DHRUV GARG 16 BCE 1190

PDC LAB 5 - COMMUNICATION

Compare the time taken for using MPI_Bcast and MPI_Send and Receive to perform the same operation

Code

```c
#include<stdio.h>
#include<stdlib.h>
#include<mpi.h>
#include<assert.h>
void my_bcast(void* data, int count, MPI_Datatype datatype, int root, MPI_Comm communicator) {
  int world_rank;
  MPI_Comm_rank(communicator, &world_rank);
  int world_size;
  MPI_Comm_size(communicator, &world_size);
 if (world_rank == root) {
    int i;
    for (i = 0; i < world_size; i++) {
      if (i != world_rank) {
        MPI_Send(data, count, datatype, i, 0, communicator);
      }
    }
  } else {
    MPI_Recv(data, count, datatype, root, 0, communicator, MPI_STATUS_IGNORE);
  }
}
int main(int argc, char** argv) {
  int num = atoi(argv[1]);
  int num_trials = atoi(argv[2]);
  MPI_Init(NULL, NULL);
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  double elapsed_time = 0.0;
  double total_mpi_bcast_time = 0.0;
  int i;
  int* data = (int*)malloc(sizeof(int) * num);
  assert(data != NULL);
```

```c
  for (i = 0; i < num_trials; i++) {
    MPI_Barrier(MPI_COMM_WORLD);
    elapsed_time -= MPI_Wtime();
    my_bcast(data, num, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Barrier(MPI_COMM_WORLD);
    elapsed_time += MPI_Wtime();
    MPI_Barrier(MPI_COMM_WORLD);
    total_mpi_bcast_time -= MPI_Wtime();
    MPI_Bcast(data, num, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Barrier(MPI_COMM_WORLD);
    total_mpi_bcast_time += MPI_Wtime();
  }
  if (world_rank == 0) {
    printf("Avg my_bcast time = %lf\n", elapsed_time / num_trials);
    printf("Avg MPI_Bcast time = %lf\n", total_mpi_bcast_time / num_trials);
  }
  free(data);
  MPI_Finalize();
}
```

Output

```
Data size = 400, Trials = 10
Average myBcast time = 0.002347
Average MPI_Bcast time = 0.0000301
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab 5/Problem 3$
```

Find the average of the random numbers generated by master and scattered to all slave nodes

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<mpi.h>
#include<assert.h>
float *create_rand_nums(int num){
 float *rand_nums = (float *)malloc(sizeof(float) * num);
 assert(rand_nums != NULL);
 int i;
 for (i = 0; i < num; i++){
 rand_nums[i] = (rand() / (float)RAND_MAX);
 }
  return rand_nums;
 }

 float avg(float *array, int num){
 float sum = 0.f;
 int i;
 for (i = 0; i < num; i++){
  sum += array[i];
 }
 return sum / num;
 }

 int main(int argc, char** argv){
 int num_per_proc = atoi(argv[1]);
 srand(time(NULL));
 MPI_Init(NULL, NULL);
 int world_rank;
 MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
 int world_size;
 MPI_Comm_size(MPI_COMM_WORLD, &world_size);
 float *rand_nums = NULL;
 if (world_rank == 0) {
  rand_nums = create_rand_nums(num_per_proc * world_size);
 }
```

```
38  float *sub_rand_nums = (float *)malloc(sizeof(float) * num_per_proc);
39  assert(sub_rand_nums != NULL);
40  MPI_Scatter(rand_nums, num_per_proc, MPI_FLOAT, sub_rand_nums, num_per_proc, MPI_FLOAT, 0, MPI_COMM_WORLD);
41  float sub_avg = avg(sub_rand_nums, num_per_proc);
42  float *sub_avgs = NULL;
43  if (world_rank == 0) {
44  sub_avgs = (float *)malloc(sizeof(float) * world_size);
45  assert(sub_avgs != NULL);
46  }
47
48  MPI_Gather(&sub_avg, 1, MPI_FLOAT, sub_avgs, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
49  if (world_rank == 0) {
50   float avg = avg(sub_avgs, world_size);
51   printf("Avg of all elements is %f\n", avg);
52   float original_data_avg = avg(rand_nums, num_per_proc * world_size);
53   printf("Computed across original data is %f\n", original_data_avg);
54   }
55   if (world_rank == 0) {
56   free(rand_nums);
57   free(sub_avgs);
58   }
59
60   free(sub_rand_nums);
61   MPI_Barrier(MPI_COMM_WORLD);
62   MPI_Finalize();
63
```

Output

```
Average of all the elements: 48.49247
Average computed using ORIGINAL data: 48.49247
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab 5/Problem 3$
```