Day13

Table of Contents

1. Scripts

1.1. compile script

```
#!/bin/sh

#source /opt/ohpc/pub/apps/spack/share/spack/setup-env.sh
#spack load gcc/5i5y5cb
#spack load openmpi/c7kvqyq
source ~/git/spack/share/spack/setup-env.sh
spack load openmpi

inputFile=$1
outputFile="${1%.*}.out"      # extract the name of the file without extension and adding extension .out
```

```
#cmd=`mpicc $inputFile -o $outputFile`
cmd="mpicc $inputFile -o $outputFile"     # running code using MPI
echo "------------------------------------------------------------------"
echo "Command executed: $cmd"
echo "------------------------------------------------------------------"
$cmd

echo "Compilation successful. Check at $outputFile"
echo "------------------------------------------------------------------"
```

1.2. run script

```
#!/bin/sh

#source /opt/ohpc/pub/apps/spack/share/spack/setup-env.sh
#spack load gcc/5i5y5cbc
source ~/git/spack/share/spack/setup-env.sh
spack load openmpi

cmd="mpirun -np $2 $1"
echo "------------------------------------------------------------------"
echo "Command executed: $cmd"
echo "------------------------------------------------------------------"
echo "##################################################################"
echo "##########                    OUTPUT                    ##########"
echo "##################################################################"
echo
mpirun -np $2 $1
echo
echo "##################################################################"
echo "##########                     DONE                     ##########"
echo "##################################################################"
```

2. task1

```
#include<stdio.h>
#include<stdlib.h>
#include<mpi.h>
int main(){
    int size, rank;
    MPI_Init(NULL, NULL);
```

```
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    const int n = 10;
    int chunksize = n / size;
    int start = rank * chunksize;
    int end = start + chunksize;
    if(rank == size - 1){
        end = n;
        chunksize += n % size;
    }
    int* arr;
    if(rank == 0){
        arr = (int*)malloc(sizeof(int) * n);
        for(int i = 0; i < n; i++){
                arr[i] = i + 1;
        }
        for(int i = 1; i < size; i++){
            MPI_Send(&arr[i * chunksize], chunksize, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
    }
    else{
        arr = (int*)malloc(sizeof(int) * chunksize);
        MPI_Recv(arr, chunksize, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
    for(int i = 0; i < chunksize; i++){
        printf("%d ", arr[i]);
    }
    printf("\n");

    free(arr);

    MPI_Finalize();
    return 0;
}
```

```
bash compile.sh task1.c
```

```
---------------------------------------------------------------------
Command executed: mpicc task1.c -o task1.out
---------------------------------------------------------------------
Compilation successful. Check at task1.out
---------------------------------------------------------------------
```

```
bash run.sh ./task1.out 4
```

```
--------------------------------------------------------------------
Command executed: mpirun -np 4 ./task1.out
--------------------------------------------------------------------
######################################################################
##########                    OUTPUT                    ##########
######################################################################

3 4
7 8 0 0
1 2
5 6

######################################################################
##########                    DONE                      ##########
######################################################################
```

3. task1 with scatter

```c
#include<stdio.h>
#include<stdlib.h>
#include<mpi.h>
int main(){
    int size, rank;
    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    const int n = 10;
    int chunksize = n / size;
    int* arr;
    int *arr2 = (int*)malloc(sizeof(int) * chunksize);
    if(rank == 0){
        arr = (int*)malloc(sizeof(int) * n);
        for(int i = 0; i < n; i++){
                arr[i] = i + 1;
        }
    }
    MPI_Scatter(arr, chunksize, MPI_INT, arr2, chunksize, MPI_INT, 0, MPI_COMM_WORLD);

    for(int i = 0; i < chunksize; i++){
        printf("%d ", arr2[i]);
```

```
    }
    printf("\n");

    free(arr2);
    if(rank == 0) free(arr);

    MPI_Finalize();
    return 0;
}
```

```
bash compile.sh task1_scatter.c
```

```
---------------------------------------------------------------------
Command executed: mpicc task1_scatter.c -o task1_scatter.out
---------------------------------------------------------------------
Compilation successful. Check at task1_scatter.out
---------------------------------------------------------------------
```

```
bash run.sh ./task1_scatter.out 4
```

```
---------------------------------------------------------------------
Command executed: mpirun -np 4 ./task1_scatter.out
---------------------------------------------------------------------
#####################################################################
##########                   OUTPUT                    ##########
#####################################################################

1 2
3 4
5 6
7 8

#####################################################################
##########                    DONE                     ##########
#####################################################################
```

4. MPI_Scatter

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int n = 10; // Size of the array
    int *array = NULL;
    int chunk_size = n / size;
    int *sub_array = (int*)malloc(chunk_size * sizeof(int));

    if (rank == 0) {
        array = (int*)malloc(n * sizeof(int));
        for (int i = 0; i < n; i++) {
            array[i] = i + 1; // Initialize the array with values 1 to n
        }
    }

    // Scatter the chunks of the array to all processes
    MPI_Scatter(array, chunk_size, MPI_LONG_LONG_INT, sub_array, chunk_size, MPI_LONG_LONG_INT, 0, MPI_COMM_WORLD);

    // Compute the local sum
    long long local_sum = 0;
    for (int i = 0; i < chunk_size; i++) {
        local_sum += (long long)sub_array[i];
    }

    // Gather all local sums to the root process
    long long final_sum = 0;
    MPI_Reduce(&local_sum, &final_sum, 1, MPI_LONG_LONG_INT, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("The total sum of array elements is %lld\n", final_sum);
        free(array);
    }

    free(sub_array);

    MPI_Finalize();
    return 0;
}
```

```
bash compile.sh scatter.c
```

```
---------------------------------------------------------------------
Command executed: mpicc scatter.c -o scatter.out
---------------------------------------------------------------------
Compilation successful. Check at scatter.out
---------------------------------------------------------------------
```

```
bash run.sh ./scatter.out 10
```

```
---------------------------------------------------------------------
Command executed: mpirun -np 10 ./scatter.out
---------------------------------------------------------------------
######################################################################
##########                    OUTPUT                    ##########
######################################################################

The total sum of array elements is -2923367478

######################################################################
##########                     DONE                     ##########
######################################################################
```

5. Scatter sum

```c
#include<stdio.h>
#include<stdlib.h>
#include<mpi.h>
#define N 10000

int main(){

    int size, rank;

    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int chunksize = N / size;
```

```
    int* arr;
    int *arr2 = (int*)malloc(sizeof(int) * chunksize);

    if(rank == 0){
        arr = (int*)malloc(sizeof(int) * N);
        for(int i = 0; i < N; i++){
            arr[i] = i + 1;
        }
    }

    MPI_Scatter(arr, chunksize, MPI_INT, arr2, chunksize, MPI_INT, 0, MPI_COMM_WORLD);

    int localsum = 0;
    for(int i = 0; i < chunksize; i++){
        localsum += arr2[i];
    }

    if (rank != 0){
        MPI_Send(&localsum, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
    }
    else{
        int totalsum = 0;
        totalsum += localsum;
        for(int i = 1; i < size; i++){
            MPI_Recv(&localsum, 1, MPI_INT, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            totalsum += localsum;
        }
        printf("totalsum = %d\n", totalsum);
    }

    free(arr2);
    if(rank == 0) free(arr);

    MPI_Finalize();
    return 0;
}
```

```
bash compile.sh scatter_sum.c
```

```
---------------------------------------------------------------------
Command executed: mpicc scatter_sum.c -o scatter_sum.out
---------------------------------------------------------------------
Compilation successful. Check at scatter_sum.out
---------------------------------------------------------------------
```

```
bash run.sh ./scatter_sum.out 10
```

```
--------------------------------------------------------------------
Command executed: mpirun -np 10 ./scatter_sum.out
--------------------------------------------------------------------
######################################################################
##########                    OUTPUT                    ##########
######################################################################

totalsum = 50005000

######################################################################
##########                     DONE                     ##########
######################################################################
```

6. task2

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

int main() {
    MPI_Init(NULL,NULL);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int N = 2;
    int *array = (int*)malloc(N * sizeof(int));
    int *totarr = NULL;
    for (int i = 0; i < N; i++) {
        array[i] = 1;
    }
    if (rank == 0) {

        totarr = (int*)malloc(N * size * sizeof(int));
    }
    if (rank != 0) {
        MPI_Send(array, N, MPI_INT, 0, 0, MPI_COMM_WORLD);
    } else {
```

```
        for (int i = 0; i < N; i++) {
            totarr[i] = array[i];
        }
        for (int i = 1; i < size; i++) {
            MPI_Recv(&totarr[i * N], N, MPI_INT, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        }
        for (int i = 0; i < N * size; i++) {
            printf("%d\n", totarr[i]);
        }

        free(totarr);
    }
    free(array);
    MPI_Finalize();
    return 0;
}
```

```
bash compile.sh task2.c
```

```
---------------------------------------------------------------------
Command executed: mpicc task2.c -o task2.out
---------------------------------------------------------------------
Compilation successful. Check at task2.out
---------------------------------------------------------------------
```

```
bash run.sh ./task2.out 5
```

```
---------------------------------------------------------------------
Command executed: mpirun -np 5 ./task2.out
---------------------------------------------------------------------
#####################################################################
##########                    OUTPUT                     ##########
#####################################################################

1
1
1
1
1
1
1
1
```

```
1
1

#################################################################
##########                      DONE                     ##########
#################################################################
```

7. MPI_Gather Example

7.1. mpi_gather_example.c

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    int send_data = rank; // Each process sends its rank
    int *recv_data = NULL;
    if (rank == 0) {
        recv_data = (int*)malloc(size * sizeof(int)); // Allocate memory for receiving data
    }
    // Gather the data from all processes to the root process
    MPI_Gather(&send_data, 1, MPI_INT, recv_data, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Gathered data at root process: ");
        for (int i = 0; i < size; i++) {
            printf("%d ", recv_data[i]);
        }
        printf("\n");
        free(recv_data);
    }
    MPI_Finalize();
    return 0;
}
```

7.2. Compilation and Execution

- Compile the program:

```
bash compile.sh mpi_gather.c
```

```
-----------------------------------------------------------------
Command executed: mpicc mpi_gather.c -o mpi_gather.out
-----------------------------------------------------------------
Compilation successful. Check at mpi_gather.out
-----------------------------------------------------------------
```

- Run the program:

```
bash run.sh ./mpi_gather.out 10
```

```
-----------------------------------------------------------------
Command executed: mpirun -np 10 ./mpi_gather.out
-----------------------------------------------------------------
#################################################################
##########                    OUTPUT                   ##########
#################################################################

Gathered data at root process: 0 1 2 3 4 5 6 7 8 9

#################################################################
##########                     DONE                    ##########
#################################################################
```

In this example, each process sends its rank as `send_data`. The `MPI_Gather` function is called to gather the values of `send_data` from all processes to the `recv_data` array in the root process. After gathering the data, the root process prints the gathered values.

7.3. Summary

- **`MPI_Gather`**: Gathers data from all processes in the communicator and collects it at the root process.

7.4. mpi_array_sum_scatter.c

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int n = 100; // Size of the array
    int *array = NULL;
    int chunk_size = n / size;
    int *sub_array = (int*)malloc(chunk_size * sizeof(int));

    if (rank == 0) {
        array = (int*)malloc(n * sizeof(int));
        for (int i = 0; i < n; i++) {
            array[i] = i + 1; // Initialize the array with values 1 to n
        }
    }

    // Scatter the chunks of the array to all processes
    MPI_Scatter(array, chunk_size, MPI_INT, sub_array, chunk_size, MPI_INT, 0, MPI_COMM_WORLD);

    // Compute the local sum
    int local_sum = 0;
    for (int i = 0; i < chunk_size; i++) {
        local_sum += sub_array[i];
    }

    // Gather all local sums to the root process
    int final_sum = 0;
    MPI_Reduce(&local_sum, &final_sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("The total sum of array elements is %d\n", final_sum);
        free(array);
    }

    free(sub_array);
```

```
    MPI_Finalize();
    return 0;
}
```

7.5. Compilation and Execution

- Compile the program:

```
bash compile.sh mpi_array_sum_scatter.c
```

```
---------------------------------------------------------------------
Command executed: mpicc mpi_array_sum_scatter.c -o mpi_array_sum_scatter.out
---------------------------------------------------------------------
Compilation successful. Check at mpi_array_sum_scatter.out
---------------------------------------------------------------------
```

- Run the program:

```
bash run.sh ./mpi_array_sum_scatter.out 10
```

```
---------------------------------------------------------------------
Command executed: mpirun -np 10 ./mpi_array_sum_scatter.out
---------------------------------------------------------------------
#####################################################################
##########                    OUTPUT                     ##########
#####################################################################

The total sum of array elements is 5050

#####################################################################
##########                     DONE                      ##########
#####################################################################
```

8. atharv

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

#define N 100

int main() {
    int rank, size;
    MPI_Init(NULL, NULL);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int chunksize = N / size;
    int* global_arr = NULL;
    int* local_arr = (int*)malloc(chunksize * sizeof(int));

    if (rank == 0) {
        global_arr = (int*)malloc(N * sizeof(int));
        for (int i = 0; i < N; i++) {
            global_arr[i] = i + 1;
        }
    }

    MPI_Scatter(global_arr, chunksize, MPI_INT, local_arr, chunksize, MPI_INT, 0, MPI_COMM_WORLD);

    int local_sum = 0;
    for (int i = 0; i < chunksize; i++) {
        local_sum += local_arr[i];
    }

    int* global_sums = NULL;
    if (rank == 0) {
        global_sums = (int*)malloc(size * sizeof(int));
    }

    MPI_Gather(&local_sum, 1, MPI_INT, global_sums, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        int total_sum = 0;
        printf ("Array of local sums: \n");
        for (int i = 0; i < size; i++) {
            printf("%d ", global_sums[i]);
            total_sum += global_sums[i];
        }
        printf("\nTotal sum = %d\n", total_sum);
        free(global_arr);
        free(global_sums);
```

```
    }

    free(local_arr);
    MPI_Finalize();
    return 0;
}
```

```
bash compile.sh task3.c
```

```
---------------------------------------------------------------
Command executed: mpicc task3.c -o task3.out
---------------------------------------------------------------
Compilation successful. Check at task3.out
---------------------------------------------------------------
```

```
bash run.sh ./task3.out 10
```

```
---------------------------------------------------------------
Command executed: mpirun -np 10 ./task3.out
---------------------------------------------------------------
###############################################################
##########                    OUTPUT                   ##########
###############################################################

Array of local sums:
55 155 255 355 455 555 655 755 855 955
Total sum = 5050

###############################################################
##########                     DONE                    ##########
###############################################################
```

Author: Abhishek Raj
Created: 2025-01-03 Fri 13:57