SA-MIRI 2025

Practice Pc: MPI

Jakub Seliga (jakub.seliga@estudiantat.upc.edu)
Thomas Aubertier (thomas.aubertier@estudiantat.upc.edu)

Task 4.1 Compile and run your first MPI program

- Script launches 4 processes with both mpirun and srun, each prints its rank (0…3)

```
$ sbatch /home/nct/nct01042/2/hello_world_bash.slurm
```

```
[nct01042@alogin1 ~]$ cat mpi_helloworld_30253621.out
[ mpirun ./hello_world_MPI ]
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
MPI startup(): Warning: I_MPI_PMI_LIBRARY will be ignored since the hydra process manager was found
I am 3 of 4
I am 2 of 4
I am 1 of 4
I am 0 of 4
[ srun ./hello_world_MPI ]
I am 2 of 4
I am 1 of 4
I am 3 of 4
I am 0 of 4
[nct01042@alogin1 ~]$
```

Task 4.2 Observe node distribution using hostnames

- Script uses 2 nodes, running 8 tasks on each one of them

```
$ salloc --nodes=2 --ntasks=16
--ntasks-per-node=8 --time=00:30:00
--account=nct_345 --qos=gp_debug bash

$ srun ./hello_world_MPI_2

$ mpirun ./hello_world_MPI_2
```

```
[nct01042@alogin1 ~]$ mpirun ./hello_world_MPI_2
I am 1 of 16 running on gs06r1b03
I am 2 of 16 running on gs06r1b03
I am 3 of 16 running on gs06r1b03
I am 5 of 16 running on gs06r1b03
I am 6 of 16 running on gs06r1b03
I am 7 of 16 running on gs06r1b03
I am 4 of 16 running on gs06r1b03
I am 0 of 16 running on gs06r1b03
I am 8 of 16 running on gs06r1b28
I am 9 of 16 running on gs06r1b28
I am 10 of 16 running on gs06r1b28
I am 11 of 16 running on gs06r1b28
I am 12 of 16 running on gs06r1b28
I am 13 of 16 running on gs06r1b28
I am 15 of 16 running on gs06r1b28
I am 14 of 16 running on gs06r1b28
[nct01042@alogin1 ~]$ srun ./hello_world_MPI_2
I am 3 of 16 running on gs06r1b03
I am 6 of 16 running on gs06r1b03
I am 4 of 16 running on gs06r1b03
I am 7 of 16 running on gs06r1b03
I am 5 of 16 running on gs06r1b03
I am 1 of 16 running on gs06r1b03
I am 2 of 16 running on gs06r1b03
I am 0 of 16 running on gs06r1b03
I am 11 of 16 running on gs06r1b28
I am 8 of 16 running on gs06r1b28
I am 15 of 16 running on gs06r1b28
I am 12 of 16 running on gs06r1b28
I am 9 of 16 running on gs06r1b28
I am 10 of 16 running on gs06r1b28
```

Task 4.3 Point-to-point communication

- Script calls MPI_Recv in increasing order of ranks, so messages are printed in rank order, regardless of when they arrive

```c
if (rank == 0)
{
    for(int source = 1; source < size; source++)
    {
        MPI_Recv(message, 100, MPI_CHAR, source, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("Process %d received: \"%s\"\n", rank, message);
    }
}
else
{
    sprintf(message, "Hello I am process %d and I love u UwU",rank);
    MPI_Send(message, strlen(message) + 1, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
}
```

```
$ srun ./send_recv_MPI
```

```
[nct01042@alogin1 ~]$ srun ./send_recv_MPI
Process 0 received: "Hello I am process 1 and I love u UwU"
Process 0 received: "Hello I am process 2 and I love u UwU"
Process 0 received: "Hello I am process 3 and I love u UwU"
Process 0 received: "Hello I am process 4 and I love u UwU"
Process 0 received: "Hello I am process 5 and I love u UwU"
Process 0 received: "Hello I am process 6 and I love u UwU"
Process 0 received: "Hello I am process 7 and I love u UwU"
Process 0 received: "Hello I am process 8 and I love u UwU"
Process 0 received: "Hello I am process 9 and I love u UwU"
Process 0 received: "Hello I am process 10 and I love u UwU"
Process 0 received: "Hello I am process 11 and I love u UwU"
Process 0 received: "Hello I am process 12 and I love u UwU"
Process 0 received: "Hello I am process 13 and I love u UwU"
Process 0 received: "Hello I am process 14 and I love u UwU"
Process 0 received: "Hello I am process 15 and I love u UwU"
```

Task 4.4 Write and Run The Sequential Program That Estimate π

```c
double f(double x)
{
    return 4.0 / (1.0 + x*x);
}

double trap_rule(double a, double b, int n)
{
    double h = (b-a)/n;
    double sum = (f(a) + f(b))/2.0;

    for(int i = 1; i < n; i++)
        sum += f(a+i*h);

    return sum*h;
}

int main(int argc, char **argv)
{
    double a = 0.0, b = 1.0;
    double pi = trap_rule(a,b,N);
    printf("Estimated PI = %.16f\n", pi);
    return 0;
}
```

- Program uses trapezoidal rule to estimate value of π

```
PS C:\Users\kubas\Downloads\SA-MIRI-main\2> ./pi
Estimated PI = 3.1415926535897927
```

Task 4.5 Write and Run the Parallel MPI Code to Estimate the Value of π

- Similar program to estimate π, but as the parallel version using 16 processes with SLURM

```
$ srun ./pi_MPI
```

```
[nct01042@alogin1 ~]$ srun ./pi_MPI
Estimated value of pi = 3.141592653589781
```

Task 4.6 Scalability Analysis of the MPI Trapezoidal Rule

- Script to compare execution times with sequential calculations and parallel parallel program with different numbers of processes (2, 4, 8, 16, 32, 64)

```
$ icx pi_seq_timed.c -o pi_seq_timed -lm
$ mpiicx pi_mpi_timed.c -o pi_mpi_timed -lm
$ sbatch pi_mpi_scaling.slurm
```

```
[nct01042@alogin1 2]$ cat pi_scaling_30314932.out
=== COMPILING ===
=== RUNNING SEQUENTIAL ===
Estimated PI = 3.1415926535897571
Sequential execution time: 4435.21 ms

=== RUNNING WITH 2 PROCESSES ===
Estimated PI = 3.1415926535897931
Parallel execution time with 2 processes: 2383.88 ms

=== RUNNING WITH 4 PROCESSES ===
Estimated PI = 3.1415926535897389
Parallel execution time with 4 processes: 1200.43 ms

=== RUNNING WITH 8 PROCESSES ===
Estimated PI = 3.1415926535897643
Parallel execution time with 8 processes: 609.53 ms

=== RUNNING WITH 16 PROCESSES ===
Estimated PI = 3.1415926535897740
Parallel execution time with 16 processes: 310.45 ms

=== RUNNING WITH 32 PROCESSES ===
Estimated PI = 3.1415926535897754
Parallel execution time with 32 processes: 155.08 ms

=== RUNNING WITH 64 PROCESSES ===
Estimated PI = 3.1415926535897989
Parallel execution time with 64 processes: 83.55 ms
```

Task 4.6 Scalability Analysis of the MPI Trapezoidal Rule

- More processes -> time drops and speedup grows, but efficiency slowly decreases. After about 32 processes, gains become smaller
- Possible reason: communication overhead (more processes need to exchange data) and limited parallelism (some work can’t be split further)
- Still no major drops up to 64 processes

Processes	Time (ms)	Speedup	Efficiency (%)
seq	4435,21	1,00	100,0%
2	2383,88	1,86	93,0%
4	1200,43	3,69	92,4%
8	609,53	7,28	91,0%
16	310,45	14,29	89,3%
32	155,08	28,60	89,4%
64	83,55	53,08	82,9%

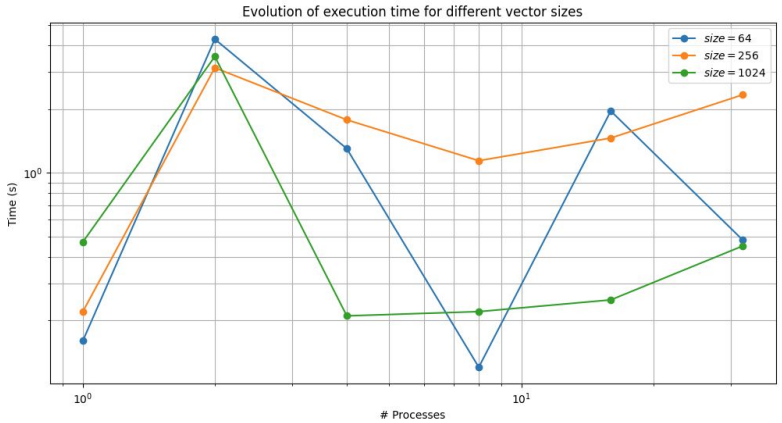
Task 4.7 Experimenting with Scatter and Gather

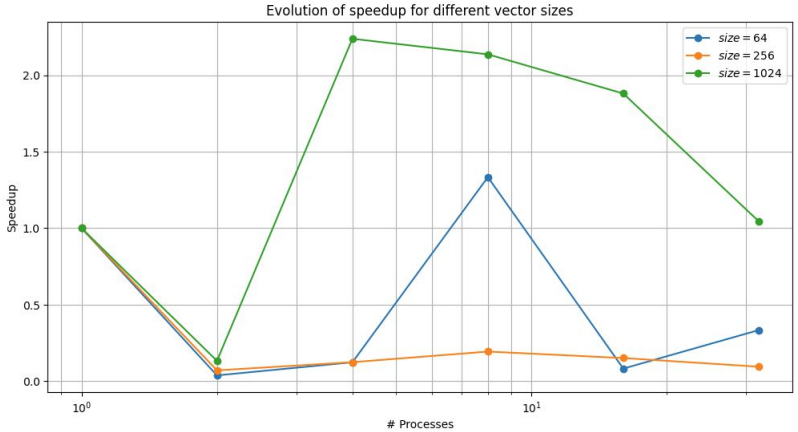
- Program compiles and executes correctly.

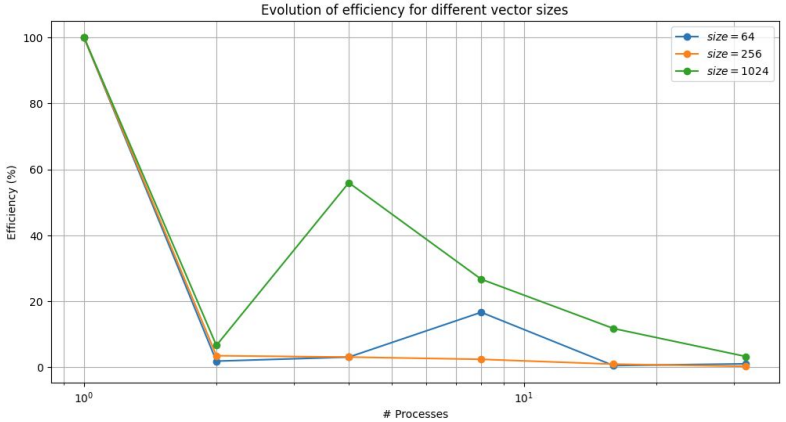
```
$ sbatch /home/nct/nct01042/2/mpi_vector_sum.slurm
```

```
[nct01042@gs04r1b66 2]$ mpiicx mpi_vector_sum.c -o mpi_vector_sum
[nct01042@gs04r1b66 2]$ srun ./mpi_vector_sum
Root process: initial vector = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Process 0 received elements: 1 2 3 4
Process 1 received elements: 5 6 7 8
Process 3 received elements: 13 14 15 16
Process 2 received elements: 9 10 11 12
Root process: gathered result = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
```

Task 4.7 Experimenting with Scatter and Gather







- Execution times for parallelised versions seemed to be very unstable on `gp_debug` even after several tries.
- Even on best cases, the efficiency did not got above 60%.