MPI를 이용한 분산 메모리 프로그래밍

메시지 패싱 프로그램에서 코어-메모리에서 실행되는 프로그램을 보통 process 라고 하고, 두 개의 프로세스는 send 함수라고 하는 프로세스와 receive 함수라고 하는 프로세스를 사용하여 통신한다. 또한, 두 개 이상의 프로세스를 포함하는 “글로벌” 통신, 함수들을 collective 통신이라고 한다. MPI 함수들에 대하여 배우는 과정에서 분산 메모리 시스템의 데이터 파티셔닝과 I/O 와 같은 메시지 패싱 프로그램 이슈를 포함한 기본적인 이슈에 대해서 배울 것이다. 병렬 프로그램 성능 관련 이슈에 대해서도 알아볼 것이다.

Hello, world!

```c
#include <stdio.h>
#include <string.h>
#include <mpi.h>

const int MAX_STRING = 100;

int main(void) {
  char  greeting[MAX_STRING];
  int   comm_sz;    /* the number of processes */
  int   my_rank;    /* process' ranking number */

  MPI_Init(NULL, NULL);
  MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

  if (my_rank != 0) {
    sprintf(greeting, "Greetings from process %d of %d!", my_rank, comm_sz);
    MPI_Send(greeting, strlen(greeting) + 1, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
  } else {
    printf("Greetings from process %d of %d!\n", my_rank, comm_sz);
    for (int q = 1; q < comm_sz; q++) {
      MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      printf("%s\n", greeting);
    }
  }

  MPI_Finalize();
  return 0;
}
```

각 프로세스가 간단하게 메시지를 출력하는 대신 하나의 프로세스가 출력을 맡고 다른 프로세스는 메시지를 전송하도록 설계했다. 병렬 프로그램에서 음수가 아닌 정수형 rank 를 사용하여 프로세스를 식별하는 것이 일반적이다. p 개의 프로세스들이 있다면 그 프로세스들은 0, 1, 2, … , p-1 의 등급을 갖고 있다. 병렬로 “hello, world”를 출력하도록 만들기 위해서 프로세스 0이 메시지를 출력하도록 하고 다른 프로세스는 메시지를 전송하도록 설계했다.

Build and Run

컴파일 명령어: mpicc (wrapper script for c compiler) ($ mpicc -g -Wall -o mpi_hello mpi_hello.c)

실행 명령어: mpiexec ($ mpiexec -n ./mpi_hello)