Project 1 Report for Probelm 1.1

Zhu Liang

September 26, 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

```
procedure FixHelloOrder
    Initialize MPI environment
    Get rank and size of processes
    Synchronize all processes
    for each process do
        if current process rank matches loop index then
            Print "Hello from <current rank> of <size> processors."
        end if
        Synchronize all processes
    end for
    Finalize MPI environment
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.

```
    Hello from 0 of 28 processors.
    Hello from 1 of 28 processors.
    Hello from 2 of 28 processors.
    Hello from 3 of 28 processors.
    Hello from 4 of 28 processors.
    Hello from 5 of 28 processors.
    Hello from 6 of 28 processors.
    Hello from 7 of 28 processors.
    Hello from 8 of 28 processors.
    Hello from 9 of 28 processors.
    Hello from 10 of 28 processors.
    Hello from 11 of 28 processors.
    Hello from 12 of 28 processors.
    Hello from 13 of 28 processors.
    Hello from 14 of 28 processors.
    Hello from 15 of 28 processors.
    Hello from 16 of 28 processors.
    Hello from 17 of 28 processors.
    Hello from 18 of 28 processors.
    Hello from 19 of 28 processors.
    Hello from 20 of 28 processors.
    Hello from 21 of 28 processors.
    Hello from 22 of 28 processors.
    Hello from 23 of 28 processors.
    Hello from 24 of 28 processors.
    Hello from 25 of 28 processors.
    Hello from 26 of 28 processors.
    Hello from 27 of 28 processors.

    real  0m1.055s
    user  0m0.931s
    sys 0m15.947s
```

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 run the same for-loop on each process and synchronize 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 term 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`

For more details, please refer to the `README.md` file in the `project1` folder.

References

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