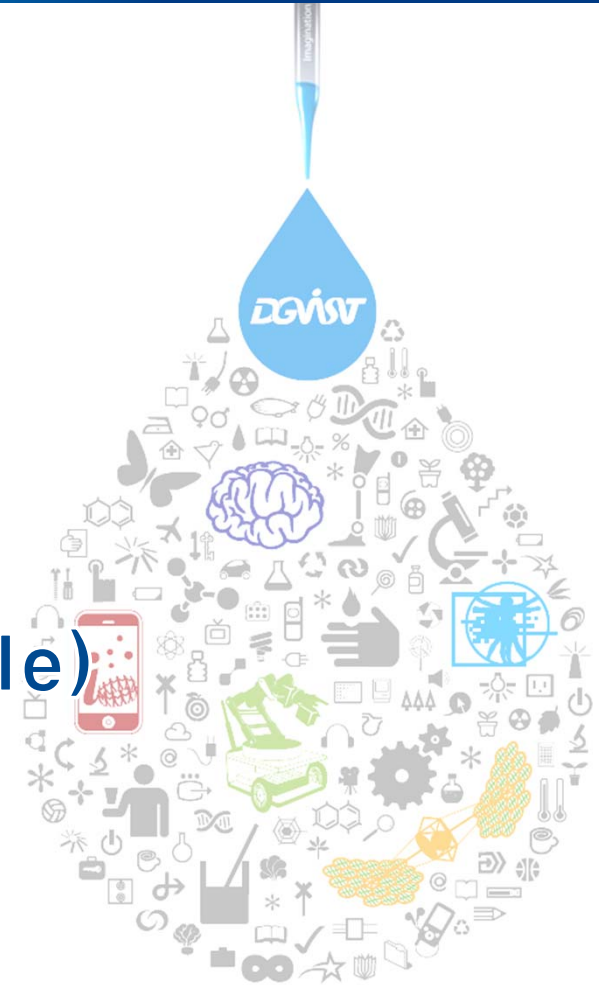


IC621: Distributed and Parallel  
Computing

**Lab 02 : MPI**

**(Hello World & Trapezoidal Rule)**

Min-Soo Kim



# Hello world example in MPI

- Implement, compile, and execute the “Hello world” example in Lecture 03

```

1 #include <stdio.h>
2 #include <string.h> /* For strlen */
3 #include <mpi.h> /* For MPI functions, etc */
4
5 const int MAX_STRING = 100;
6
7 int main(void) {
8     char    greeting[MAX_STRING];
9     int     comm_sz; /* Number of processes */
10    int     my_rank; /* My process rank */
11
12    MPI_Init(NULL, NULL);
13    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
14    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
15
16    if (my_rank != 0) {
17        sprintf(greeting, "Greetings from process %d of %d!",
18                my_rank, comm_sz);
19        MPI_Send(greeting, strlen(greeting)+1, MPI_CHAR, 0, 0,
20                MPI_COMM_WORLD);
21    } else {
22        printf("Greetings from process %d of %d!\n", my_rank,
23                comm_sz);
24        for (int q = 1; q < comm_sz; q++) {
25            MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q,
26                    0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
27            printf("%s\n", greeting);
28        }
29
30        MPI_Finalize();
31        return 0;
32    } /* main */

```

# Trapezoidal rule example in MPI

- Implement, compile, and execute the “Trapezoidal rule” example in Lecture 03
- You should implement three functions
  - `main()`
  - `Trap()`
  - `Get_input()`
- You can use the following setting
  - $f() : y = x^2$
  - $a : 1$
  - $b : 4$
  - $n : 1024$

```

1  int main(void) {
2      int my_rank, comm_sz, n = 1024, local_n;
3      double a = 0.0, b = 3.0, h, local_a, local_b;
4      double local_int, total_int;
5      int source;
6
7      MPI_Init(NULL, NULL);
8      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
9      MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
10
11     h = (b-a)/n;          /* h is the same for all processes */
12     local_n = n/comm_sz; /* So is the number of trapezoids */
13
14     local_a = a + my_rank*local_n*h;
15     local_b = local_a + local_n*h;
16     local_int = Trap(local_a, local_b, local_n, h);
17
18     if (my_rank != 0) {
19         MPI_Send(&local_int, 1, MPI_DOUBLE, 0, 0,
20                MPI_COMM_WORLD);
21     } else {
22         total_int = local_int;
23         for (source = 1; source < comm_sz; source++) {
24             MPI_Recv(&local_int, 1, MPI_DOUBLE, source, 0,
25                    MPI_COMM_WORLD, MPI_STATUS_IGNORE);
26             total_int += local_int;
27         }
28     }
29
30     if (my_rank == 0) {
31         printf("With n = %d trapezoids, our estimate\n", n);
32         printf("of the integral from %f to %f = %.15e\n",
33                a, b, total_int);
34     }
35     MPI_Finalize();
36     return 0;
37 } /* main */

```

```

1 double Trap(
2     double left_endpt /* in */,
3     double right_endpt /* in */,
4     int trap_count /* in */,
5     double base_len /* in */) {
6     double estimate, x;
7     int i;
8
9     estimate = (f(left_endpt) + f(right_endpt))/2.0;
10    for (i = 1; i <= trap_count-1; i++) {
11        x = left_endpt + i*base_len;
12        estimate += f(x);
13    }
14    estimate = estimate*base_len;
15
16    return estimate;
17 } /* Trap */

```

```

1 void Get_input(
2     int      my_rank    /* in */,
3     int      comm_sz   /* in */,
4     double*  a_p        /* out */,
5     double*  b_p        /* out */,
6     int*     n_p        /* out */) {
7     int dest;
8
9     if (my_rank == 0) {
10        printf("Enter a, b, and n\n");
11        scanf("%lf %lf %d", a_p, b_p, n_p);
12        for (dest = 1; dest < comm_sz; dest++) {
13            MPI_Send(a_p, 1, MPI_DOUBLE, dest, 0, MPI_COMM_WORLD);
14            MPI_Send(b_p, 1, MPI_DOUBLE, dest, 0, MPI_COMM_WORLD);
15            MPI_Send(n_p, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
16        }
17    } else { /* my_rank != 0 */
18        MPI_Recv(a_p, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
19                MPI_STATUS_IGNORE);
20        MPI_Recv(b_p, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
21                MPI_STATUS_IGNORE);
22        MPI_Recv(n_p, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
23                MPI_STATUS_IGNORE);
24    }
25 } /* Get_input */

```