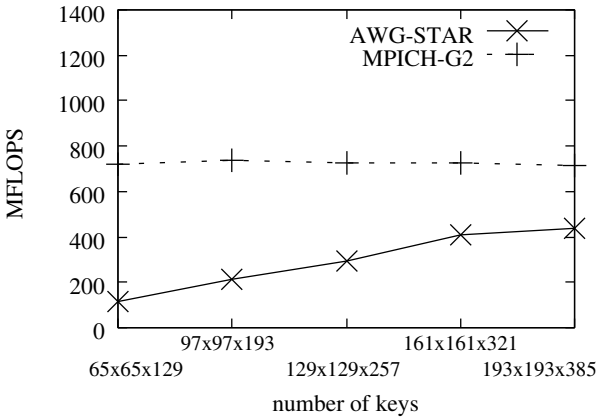

Figure 5: Performance with two processes.

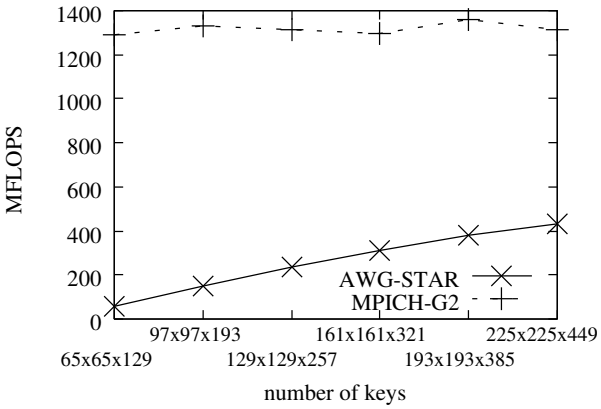

Figure 6: Performance with four processes.

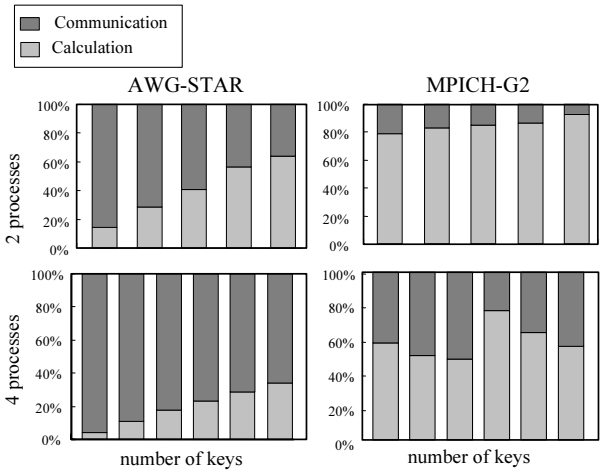

Figure 7: Time ratio of calculation and communication.

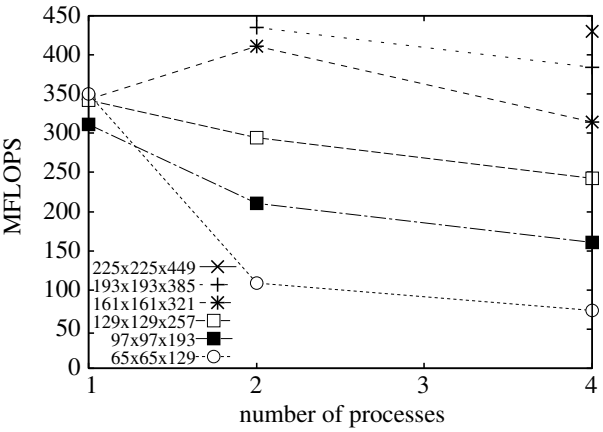

Figure 8: Comparison with the number of the processes.

formance is higher by two processes. This is also caused by the overhead of message-passing.

In the case when the array size is larger than 193×193×385, application can not be executed by one process because of too large size. In the same reason, two processes can not calculate the problem of array size 225×225×449. That is, large size problems which can not be performed by single process can be solved by parallelizing and distributing with multiple nodes.

4.5 Evaluation by the Decomposition

In Himeno Benchmark program, the problem is parallelized by domain decomposition, and a process transfers the data to the another which share the boundary region. The size of the data is almost proportional to the size of the boundary region. It means that, in the case when the domain decomposition is $1 \times 4 \times 1$, one process transmit twice the size of the case when that is $1 \times 1 \times 4$ (see figure 9). Figure 10 shows the performance of the program of which domain decomposition are $1 \times 4 \times 1$ and $1 \times 1 \times 4$. The performance at the smaller exchanging data size is better than the larger by the speed from 1.3 to 1.6 times, and it is clear that the performances are largely affected by the size. So we can conclude that the slowness of access to the SMB is the bottleneck.

5 Conclusion

In this paper, we have established the λ computing environment by the AWG–STAR system. Adding that, we implemented an MPI library and evaluated the performance of distributed-parallel computation in the λ computing environment. As a result, we confirmed that the performance depends on the transmitting data size and the number of accesses to the shared memory, which is due to the slow accesses to the SMB. The new architecture of SMB is now investigated, by which we expect to attain higher throughput for distributed computation uti-

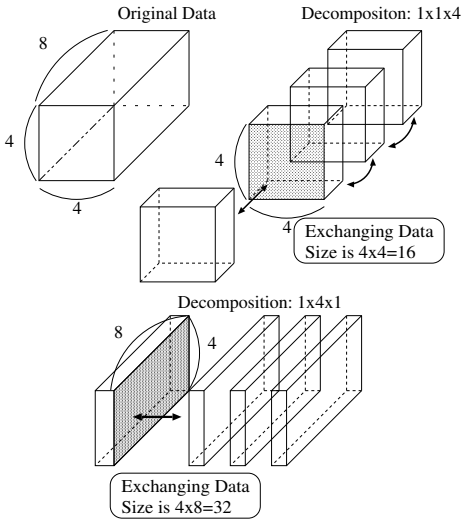

Figure 9: Domain decomposition and the size of the exchanging data.

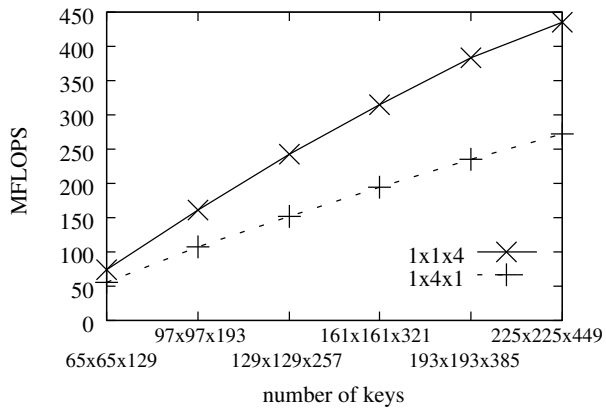

Figure 10: Comparison with the decomposition.

lizing wavelength paths of optical networks

Acknowledgment

We thank Dr. Akira Okada at NTT and Associate Professor Noriyuki Fujimoto at Osaka University for their helpful suggestions and supports.

References

[1] "Globus Toolkit," available at `http://www.globus.org/`.

[2] M. Murata and K. Kitayama, "A 1,000-channel WDM network can resolve network bottleneck," Proceedings of the 7th Asia-Pacific Conference on Communications (APCC 2001) (Tokyo), pp.113–116, Sep 2001.

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

[4] S. L. Danielsen, C. Joergesen, B. Mikkelsen, and K. E. Stubkjaer, "Analysis of a WDM packet switch with improved performance under bursty traffic conditions due to tuneable wavelength converters," IEEE Journal of Lightwave Technology, vol.16, pp.729–735, May 1998.

[5] D. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," IEEE Journal of Lightwave Technology, vol.16, pp.2081–2094, Dec 1998.

[6] K. L. Hall and K. A. Rauschenbach, "All-optical buffering of 40-gb/s data packets," IEEE Photonic Technology Letters, pp.442–444, 1998.

[7] 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, no.8, pp.2310–2318, Aug 2003.

[8] K. Baba, R. Takemori, M. Murata, and K. Kitayama, "A packet scheduling algorithm for the 2x2 photonic packet switch with FDL buffers," Proceedings of 28th Europian Conference on Optical Communication 2002 (ECOC2002), Sep 2002.

[9] H. Nakamoto, K. Baba, and M. Murata, "Proposal and Evaluation of Realization Approach for a Shared Memory System in λ Computing Environment," in Proceedings of the forth International Conference on Optical Internet (COIN2005), pp.90–95, May 2005.

[10] E. Taniguchi, K. Baba, and M. Murata, "Implementation and Evaluation of Shared Memory System for Establishing λ Computing Environment," in Proceedings of 10th OptoElectronics and Communications Conference (OECC2005), 5A2-3, pp.20–21, Jul 2005.

[11] Y. Sakai, K. Noguchi, R. Yoshimura, T. Sakamoto, A. Okada, and M. Matsuoka, "Management system for full-mesh WDM AWG–STAR network," 27th European Conference on Optical Communication, 2001, pp.264–265, Sep 2001.

[12] A. Okada, H. Tanobe, and M. Matsuoka, "Dynamically reconfigurable real-time information-sharing network system based on a cyclic-frequency AWG and tunable-wavelength lasers," in Proceedings of ECOC2003, Sep 2003.

[13] "MPICH-G2," available at `http://www3.niu.edu/mpi/`.

[14] "Himeno Benchmark," available at `http://accc.riken.jp/HPC/HimenoBMT/`.