

Project 1 Report for Problem 1.1

Zhu Liang

September 14, 2023

1 Project Description

The objective of this project is to write a parallel program in C that prints “Hello from <rank> of <size> processors.” in order by processors’ rank. Test it for a system with at least 13 processors.

2 Algorithm Description

This program utilizes the MPI parallel programming library to execute the code in parallel. The primary objective of the program is to have each processor print its identifier in order. To ensure the printing is in order, the program employs the `MPI_Barrier` function to synchronize all processes. Then, each process will parallelly run the same for-loop and only print its identifier when its rank matches the loop index. After printing, the program will synchronize all processes again to ensure the printing is in order.

The pseudo-code of the program `hello.c` is shown in Algorithm 1. This program is based on the coding from lecture slides [1] and compiled by the shell script `project1.sh`.

Algorithm 1 Fix Hello Order

```
1: procedure FIXHELLOORDER
2:   Initialize MPI environment
3:   Get rank and size of processes
4:   Synchronize all processes
5:   for each process do
6:     if current process rank matches loop index then
7:       Print “Hello from <current rank> of <size> processors.”
8:     end if
9:     Synchronize all processes
10:  end for
11:  Finalize MPI environment
12: end procedure
```

3 Results

The program was executed on a `short-28cores` node by the shell script `project1.sh`. The output of the program is shown below.

```
1 Hello from 0 of 14 processors.
2 Hello from 1 of 14 processors.
3 Hello from 2 of 14 processors.
4 Hello from 3 of 14 processors.
5 Hello from 4 of 14 processors.
6 Hello from 5 of 14 processors.
7 Hello from 6 of 14 processors.
8 Hello from 7 of 14 processors.
9 Hello from 8 of 14 processors.
10 Hello from 9 of 14 processors.
11 Hello from 10 of 14 processors.
12 Hello from 11 of 14 processors.
13 Hello from 12 of 14 processors.
14 Hello from 13 of 14 processors.
15
16 real 0m0.655s
17 user 0m0.476s
18 sys 0m5.228s
```

4 Analysis

The program successfully prints the identifier of each process in order. The program can be executed on a system with at least 13 processors.

The program runs the same for-loop on each process and synchronizes the processes after each iteration. It ensures that each process will print its identifier in order. The program is parallel because each process runs the same for-loop in parallel. Although it is not the most efficient in terms of performance, it is a simple way to solve the problem.

File Notes

The source code of the program is in the `project1` folder. The source code file is named `hello.c`. The shell script file is named `project1.sh`. To compile and run the program, run the shell script `project1.sh` in Seawulf. The output will be printed on `output.txt`

References

- [1] Yuefan Deng. *Principles of Parallel Computing: High-performance Computing and Algorithms for Big Data, Topic 3 Software & MPI*. 2023, p.30.