Introduction to

MPI and OpenMP

myson @ postech.ac.kr

CSE700-PL @ POSTECH

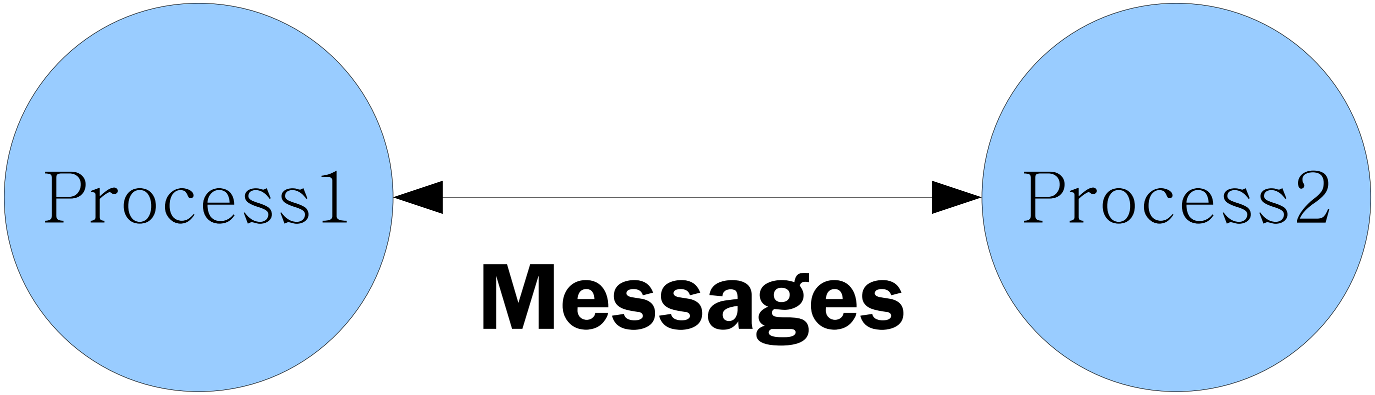
Outline

- MPI and OpenMP
  - Definition
  - Characteristics
  - Flow models
- Examples
- Compiling and Execution
- Resources

What are MPI and OpenMP?

- Message Passing Interface (MPI)

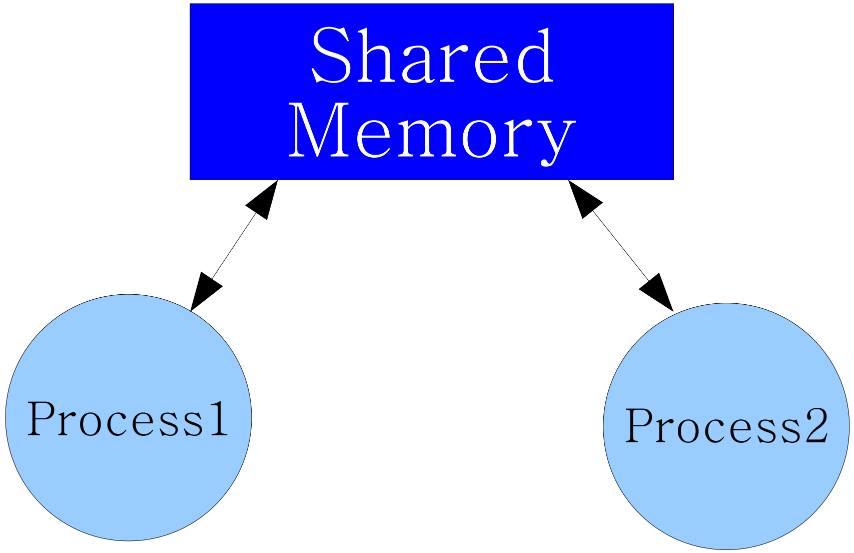
  - MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users.



What are MPI and OpenMP?

- Open Multi Processing (OpenMP)

  - OpenMP is a specification for a set of compiler directives, library routines, and environment variables that can be used to specify shared memory parallelism in Fortran and C/C++ programs.

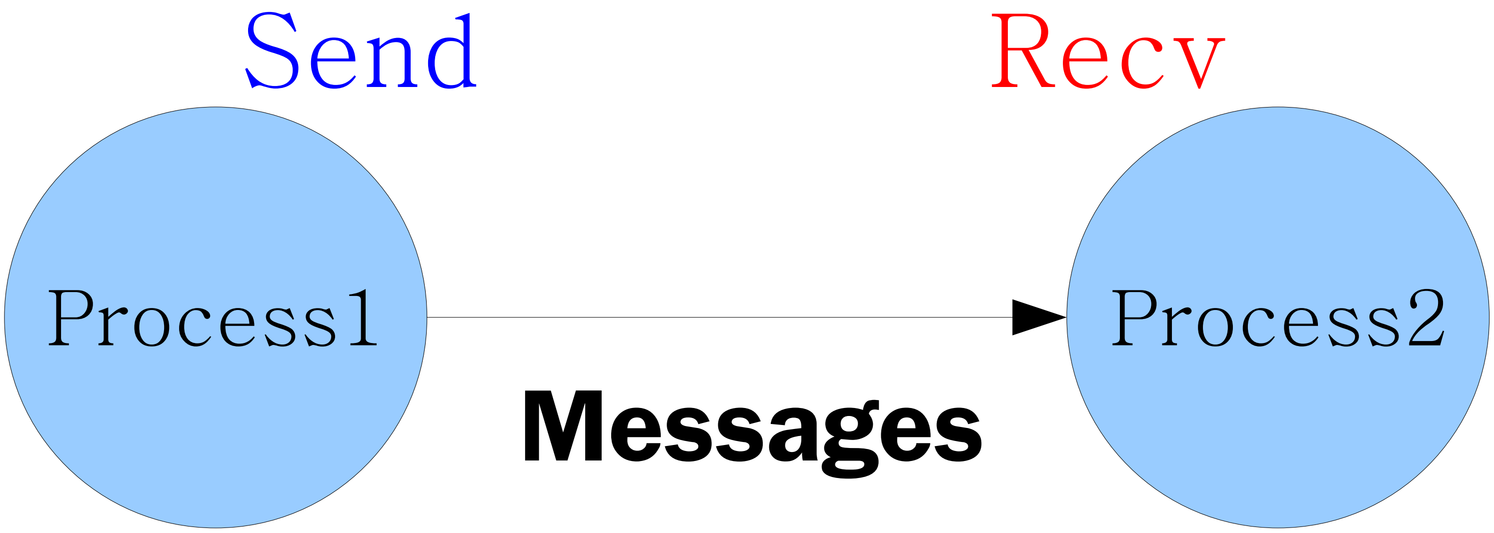


MPI vs. OpenMP

MPI	OpenMP
Distributed memory model	Shared memory model
on Distributed network	on Multi-core processors
Message based	Directive based
Flexible and expressive	Easier to program and debug

MPI Flow Model

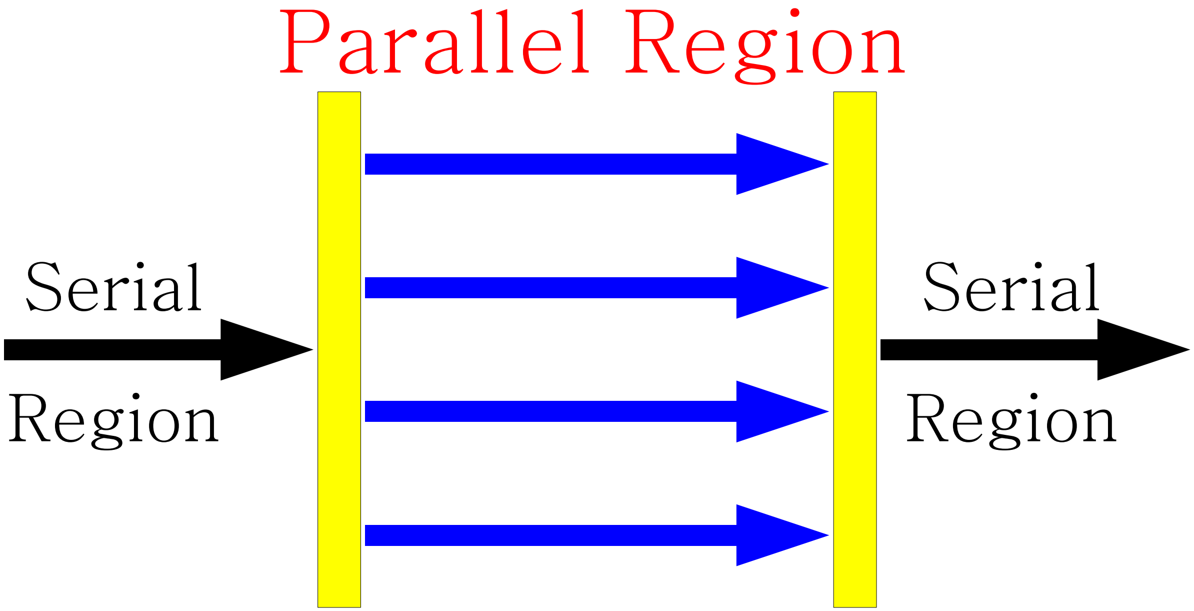
- Message Passing - Send and Receive



a message, size, type, source, dest,
tag, communicator, status

OpenMP Flow Model

- Directives (C/C++) - #pragma omp *directives* [clauses]



directives - parallel, for, single, etc.

A Simple Example

- A serial program

```
#include<stdio.h>
#define PID 0

main(){
  int i;
  printf("Greetings from process %d!/n", PID);
}
```

```
Greetings from process 0
```

A Simple Example(cont.)

- A parallel program using MPI (cont.)

```
#include<mpi.h>
main(int argc, char** argv){
  ⋮
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
  MPI_Comm_size(MPI_COMM_WORLD, &p);

         Parallel Region

  MPI_Finalize();
}
```

A Simple Example(cont.)

- A parallel program using MPI

```
if ( my_rank != 0){
  sprintf(message,
         "Greetings from process %d!", my_rank);
  dest = 0;
  MPI_Send(message, strlen(message)+1, MPI_CHAR,
           dest, tag, MPI_COMM_WORLD);
} else{  /* my_rank = 0 */
  for (source = 1; source < p; source++){
    MPI_Recv(message, 100, MPI_CHAR, source, tag,
             MPI_COMM_WORLD, &status);
    printf("%s/n", message);
  }
}
```

A Simple Example(cont.)

- A parallel program using MPI (cont.)

```
Greetings from process 1
Greetings from process 2
Greetings from process 3
```

A Simple Example(cont.)

- A parallel program using OpenMP

```
#include<stdio.h>
#include<omp.h>
main(){
  int id;
#pragma omp parallel
  {
    id = omp_get_thread_num();
    printf("Greetings from process %d!/n", id);
  }
}
```

A Simple Example(cont.)

- A parallel program using OpenMP (cont.)

```
Greetings from process 1
Greetings from process 0
Greetings from process 2
Greetings from process 3
```

Which is better?

Compiling

- GCC and MPICH2 for MPI
- GCC-4.2 with library libgomp for OpenMP

- MPI
  - mpicc -o example.out example.c
- OpenMP
  - gcc-4.2 -o example.out example.c -fopenmp

Execution

- ∼/.mpd.conf for MPI execution
  - vi(or emacs) ∼/.mpd.conf
    secretword=<your secretword>
  - chmod 600 ∼/.mpd.conf
- MPI (using multi-core processors)
  - mpdboot
  - mpiexec -n #processes ./example.out
  - mpdallexit
- OpenMP
  - ./example.out

Resources

- Machine (Plquad: plquad.postech.ac.kr)
  - Intel Core 2 Quad Q6600 (quad-core)
  - 1G DDR RAM
  - If you want to use it, email the instructors.

- Materials - resource tab on the course web-page
  - MPI & OpenMP install guides
  - MPI & OpenMP tutorials
  - ⋮

End

Any Questions. . . ?