CSE 4001

PARALLEL AND DISTRIBUTED COMPUTING

Lab FAT Exam

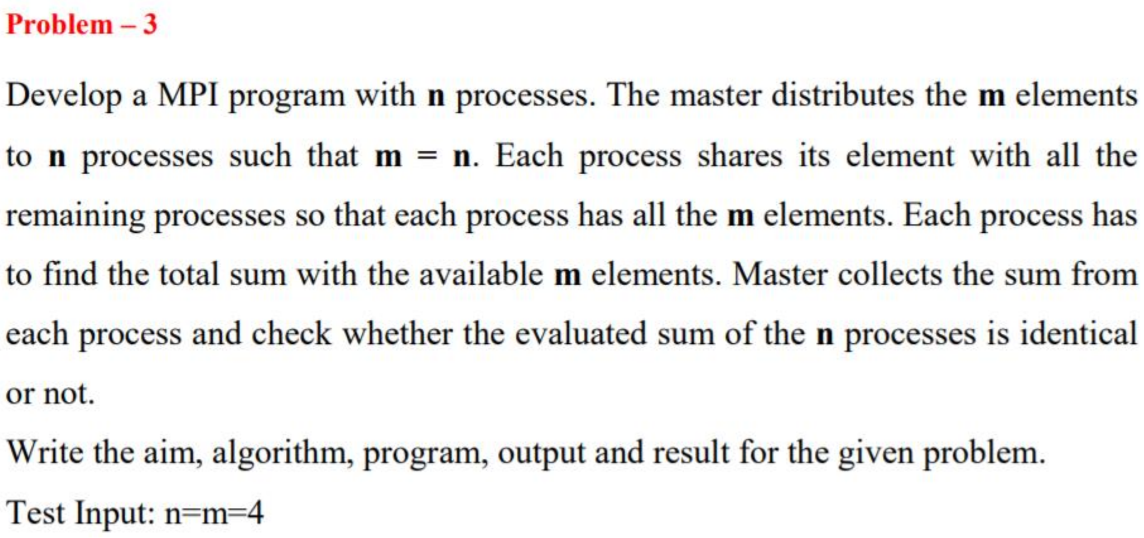
L27+L28 | PLBG04
Dr. Narayanan Prasanth

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI
19BCE2105

19BCE2105	SHARADINDU ADHIKARI	Problem 3



Problem – 3

Develop a MPI program with **n** processes. The master distributes the **m** elements to **n** processes such that **m = n**. Each process shares its element with all the remaining processes so that each process has all the **m** elements. Each process has to find the total sum with the available **m** elements. Master collects the sum from each process and check whether the evaluated sum of the **n** processes is identical or not.

Write the aim, algorithm, program, output and result for the given problem.

Test Input: n=m=4

Solution:

Aim: To calculate the sum of **m** elements using **n** processes distributed by master (and verified by the master), so that **m = n**.

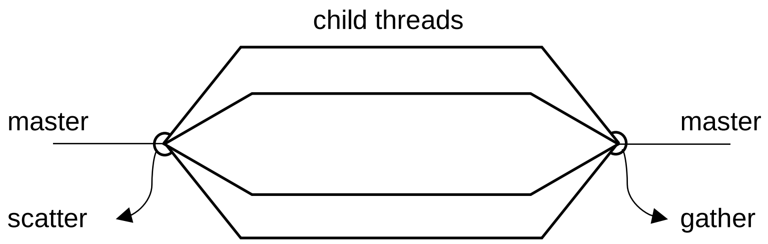
Algorithm: Algo I followed to implement the given program:

Step 1: First, I generated a random array of the required number of elements.

Step 2: Thereafter, I generated a function to calculate the sum of all the elements that are passed through it.

Step 3: From the master thread, I then divided the task. That is, scatter it to the number of processes required, which in here, is 4.

Step 4: From each of the processes, I then have taken the output, and gathered them back to the master thread.

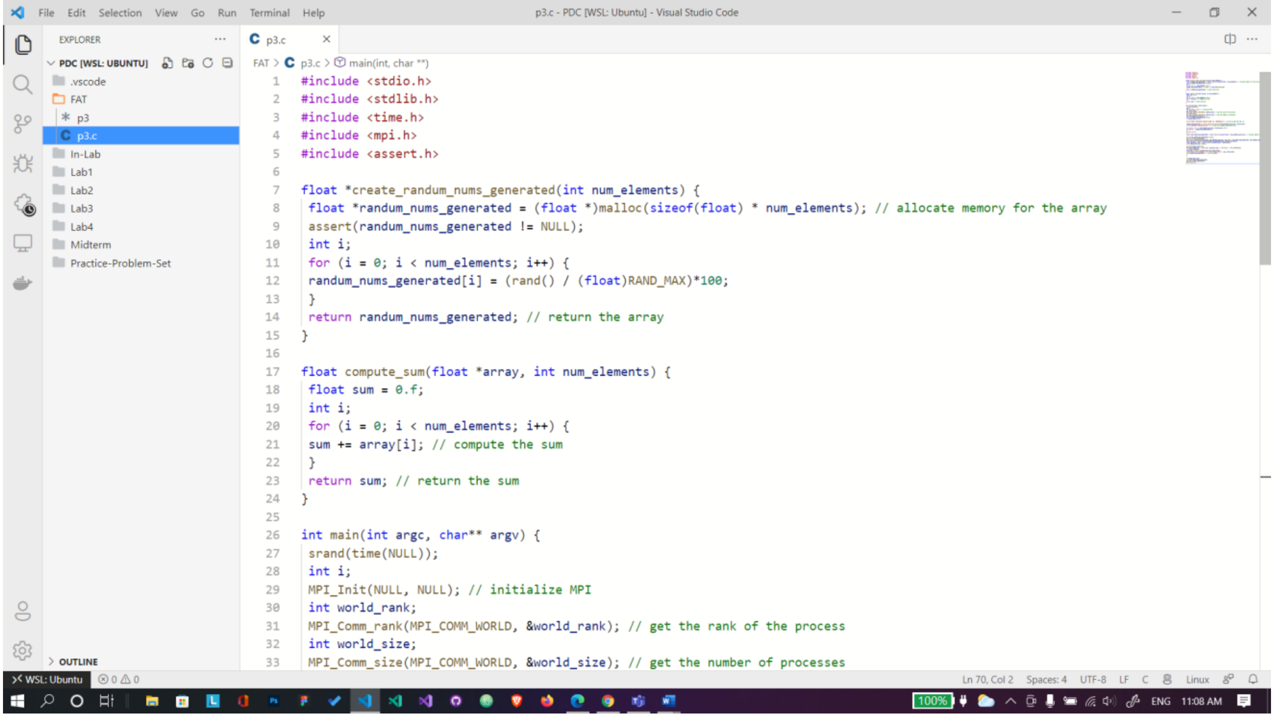


Step 5: Then I've verified the sum from each process.

Step 6: Followed by freeing the variables, finalising the barrier, and finalising the Message Passing Interface.

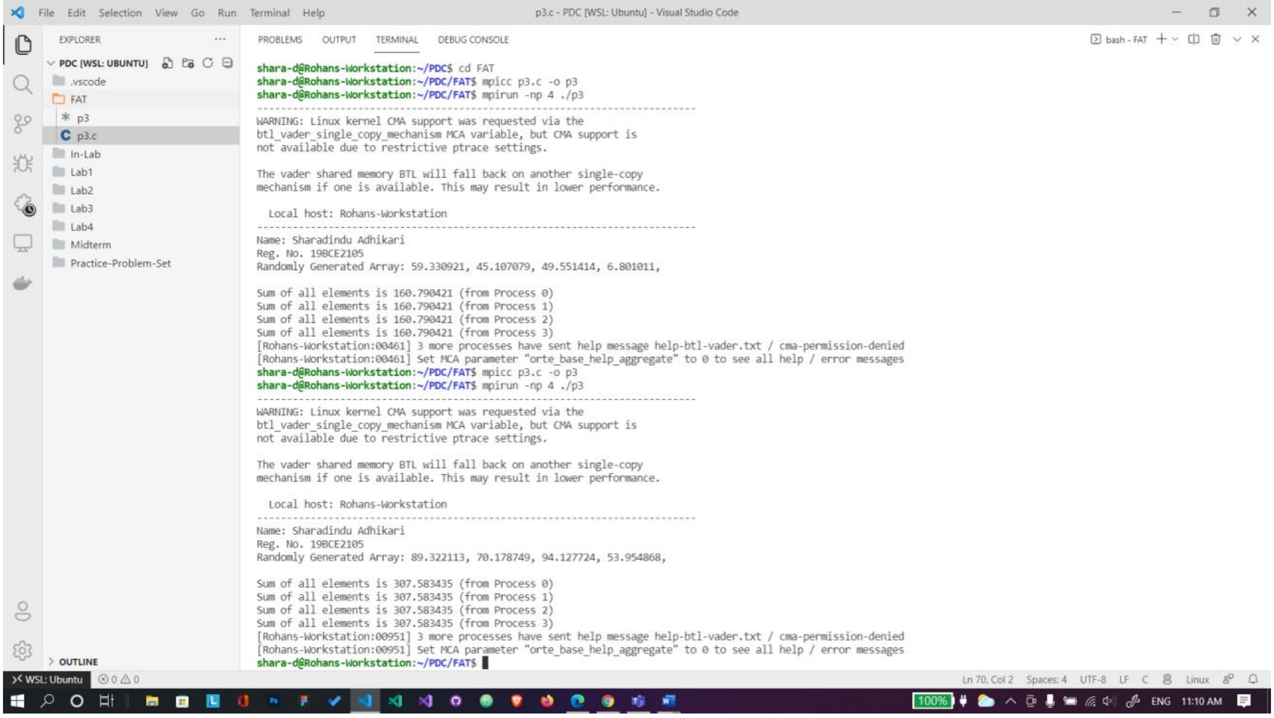
Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>
#include <assert.h>

float *create_randum_nums_generated(int num_elements) {
 float *randum_nums_generated = (float *)malloc(sizeof(float) *
num_elements); // allocate memory for the array
 assert(randum_nums_generated != NULL);
 int i;
 for (i = 0; i < num_elements; i++) {
 randum_nums_generated[i] = (rand() / (float)RAND_MAX)*100;
 }
 return randum_nums_generated; // return the array
}

float compute_sum(float *array, int num_elements) {
 float sum = 0.f;
 int i;
 for (i = 0; i < num_elements; i++) {
 sum += array[i]; // compute the sum
 }
 return sum; // return the sum
}

int main(int argc, char** argv) {
 srand(time(NULL));
 int i;
 MPI_Init(NULL, NULL); // initialize MPI
 int world_rank;
 MPI_Comm_rank(MPI_COMM_WORLD, & world_rank); // get the rank of the
process
 int world_size;
 MPI_Comm_size(MPI_COMM_WORLD, &world_size); // get the number of processes
 int num_elements_per_proc = 1;
 float *randum_nums_generated = NULL;
 if (world_rank == 0) {

 printf("Name: Sharadindu Adhikari\nReg. No. 19BCE2105\n"); // print my
name and reg. no.

 randum_nums_generated = create_randum_nums_generated(num_elements_per_proc
* world_size);
 printf("Randomly Generated Array: "); // print the randomly generated
array

 for (int i = 0; i < num_elements_per_proc * world_size; i++) {
 printf("%f, ",randum_nums_generated[i]);
 }
 printf("\n\n");
 }
 float *sub_randum_nums_generated = (float *)malloc(sizeof(float) *
num_elements_per_proc); // allocate memory for the array

 assert(sub_randum_nums_generated != NULL);
 MPI_Barrier(MPI_COMM_WORLD);
 MPI_Scatter(randum_nums_generated, num_elements_per_proc, MPI_FLOAT,
sub_randum_nums_generated, num_elements_per_proc, MPI_FLOAT, 0,
MPI_COMM_WORLD);
 float sub_sum =
compute_sum(sub_randum_nums_generated,num_elements_per_proc);
 float *each_attr_sum = (float *)malloc(sizeof(float) * world_size);
 assert(each_attr_sum != NULL);

 for(i=0;i<world_size;i++){
 MPI_Gather(&sub_sum, 1, MPI_FLOAT, each_attr_sum, 1, MPI_FLOAT, i,
MPI_COMM_WORLD);
 if (world_rank==i){
 float avg = compute_sum(each_attr_sum, world_size);
 printf("Sum of all elements is %f (from Process %d)\n", avg, world_rank);
 free(randum_nums_generated); // free the memory
 }
 }

 free(each_attr_sum);
 free(sub_randum_nums_generated);
 MPI_Barrier(MPI_COMM_WORLD);
 MPI_Finalize();
}
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>
#include <assert.h>

float *create_randum_nums_generated(int num_elements) {
 float *randum_nums_generated = (float *)malloc(sizeof(float) * num_elements); // allocate memory for the array
 assert(randum_nums_generated != NULL);
 int i;
 for (i = 0; i < num_elements; i++) {
 randum_nums_generated[i] = (rand() / (float)RAND_MAX)*100;
 }
 return randum_nums_generated; // return the array
}

float compute_sum(float *array, int num_elements) {
 float sum = 0.f;
 int i;
 for (i = 0; i < num_elements; i++) {
 sum += array[i]; // compute the sum
 }
 return sum; // return the sum
}

int main(int argc, char** argv) {
 srand(time(NULL));
 int i;
 MPI_Init(NULL, NULL); // initialize MPI
 int world_rank;
 MPI_Comm_rank(MPI_COMM_WORLD, &world_rank); // get the rank of the process
 int world_size;
 MPI_Comm_size(MPI_COMM_WORLD, &world_size); // get the number of processes
```

Output:



```
shara-d@Rohans-Workstation:~/PDC$ cd FAT
shara-d@Rohans-Workstation:~/PDC/FAT$ mpicc p3.c -o p3
shara-d@Rohans-Workstation:~/PDC/FAT$ mpirun -np 4 ./p3
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: Rohans-Workstation
--------------------------------------------------------------------------
Name: Sharadindu Adhikari
Reg. No. 19BCE2105
Randomly Generated Array: 59.330921, 45.107079, 49.551414, 6.801011,

Sum of all elements is 160.790421 (from Process 0)
Sum of all elements is 160.790421 (from Process 1)
Sum of all elements is 160.790421 (from Process 2)
Sum of all elements is 160.790421 (from Process 3)
[Rohans-Workstation:00461] 3 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:00461] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
shara-d@Rohans-Workstation:~/PDC/FAT$ mpicc p3.c -o p3
shara-d@Rohans-Workstation:~/PDC/FAT$ mpirun -np 4 ./p3
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: Rohans-Workstation
--------------------------------------------------------------------------
Name: Sharadindu Adhikari
Reg. No. 19BCE2105
Randomly Generated Array: 89.322113, 70.178749, 94.127724, 53.954868,

Sum of all elements is 307.583435 (from Process 0)
Sum of all elements is 307.583435 (from Process 1)
Sum of all elements is 307.583435 (from Process 2)
Sum of all elements is 307.583435 (from Process 3)
[Rohans-Workstation:00951] 3 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:00951] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
shara-d@Rohans-Workstation:~/PDC/FAT$ 
```

Result:

I have successfully executed the program and I have got identical sum from all the slave processes.