Mustafa Can AYDIN

MPI Project

Introduction

This project's aim is to simulate parallel programming interface for relief algorithm. In this project I implement an algorithm called relief with MPI interface. With the help of MPI interface I will be simulating multiple processor environment working parallel to solve relief algorithm as fast as possible.

Program Execution

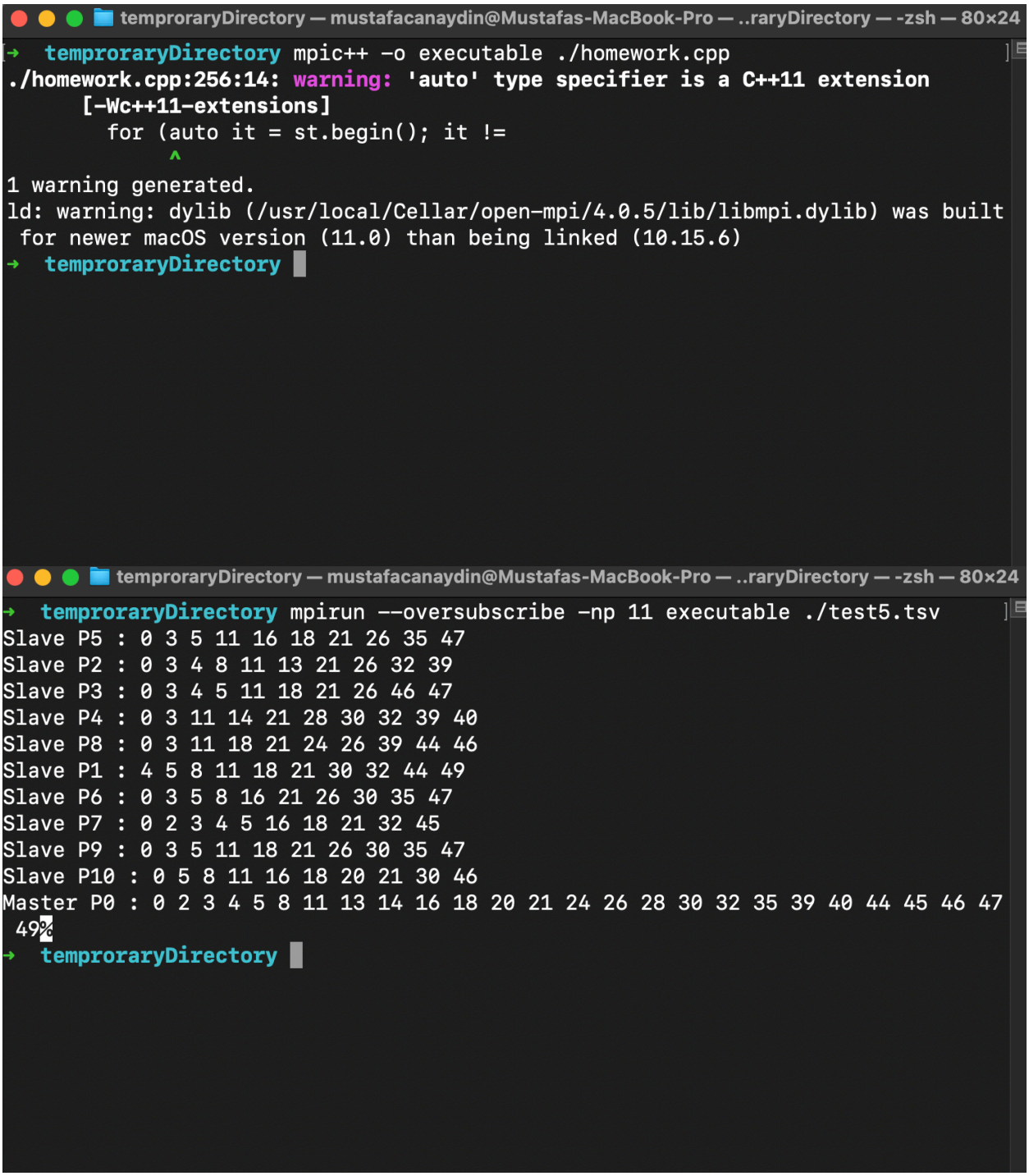
Program execution consists of two steps first one being compilation and the second one being running of the compiled C++ code using MPI interface tags. The C++ compiler that I use for this project is Apple clang version 12.0.0 (clang-1200.0.32.2), target: x86_64-apple-darwin20.2.0 and the environment is MacOS BigSur version 11.1.
Compilation Step:

```
mpic++ –o executable ./homework.cpp
```

Running Step :

```
mpirun ––oversubscribe –np Number executable
./test.tsv
```

Number variable is the number of processors that the algorithm will be using.
Examples of compile and run outputs in terminal:
(test5.tsv file is equivalent to test case provided with the name mpi_project_dev2.tsv)



Program Structure

In order to handle parallel algorithm, I used MPI interface's initialization and ranking commands which are `MPI_Init(&argc, &argv);` and `MPI_Comm_rank(MPI_COMM_WORLD, &rank);` respectively. `MPI_Init(&argc, &argv);` command will create as much processors as given as a argument while running the program and ranking command will help

us keep track of which processor is currently executing to which part of the program. We will have 1 master and N-1 slave processors. Master will share given data with slaves and slave processors will execute the algorithm on given data. This way relief algorithm will be run concurrently by the slave processors which will help reduce the time of execution. After the slave processors complete relief algorithm, they will send their results to master and master will unite all the results and print the final unified result to the console. In order to communicate with the slaves master processor uses mpi interface commands of scatter, send , and receive. Master processor send necessary information to slaves using MPI_Send() and receives from its slaves using the command MPI_Recv(). Scattering the data to slave processors is done using the command MPI_Scatter().

Difficulties Encountered

There were difficulties when ensuring transmission of the information master shares with its slaves. Also, each slave processor's message should be received by master without loss. After ensuring scattering and send/receive parts, synching was important for the master to finish after slaves. After each difficulty overcame, correct implementation of the relief algorithm ended up with true results.

Conclusion

It was a challenging and important project. Parallel programming is being used more and more in real world applications. Using parallel computation's power in important real-world application with relief algorithm gave me important insight about its importance and implementation methodology.