INTRODUCTION TO PARALLEL COMPUTING

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

DEPARTMENT OF INFORMATION AND COMPUTER ENGINEERING

MPI COLLECTIVE AND BROADCAST COMMUNICATION

STUDENT DETAILS

NAME: ATHANASIOU VASILIOS
EVANGELOS REGISTRATION NUMBER: 19390005
STUDENT **SEMESTER:** 7th
STUDENT STATUS : UNDERGRADUATE
STUDY PROGRAM : PADA
LABORATORY DEPARTMENT : E3 MONDAY 2:00 PM – 4:00 PM
LABORATORY TEACHER : MICHAEL IORDANAKIS
DELIVERY DATE : 1/15/2023

STUDENT PHOTO:

The point

The exercise aims to achieve collective communication between " p " processes given a vector "X" of size " N ". In more detail, a process is defined as a manager and is responsible for distributing the elements of the vector to the processors and receiving the information it wants. For example, how many elements have a value greater than the mean and how many less (question a), what is the dispersion of the elements of X (question b), what are the elements of the percentage relationship vector of the corresponding Xi with the maximum-minimum difference of the values of all the elements of X (question c), what is the largest value of the vector Δ and for which element xi in particular is observed (question d) and what is the vector of prefixes of the sum of the elements of X (question e). This process will run iteratively with a menu of options to continue or exit the process. The choice will be given by the user.

The problem and the implementation in natural language

The process " P 0" takes on the duties of "manager", that is, it will distribute the elements equally to the other processes, so that it also gets the same number of elements. The calculations are done locally at the beginning and with collective communication functions the total demand is calculated.

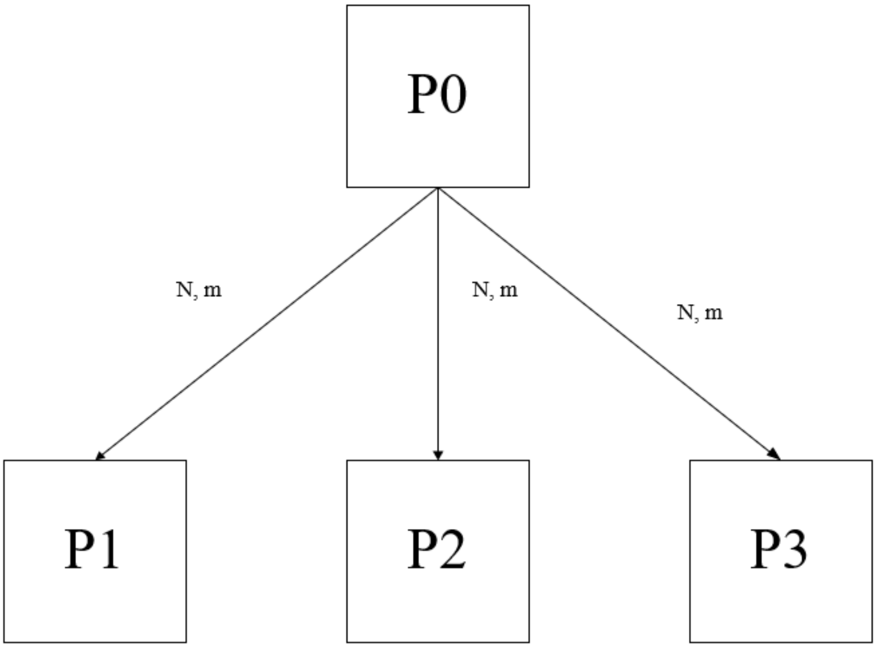

Figure 1. The manager process (P 0) sends the same data to the other processes (MPI _ Bcast)

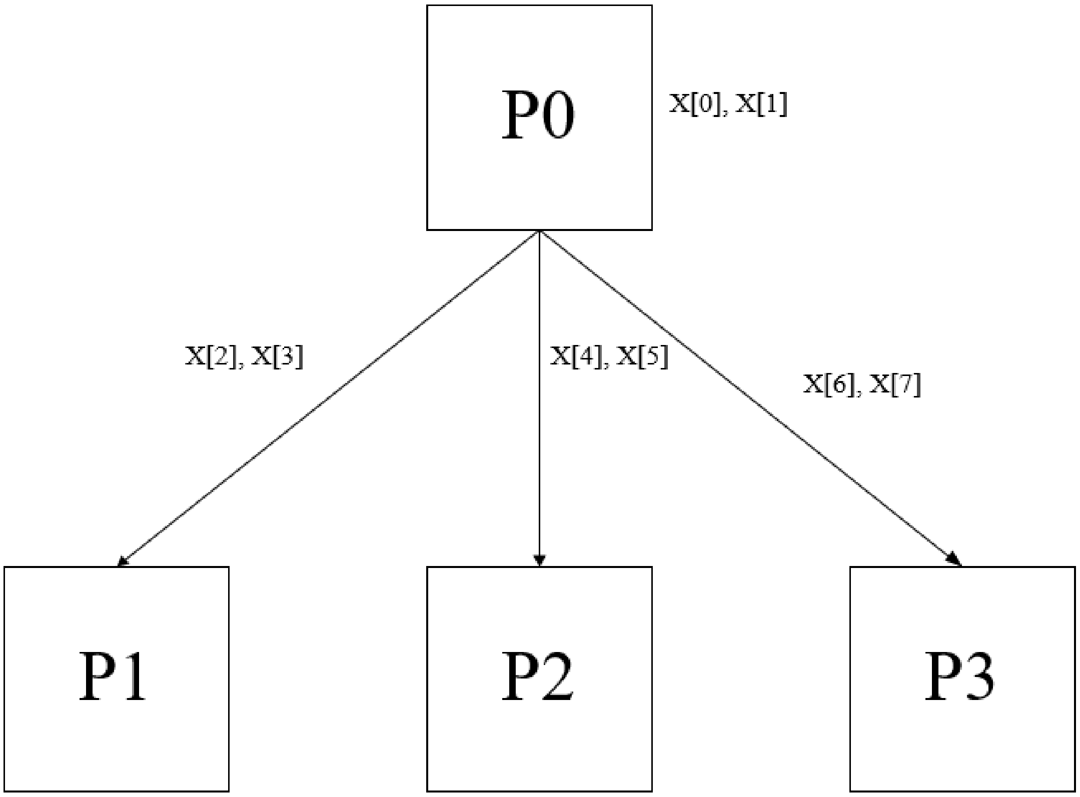

Figure 2. The manager process (P 0) distributes the elements of X to the other processes (MPI _ Scatter)

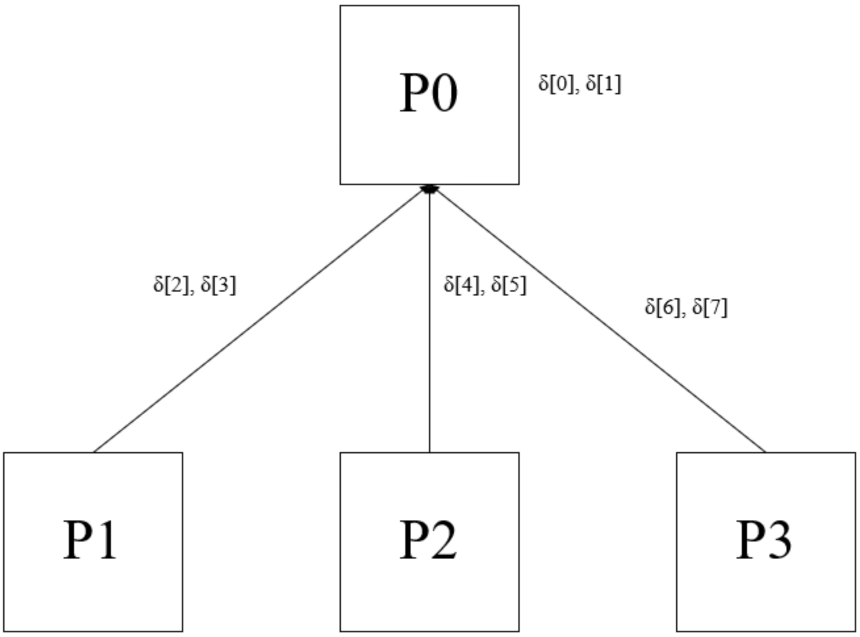

Figure 3. The manager process (P 0) collects the elements δ[i] from the remaining processes (MPI _ Gather)

The computational load of each process

Communication is collective, that is, all processes call the same function that the data is managed by the manager process. Each process computes locally based on the data it owns and the manager process prints the results along with memory allocations to store the data.

The problem and implementation in C language

To implement the request, the MPI environment that achieves parallel calculation and the C language was used . The MPI routines used are MPI_Init , MPI_Comm_rank , MPI_Comm_size , MPI_Send , MPI_Recv , MPI_Bcast , MPI_Gather , MPI_Reduce , MPI_Barrier , MPI_Scan and MPI _ Finalize . Communication between processes is collective and " point - to - point " blocking for the purposes of query (e). The program is analyzed in the comments of the source code project 2. c .

Difficulties

My difficulty was in allocating the data for n < p as I couldn't figure out how to distribute the elements to the processors as inevitably some processors would have no elements. Therefore, I was unable to extend the program for any combination of n and p values , so I limited myself to n being an integer multiple of p .

Indicative Runs

Compile: mpicc - o project 2 project 2. c

mpicc -o collective_communication collective_communication.c

Example 1 n > p (mpirun -np 4 ./project2) mpirun -np 4 ./collective_communication

```
Number of processors are 4

Size of integers' sequence must be integer multiple of number of processors (N mod processors == 0).

Input the size of integers' sequence : 8

Input elements of vector X

X[ 0] : 1

X[ 1] : 6

X[ 2] : 7

X[ 3] : 9

X[ 4] : 10

X[ 5] : 4

X[ 6] : 8
```

```
X[ 7] : 2

Rank : 0, X[0] : 1

Rank : 2, X[0] : 10

Rank : 2, X[1] : 4

Rank : 3, X[0] : 8

Rank : 1, X[0] : 7

Rank : 1, X[1] : 9

Rank : 3, X[1] : 2

Rank : 0, X[1] : 6

--------------------------------------

Question A

Number of elements that are greater than average 5.000000 : 5

Number of elements that are less than average 5.000000 : 3

--------------------------------------

Question B

var = ((X0 - m)^ 2 + (X1 - m)^2 + ... + (Xn-1 - m)^2) / n

var : 10.125000

--------------------------------------

Question C

Di = ((Xi - Xmin ) / ( Xmax - Xmin )) * 100

D[ 0 ] : 0.000000

D[ 1] : 55.555557

D[ 2] : 66.666672

D[ 3] : 88.888893

D[ 4] : 100.000000

D[ 5] : 33.333336

D[ 6] : 77.777779

D[ 7] : 11.111112

--------------------------------------

Question D

Element's index of vector X with the highest D[ i ] : 4

Vector X's element with the highest D[ i ] : 10
```

```
Max D[ i ] : 100.000000
--------------------------------------
Question E
Prefix[ 0] : 1
Prefix[ 1] : 7
Prefix[ 2] : 14
Prefix[ 3] : 23
Prefix[ 4] : 33
Prefix[ 5] : 37
Prefix[ 6] : 45
Prefix[ 7] : 47

[1] Continue...
[2] Exit...
```

Example 2 n == p (mpirun -np 3 ./project2) mpirun -np 3 ./collective_communication

```
Number of processors are 3
Size of integers' sequence must be integer multiple of number of processors (N mod processors == 0).
Input the size of integers' sequence : 3
Input elements of vector X
X[ 0] : 4
X[ 1] : 10
X[ 2] : 14
Rank : 0, X[0] : 4
Rank : 1, X[0] : 10
Rank : 2, X[0] : 14
--------------------------------------
Question A
Number of elements that are greater than average 9.000000 : 2
Number of elements that are less than average 9.000000 : 1
--------------------------------------
```

```
Question B

var = ((X0 - m)^ 2 + (X1 - m)^2 + ... + (Xn-1 - m)^2) / n

var : 17.000000

--------------------------------------

Question C

Di = ((Xi - Xmin ) / ( Xmax - Xmin )) * 100

D[ 0 ] : 0.000000

D[ 1] : 60.000004

D[ 2 ] : 100.000000

--------------------------------------

Question D

Element's index of vector X with the highest D[ i ] : 2

Vector X's element with the highest D[ i ] : 14

Max D[ i ] : 100.000000

--------------------------------------

Question E

Prefix[ 0 ] : 4

Prefix[ 1] : 14

Prefix[ 2] : 28


[1] Continue...

[2] Exit...

Input a choice:
```

modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier

Thank you for your attention.