Message Passing Interface (MPI)

Fitriyani
Program Studi Ilmu Komputasi
Fakultas Informatika, Universitas Telkom
fitriyani@telkomuniversity.ac.id

Persiapan personal

1. Menggunakan laptop dengan OS Ubuntu/sejenisnya atau laptop OS Windows plus SSH/ putty
2. Filezilla atau winscp untuk transfer file

Intro

- MPI adalah standar untuk menulis aplikasi paralel pada arsitektur *distributed memory*.
- MPI mendukung bahasa pemrograman C, C++, dan Fortran.
- Header: #include <mpi.h>

Compile dan Run OpenMPI pada server Ilmu Komputasi

- Headnode: 10.30.40.113
- User dan pass: minta ke dosen pengajar
- Cara menjalankan program MPI:
  - $ module add openmpi-x86_64
  - Cara compile: $ mpicc fileparallel.c
  - Cara eksekusi: $ mpirun -np 8 ./a.out

Outlines

- MPI Basics
- MPI point to poitn communication
- MPI collective communication

Basic MPI Calls

- MPI_INIT → Initialize MPI
- MPI_COMM_RANK →Get the processor rank
- MPI_COMM_SIZE →Get the number of processors
- MPI_Send → Send data to another processor
- MPI_Recv → Get data from another processor
- MPI_FINALIZE → Finish MPI

Inisiasi program MPI

- Initialize the MPI environment

  MPI_Init(NULL, NULL);
- Get the number of processes

  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
- Get the rank of the process

  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
- Get the name of the processor

  MPI_Get_processor_name(processor_name, &name_len);
- Finalize the MPI environment

  MPI_Finalize();

Blocking Point to Point Communication

- Sending and receiving with MPI_Send and MPI_Recv
- Dynamic receiving with MPI_Probe and MPI_Status

Sending and receiving with MPI_Send and MPI_Recv

- MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest,int tag, MPI_Comm comm)
- MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)
- Parameter:
  - Buf → Initial address of send buffer (choice)
  - Count → Number of elements send (nonnegative integer)
  - Datatype →Datatype of each send buffer element (handle)
  - Dest → Rank of destination (integer)
  - Tag → Message tag (integer)
  - Comm → Communicator (handle)

MPI C Standard Data Types

• MPI_CHAR	signed char
• MPI_SHORT	signed short int
• MPI_INT	signed int
• MPI_LONG	signed long int
• MPI_UNSIGNED_CHAR	unsigned char
• MPI_UNSIGNED_SHORT	unsigned short int
• MPI_UNSIGNED	unsigned int
• MPI_UNSIGNED_LONG	unsigned long int
• MPI_FLOAT	float
• MPI_DOUBLE	double
• MPI_LONG_DOUBLE	long double
• MPI_BYTE	8 binary digits
• MPI_PACKED	data packed or unpacked with MPI_Pack()/ MPI_Unpack

Contoh

```
if(rank==0) {
int a = 1; float b = 0.5;
printf("saya master node  \n");

MPI_Send(&a,1,MPI_INT,1,0,MPI_COMM_WORLD);
MPI_Send(&b,1,MPI_FLOAT,2,0,MPI_COMM_WORLD);
}
else if(rank==1){
MPI_Recv(&a,1,MPI_INT,0,0,MPI_COMM_WORLD,&status);
printf("saya computenode 1  menerima integer %i\n",a);
}
else if(rank==2){
MPI_Recv(&b,1,MPI_FLOAT,0,0,MPI_COMM_WORLD,&status);
printf("saya computenode2  menerima float %f\n",b);
}
```

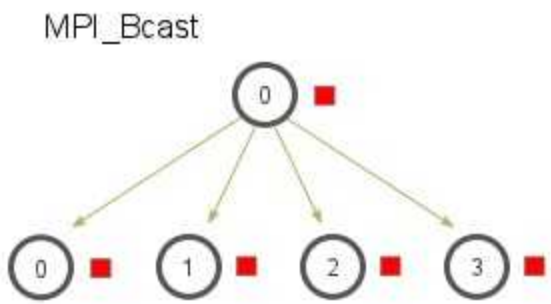
```
int a = 1; float b = 0.5;

if(rank==0) {
printf("saya master node  \n");

MPI_Send(&a,1,MPI_INT,1,0,MPI_COMM_WORLD);
MPI_Send(&b,1,MPI_FLOAT,2,0,MPI_COMM_WORLD);
}
else if(rank==1){
int a;
MPI_Recv(&a,1,MPI_INT,0,0,MPI_COMM_WORLD,&status);
printf("saya computenode 1  menerima integer %i\n",a);
}
else if(rank==2){
float  y;
MPI_Recv(&y,1,MPI_FLOAT,0,0,MPI_COMM_WORLD,&status);
printf("saya computenode2  menerima float %f\n",y);
}
```

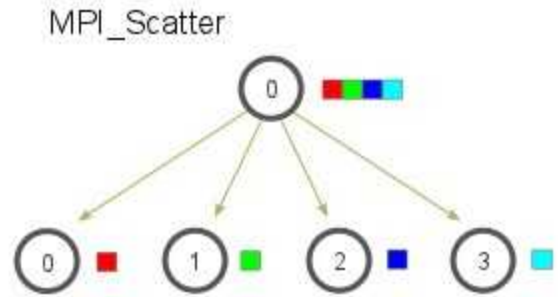
MPI Collective Communication

- MPI_Barrier
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce

MPI Collective Communication

- MPI_Barrier (MPI_Comm communicator)
- MPI_Bcast (void* buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)
- MPI_Scatter(void* send_data,int send_count,MPI_Datatype send_datatype,void* recv_data,int recv_count,MPI_Datatype recv_datatype, int root,MPI_Comm communicator)

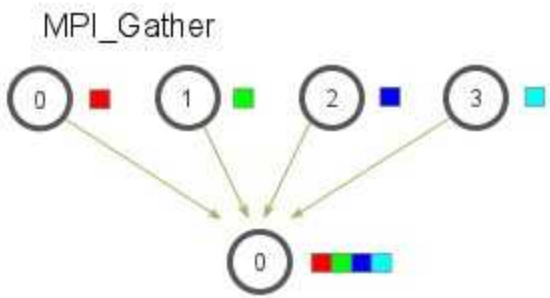


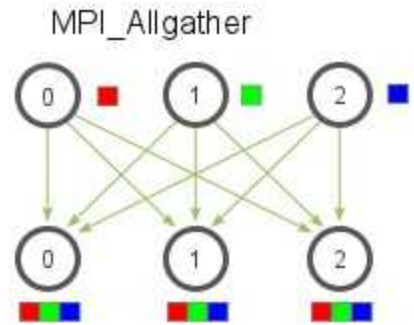


Gambar diambil dari mpitutorial.com

MPI Collective Communication

- MPI_Gather(void* send_data,int send_count,MPI_Datatype send_datatype,void* recv_data,int recv_count, MPI_Datatype recv_datatype, int root,MPI_Comm communicator)
- MPI_Allgather(void* send_data,int send_count,MPI_Datatype send_datatype,void* recv_data,int recv_count,MPI_Datatype recv_datatype,MPI_Comm communicator)

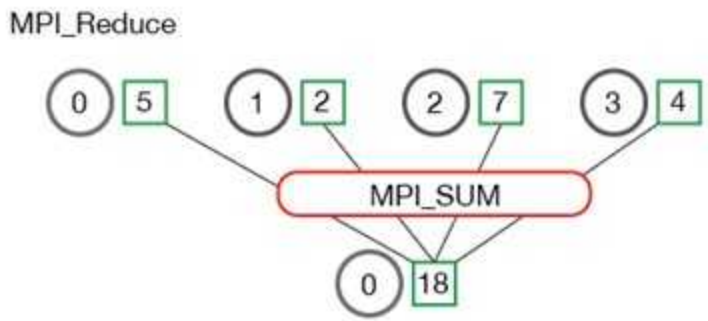


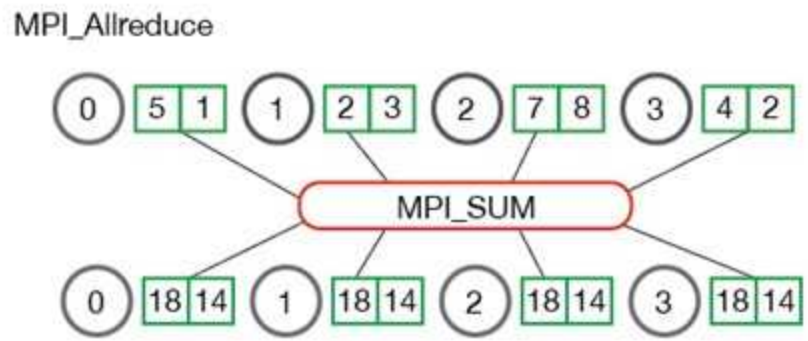


Gambar diambil dari mpitutorial.com

MPI Collective Communication

- MPI_Reduce(void* send_data,void* recv_data,int count, MPI_Datatype datatype,MPI_Op op,int root,MPI_Comm communicator)
- MPI_Allreduce(void* send_data,void* recv_data,int count, MPI_Datatype datatype,MPI_Op op,MPI_Comm communicator)





Gambar diambil dari mpitutorial.com

MPI Reduction Operation

- MPI_MAX maximum
- MPI_MIN minimum
- MPI_SUM sum
- MPI_PROD product
- MPI_LAND logical AND
- MPI_BAND bit-wise AND
- MPI_LOR logical OR
- MPI_BOR bit-wise OR
- MPI_LXOR logical XOR
- MPI_BXOR bit-wise XOR
- MPI_MAXLOC max value and location
- MPI_MINLOC min value and location

Referensi

- http://mpitutorial.com/
- https://www.open-mpi.org/
- https://cvw.cac.cornell.edu/MPIcc/broadcast
- https://computing.llnl.gov/tutorials/mpi/