CmpE300 Analysis of Algorithms

Project 2-MPI

Boğaziçi University

Fall-2023

DOCUMENTATION REPORT

Soner Kuyar

soner.kuyar@boun.edu.tr

2021400297

Yunus Kağan Aydın

yunus.aydin1@boun.edu.tr

2021400123

Contents

1-Introduction

In this project, we are assigned as hypothetical managers of a company. In order to adjust Industry 4.0 goals, we are asked to create a digital version of our factory. This factory consists of various machines that has numerous types of operations. While doing these operations, our company must be strict to some maintenance rules for each machine. We are also asked to struct a communication system between machines with parallel algorithm. MPI (Message Passing Interface) will be implemented for increased efficiency. The advantages of parallel programming for this project is concurrent programming. By concurrent programming we are able to execute processes, send and receive messages at the same time and our time overhead will be reduced by a reasonable amount.

2-Program Interface

This project is written in python language so correct version of python should be downloaded. Program will be executed by this command:

mpiexec -n 1 python control_room.py input.txt output.txt

IMPORTANT: If the Operating System(like MacOS) doesn't allow the create processes more than the number of processors in the computer, you should add "—oversubscribe" flag and command will be

mpiexec -n 1 –oversubscribe python control_room.py input.txt output.txt

3-Program Execution

In input.txt, first line consists of number of machines.

Second line consists of production cycles.

Third line states wear factor for each operation (enhance, reverse, chop, trim, split). Add operation doesn't have a wear factor.

Fourth line consists of threshold value for maintenance.

When we say N as number of machines, next N-1 lines are the adjacency list of machines. These lists contain child machine's id, parent machine's id, and the name of the operation that will be executed.

Remaining lines are the products that will be given to leaf machines.

In output.txt, firstly last product will be written as output. Then logs will be printed line by line.

4-Program Structure

4.1-control_room.py file

Firstly, we gathered input and output files from given argument and assigned number of machines, cycles number, wear factors for each operation, threshold value, machine tree and products. While reading machine tree we differentiate machines if they have parent in tree or not. If they have a parent, we assign that machine as the child of that machine. Then we add products to products list.

Secondly, we spawned N number of machines(processes) which are stated as machine.py. Then we broadcasted wear factors, number of machines, number of cycles and threshold value to machines. After that we sent machine trees and products to machines.

Thirdly, we enter into the loop of cycles and execute loop as the number of given cycles. In this loop we receive logs and when a log is received and if it is available it is appended to the list and written in the output file.

4.2-machine.py file

Machine is a class to determine the specific operations in a machine. It has initialized by numerous features such as: id, tree, parente, operation, children wear_factors, threshold, accumulated_wear and cycle number. The class implements 4 functions called as follows: *machine_operation_cycle()*, *_apply_operation()* , *_calculate_accumulated_wear()* and *_change_operation().*

In *machine_operation_cycle*, operations are applied to product by *_apply_operation* function and maintenance cost is calculated by *_calculate_accumulated_wear* if threshold level is exceeded and a log will be created. After this operation will be changed by *_change_operation* and the product is returned.

In *_apply_operation*, operations are being applied to products returns processed product.

In *_calculate_accumulated_wear*, maintenance cost is returned if threshold is exceeded.

In *_change_operation*, type of operations are switched after the operation was processed.

After these functions, wear factors, number of machines, number of cycles and threshold values are broadcasted from control room. Then machine tree is received from control room.

Lastly, a machine instance is created with arguments: machine_id, machine_tree, wear_factors and threshold. If machine doesn't have any children (if it is a leaf machine), initial products will be given. Then a while loop is created by the number of cycles. If the machine is a leaf machine products will be operated and given to the parent. If the machine is not a leaf machine than it will receive products from the child machine. This process will continue until the program reaches to root machine. When the program reaches the root machine, last product will be sent to control room and that cycle is finished program goes to upper cycle.

5-Mock Example

In the description file, we are asked to implement a mock input that has at least 7 machines in it and show the first production cycle using diagrams. Here's an example:

mock_input:

8

6

3 2 1 5 2

15

2 1 split

3 2 trim

4 2 enhance

5 3 reverse

6 3 split

7 4 trim

8 4 enhance

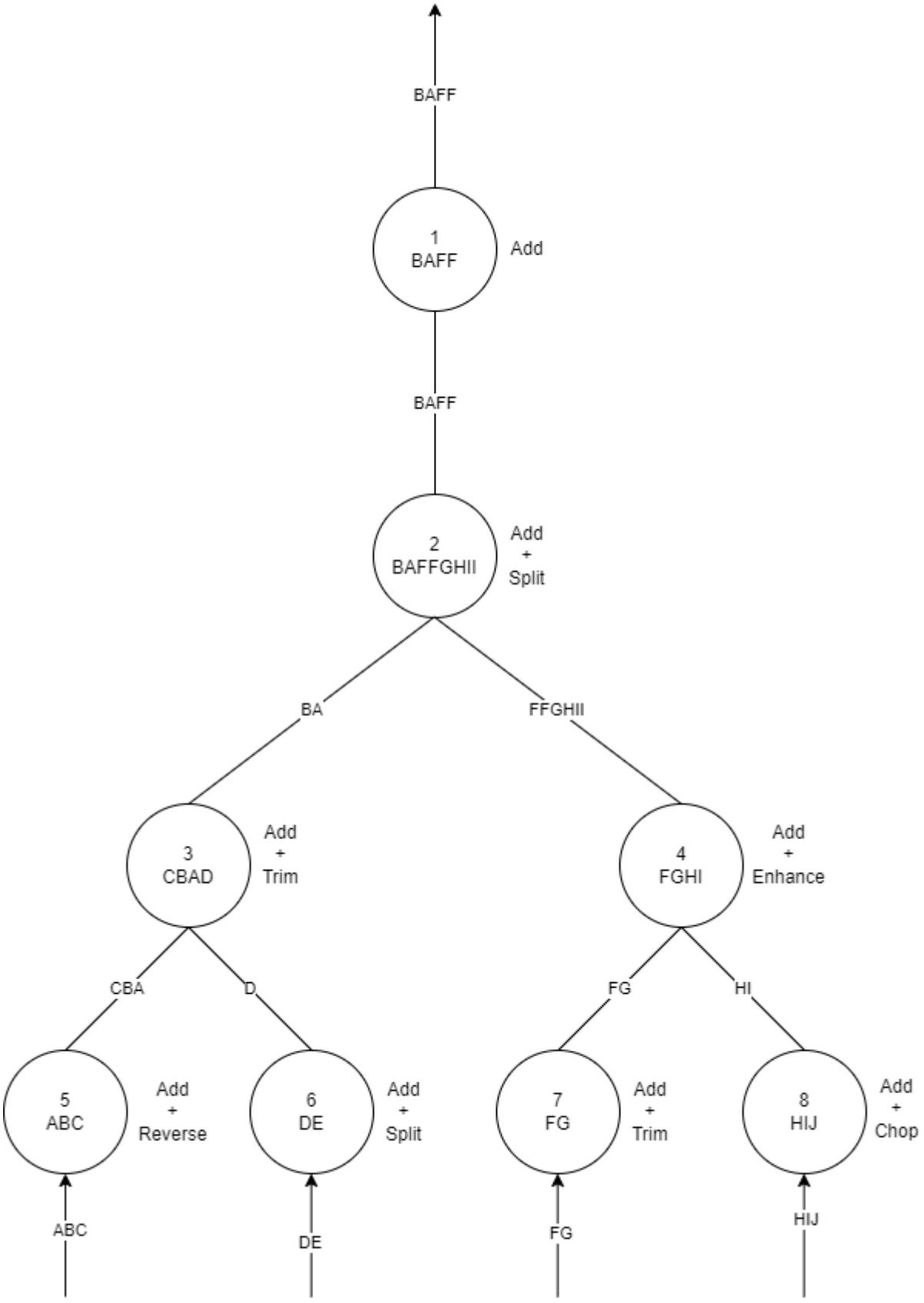
ABC

DE

FG

HIJ

Diagram of the mock input:



6-Difficulties Encountered and Methods

In this project coordinating multiple processes were challenging both in terms of the efficiency and scalability. We tried to keep process memories as simple as possible. By applying object-oriented method for each machine, we kept the machine configuration in compact structure.

Blocking communication was applied during the production process but machines sent their waring informations in a non-blocking way with particular message tag. After the final product has arrived control_room collected the logs of the machines.

We also pass the common data fields of the machines by broadcasting with numpy arrays. This method also increased the efficiency.

7-Conclusion

By this project we examined how to use MPI in parallel programming. Creation of parallel processes with MPI in our factory increased the efficiency and it gave an option to have communication between processes.

8-Bonus

If we would like to apply this digital twinning project to real life, our products won't be strings anymore and this would cause difficulties in sampling real life data especially in physical quantities.

But there is more, machines in production cycles may not age with a constant number in real life .Due to humidity, temperature, battery lifes, and other physical indeterminable things in production line, real time rules are quite complex and sometimes impossible. We've made a little research about this subject. NASA might be the amiral institution of simulating and twinning real life scenarios on digital environment. 6 months ago, they published prognostic algorithms that they are use in prediction of conditions here. The result is we cannot be exactly confident about most of the real life physics rules in digital environment. It includes friction between machine parts, state of health of electronic devices and batteries, etc... On the other hand, machines never have 100% reliability, they may produce unwanted or false results.

Lastly, using a machine infinitely by applying maintenance rules seems quite unrealistic in real life production cycles and again prediction of when the machines will be useless is just an approximation.