Logo

Description automatically generated

**Ain Shams University**

**Faculty of Engineering**

**Computer and Systems Engineering Department**

**CSE411: Distributed Computer Systems**

**Assignment 3**

**Submitted by:**

**Mariam mohamed reda ahmed mohamed**

**ID: 1701405**

**Table of Contents**

[**introduction 1**](#_Toc90299180)

[**The Program Idea 1**](#_Toc90299181)

[**description of the solution 2**](#_Toc90299182)

[**The Results, and Difference in Performance 3**](#_Toc90299183)

[**compare the time for parallel version to sequential version 3**](#_Toc90299184)

[**number of computing processes changes. 4**](#_Toc90299185)

**Table of figures**

[**Figure 1 taking inputs from user 2**](#_Toc90299374)

[**Figure 2:local start and local end 2**](#_Toc90299375)

[**Figure 3:last calculation 3**](#_Toc90299376)

[**Figure 4:sequential version with upper limit of i = 5000 3**](#_Toc90299377)

[**Figure 5:parallel version with upper limit of i = 5000 4**](#_Toc90299378)

[**Figure 6:parallel version with number of processes = 1 4**](#_Toc90299379)

[**Figure 7:parallel version with number of processes = 4 5**](#_Toc90299380)

[**Figure 8:parallel version with number of processes = 6 5**](#_Toc90299381)

# introduction

This report shows how the different nodes communicate with each other,

via message passing, by giving example, which is using multiple processes to calculate cos(x) value, using the MPI (**Message Passing Interface**) library.

Message Passing Interface is a standardized and portable message-passing standard designed to function on parallel computing architectures

**Here is some of the main functions in MPI (C Function Call):**

1. int **MPI\_Init**(int \*argc, char \*\*argv)used to Initialize MPI
2. int **MPI\_Comm\_size**(MPI\_Comm comm, int \*size) used to Determine number of processes within  a communicator
3. int **MPI\_Comm\_rank**(MPI\_Comm comm, int \*rank) used to Determine processor rank within a communicator
4. int **MPI\_Finalize**() used to Exit MPI (must be called last by all processors)
5. int **MPI\_Send** (void \*buf,int count, MPI\_Datatype  
   datatype, int dest, int tag, MPI\_Comm comm) used to Send a message
6. int **MPI\_Recv** (void \*buf,int count, MPI\_Datatype  
   datatype, int source, int tag, MPI\_Comm comm, MPI\_Status \*status) used to Receive a message

# The Program Idea

We need to write a C program that uses MPI parallelization to compute the value of cos(x) using this formula 𝑐𝑜𝑠(𝑥) .

By getting the upper value of i and the value of x from the user, where the program makes the computation by dividing i equally among the processes, it should use n processes to do this computation, where n is provided as input by user. Then, it displays the computed value of cos(x), time taken by the program to compute it.

# description of the solution

1. By making the process with rank = 0 take the inputs ang sending theme to the other processes as shown in figure 1

*Text

Description automatically generated*

Figure 1 taking inputs from user

1. Calculate the local start and local end for each process and make each process calculate it’s part of cos function

*Text

Description automatically generated*

Figure 2:local start and local end

1. Make each process send it’s result to the process with rank =0 So this process can calculate the sum

Text

Description automatically generated

Figure 3:last calculation

# The Results, and Difference in Performance

## compare the time for parallel version to sequential version

**with upper value of I = 5000 and x= 1.0471975.**

sequential version:

Text

Description automatically generated

Figure 4:sequential version with upper limit of i = 5000

parallel version:

Text

Description automatically generated

Figure 5:parallel version with upper limit of i = 5000

Time of parallel version is less than time of sequential version as shown.

## number of computing processes changes.

**discuss the differences in these times as the number of computing processes changes.**

Number of processes = 1 with I = 500000 and x= 1.0471975

*Graphical user interface, text

Description automatically generated*

Figure 6:parallel version with number of processes = 1

Number of processes = 4 with I = 500000 and x= 1.0471975

Text

Description automatically generated

Figure 7:parallel version with number of processes = 4

Number of processes = 6 with I = 500000 and x= 1.0471975

*Text

Description automatically generated*

Figure 8:parallel version with number of processes = 6

As shown, by increasing the number of processes the time required for calculation decrease while all resulting in the same answer, which is expected.