Advanced Network Architecture research group

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

Mai IMOTO
Osaka University, Japan

Outline

- Research background
  - λ computing environment
- Objective of our research
- Implementation of MPI library utilizing shared memory in λ computing environment
  - Experimental system with AWG-STAR system
- Evaluation
- Conclusion and future work

Research Background

- Grid computing
  - Connect distributed computing nodes
  - Share resources and storages

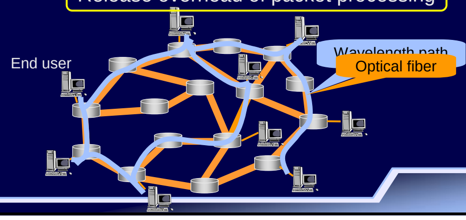
  - Large scale computing distributed in wide area
  - Transmitting huge data
- Overhead of transmission over TCP/IP
  - Overhead from packet processing
  - Re-transmission due to packet loss

New technology which provides highly reliable, high speed connections for end users is needed

λ Computing Environment (1/2)

- Connect computing nodes and optical routers by optical fibers
- Establish wavelength paths on the fibers
- Do not use TCT/IP
- Data is exchanged on wavelength paths which are treated as a granular units

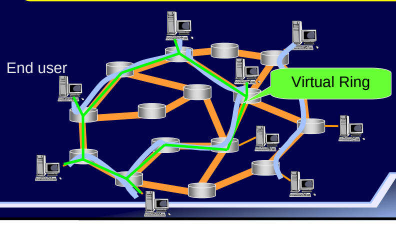
Release overhead of packet processing



λ Computing Environment (2/2)

- Establish a virtual ring by connecting wavelength paths
- By building a virtual ring, the optical ring is used as exclusive high speed transmission channel

Achieve highly reliable, high speed communication between end users



Establishing our Grid Environment with Globus Toolkit

- Globus Toolkit
  - Middleware of the grid environment for communication, authentication and job control
  - Provides users with interface which is independent of implementation
  - De-facto standard of grid middleware
- Adopt the Globus Toolkit as a upper layer of the λ computing environment

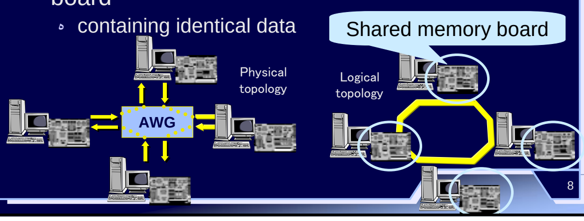
Users can perform high speed distributed computation without changing their original program

Objective of our Research

- Adopt the Globus Toolkit to the λ computing environment
- Implement and evaluate MPI library in the λ computing environment
  - Use the AWG-STAR system developed by NTT Photonics Laboratory

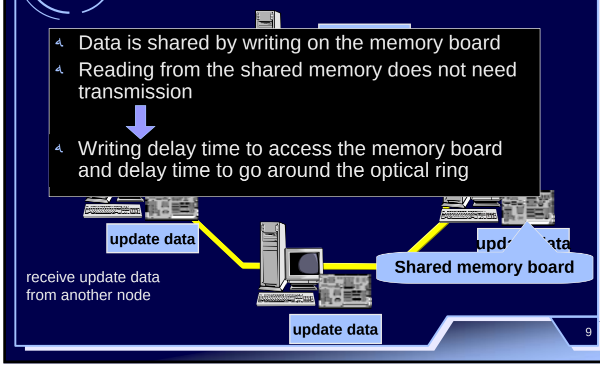
AWG-STAR System

- Information sharing network platform
- Computing nodes connected to the AWG router in ring topology
- Each node is equipped with a shared memory board
  - containing identical data



Data Sharing with the AWG-STAR System

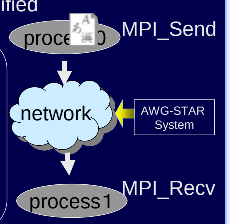
- Data is shared by writing on the memory board
- Reading from the shared memory does not need transmission
- Writing delay time to access the memory board and delay time to go around the optical ring



MPI (Message Passing Interface)

- Interface specification for data transmission in parallel computing
  - Underlying network is not specified

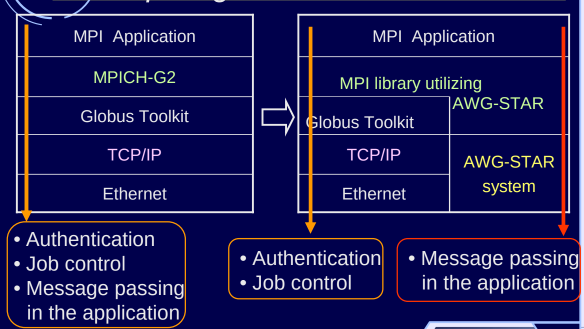
```
if(my_rank == 0){
  sprintf(message,"%s", Hello);
  MPI_Send ( Destination rank: 1 );
}
else if (my_rank == 1){
  MPI_Recv ( Source rank: 0 );
}
```



Implementation of MPI library

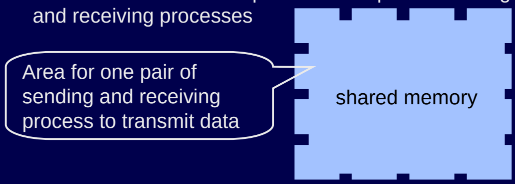
- Implementation strategy
  - Create MPI library from scratch
  - Create MPI library based on another MPI library
- Our MPI library bases on MPICH-G2 which works on the Globus Toolkit
  - Authentication and job control are same as MPICH-G2
  - Data exchange during the application is switched from TCP/IP to the AWG-STAR system

Parallel Computing in the λ Computing Environment



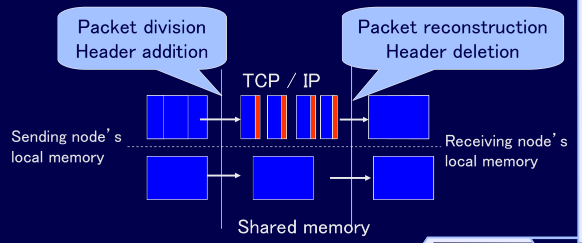
Implementation of MPI Library utilizing Shared Memory

- Dynamic memory allocation is not yet supported
  - Number of process: *n*
  - Shared memory is divided into *n* x *n* areas
  - Each area is used as a queue for one pair of sending and receiving processes



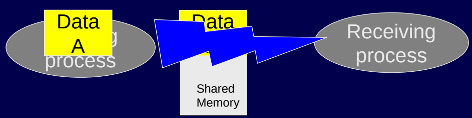
Message Passing

- Whole transmit data is written to the shared memory at a time



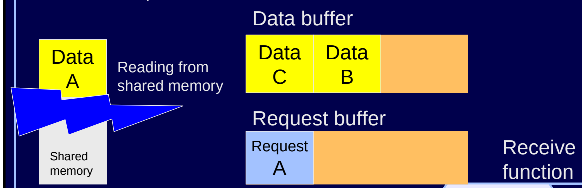
Sending Process

- When a sending function is called, process enqueues transmit data in the shared memory
- After writing, the process sends a signal to the receiving process
  - The signal is provided as a function of the AWG-STAR system



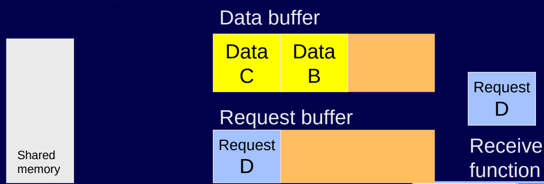
Receiving Process

- Timings when the process receives the data and when the receiving function is called are different
  - Implementing two buffers in the local memory
    - Data buffer
    - Request buffer



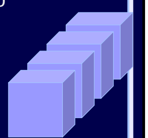
Receiving Process

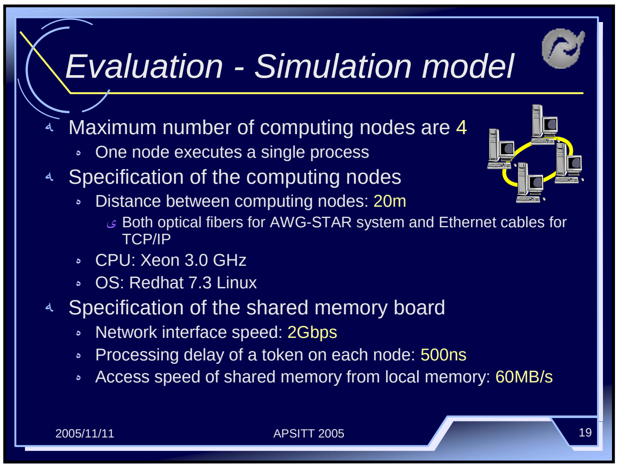
- Timings when the process receives the data and when the receiving function are different
  - Implementing two buffers on the local memory
    - Data buffer
    - Request buffer

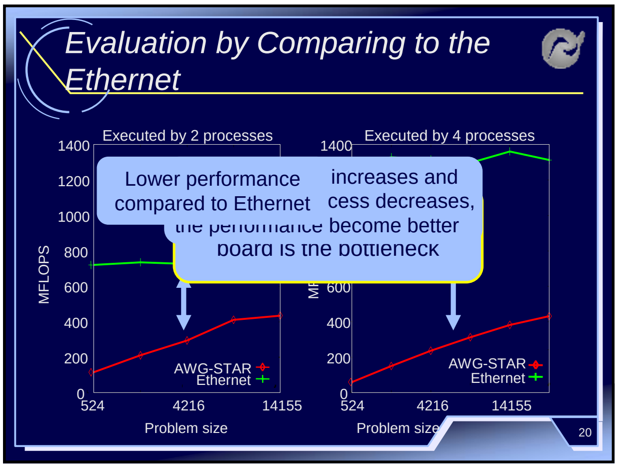


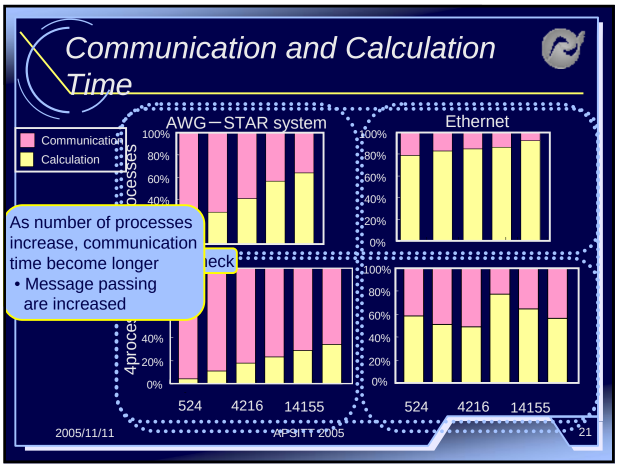
Evaluation - Application

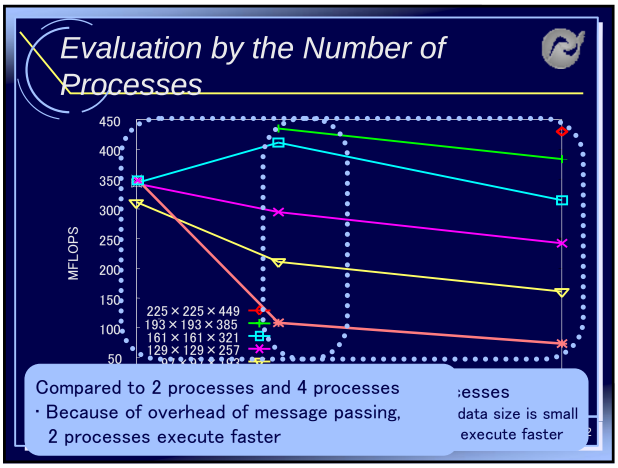
- Evaluation by Himeno Benchmark
  - It measures the computational time in MFLOPS by solving Jacobi method
  - Three dimensional array is partitioned into smaller arrays which are assigned to all computing nodes
  - Every process calculates their partitioned array and transmits the boundary region to the next process by utilizing MPI.
  - Problem size is variable
    - Transmit data size are proportion to problem size
    - Number of message-passing is inverse proportion to problem size

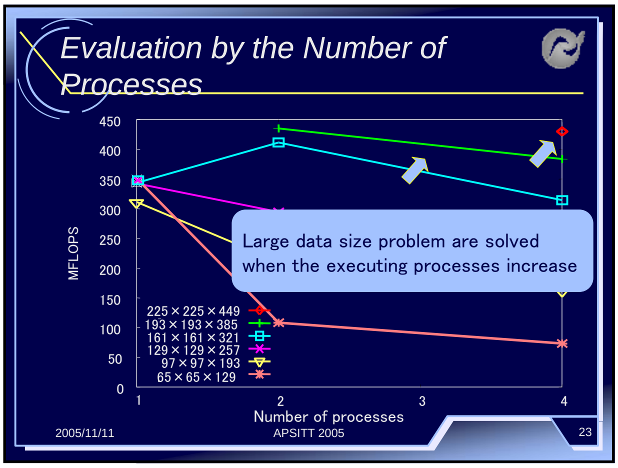


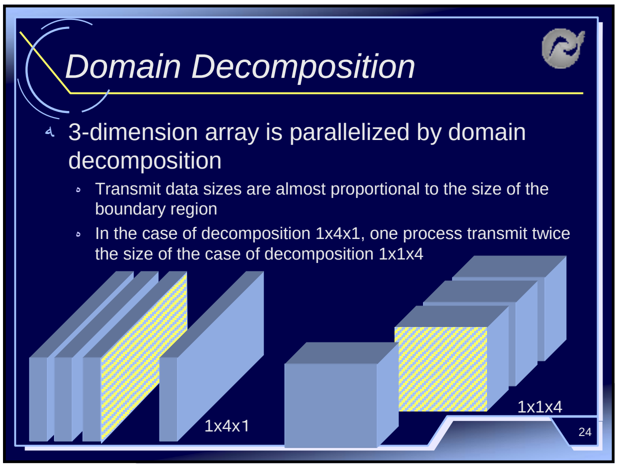

Evaluation - Simulation model
Maximum number of computing nodes are 4
One node executes a single process
Specification of the computing nodes
Distance between computing nodes: 20m
Both optical fibers for AWG-STAR system and Ethernet cables for TCP/IP
CPU: Xeon 3.0 GHz
OS: Redhat 7.3 Linux
Specification of the shared memory board
Network interface speed: 2Gbps
Processing delay of a token on each node: 500ns
Access speed of shared memory from local memory: 60MB/s
2005/11/11
APSITT 2005
19


Evaluation by Comparing to the Ethernet
Executed by 2 processes
Executed by 4 processes
Lower performance compared to Ethernet
increases and cess decreases, the performance become better board is the bottleneck
MFLOPS
AWG-STAR
Ethernet
524
4216
14155
Problem size
20


Communication and Calculation Time
Communication
Calculation
AWG－STAR system
Ethernet
As number of processes increase, communication time become longer
• Message passing are increased
524
4216
14155
2005/11/11
APSITT 2005
21

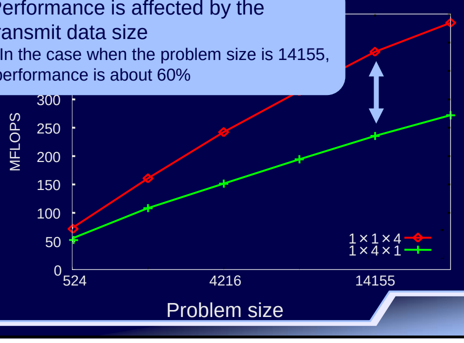

Evaluation by the Number of Processes
MFLOPS
225 × 225 × 449
193 × 193 × 385
161 × 161 × 321
129 × 129 × 257
Compared to 2 processes and 4 processes
· Because of overhead of message passing, 2 processes execute faster
data size is small execute faster


Evaluation by the Number of Processes
MFLOPS
Large data size problem are solved when the executing processes increase
225 × 225 × 449
193 × 193 × 385
161 × 161 × 321
129 × 129 × 257
97 × 97 × 193
65 × 65 × 129
Number of processes
2005/11/11
APSITT 2005
23


Domain Decomposition
3-dimension array is parallelized by domain decomposition
Transmit data sizes are almost proportional to the size of the boundary region
In the case of decomposition 1x4x1, one process transmit twice the size of the case of decomposition 1x1x4
1x4x1
1x1x4
24

Evaluation by the Decomposition

Performance is affected by the transmit data size
- In the case when the problem size is 14155, performance is about 60%



Conclusion and Future Work

- Conclusion
  - Adopted the Globus Toolkit into the λ computing environment
  - Implemented and evaluating MPI library in the λ computing environment
  - Delay of access to the shared memory board is the bottleneck
- Future work
  - New architecture of the shared memory board is now investigated