Implementation and Evaluation of MPI Library with Globus Toolkit for Establishing $\lambda$ Computing Environment

Mai IMOTO
Osaka University, Japan

Outline
- Research background
  - $\lambda$ computing environment
- Objective of our research
- Implementation of MPI library utilizing shared memory in $\lambda$ 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

$\lambda$ Computing Environment (1/2)
- Connect computing nodes and optical routers by optical fibers
- Establish wavelength paths on the fibers
- Data is exchanged on wavelength paths which are treated as a granular units
- Release overhead of packet processing
- Overhead of transmission over TCP/IP

$\lambda$ 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 $\lambda$ computing environment
- Users can perform high speed distributed computation without changing their original program
Objective of our Research

- Adopt the Globus Toolkit to the computing environment
- Implement and evaluate MPI library in the 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

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

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 Computing Environment

- Authentication
- Job control
- Message passing in the application

Parallel Computing in the Computing Environment

- Authentication
- Job control
- Message passing in the application
Implementation of MPI Library utilizing Shared Memory

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

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

Message Passing

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

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
Maximum number of computing nodes are 4
• One node executes a single process

Specification of the computing nodes
• Distance between computing nodes: 20m
• Four 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

Evaluation - Simulation model

Evaluation by Comparing to the Ethernet

Evaluation by the Number of Processes

Communication and Calculation Time

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
**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