Лабораторна робота №3

Тема роботи: Вивчення MPI-функцій попарного обміну повідомленнями.

Мета роботи: Вивчення та визначення ефективності функцій попарного обміну повідомленнями в MPI і роботи з часом.

3.1 Основні теоретичні відомості

При складанні програм мовою C++ з використанням середовища передавання повідомлень MPI слід користуватися наступним алгоритмом для реалізації програмного коду:

1. Підключення бібліотек;
2. Оголошення змінних в програмі;
3. Фрагмент коду програми на C++;
4. Ініціалізація паралельної частини програми (MPI_Init);
5. Визначення кількості процесів у паралельній програмі (MPI_Comm_size);
6. Визначення номера поточного процесу (MPI_Comm_rank);
7. Код програми на C++ з використанням функцій MPI. В коді необхідно передбачити різну роботу для одного (або кожного) процесу.
8. Завершення паралельної частини програми (MPI_Finalize);
9. Фрагмент коду програми на C++;
10. Завершення програми.

Нище наведено типовий шаблон MPI програми, який побудований по вищенаведеному алгоритму.

```
#include "mpi.h"
int main ( int argc, char *argv[] ) {
int ProcNum, ProcRank;
// програмний код без використання MPI функцій
MPI_Init ( &agrc, &argv );
MPI_Comm_size ( MPI_COMM_WORLD, &ProcNum);
MPI_Comm_rank ( MPI_COMM_WORLD, &ProcRank);
// програмний код без використання MPI функцій
MPI_Finalize();
```

```
    // програмний код без використання MPI функцій
    return 0;
    }
```

Для реалізації процедур попарного обміну повідомленнями використовуються функції MPI_Send (для передавання) і MPI_Recv (для приймання), які мають наступний формат:

int **MPI_Send**(void *buf, int count, MPI_Datatype type, int dest,int tag, MPI_Comm comm),

де

- buf – адреса буфера пам'яті, де розміщується повідомлення;
- count – кількість елементів у повідомленні,
- type – тип елементів у повідомленні,
- dest – номер процесу отримувача,
- tag – тег повідомлення, використовується для ідентифікації повідомлення. Це число від 0 до 255.
- comm - група процесів, в рамках якої відбувається передавання.

int **MPI_Recv**(void *buf, int count, MPI_Datatype type, int source, int tag, MPI_Comm comm, MPI_Status *status)

де

- buf – адреса буфера пам'яті, куди прийматиметься повідомлення;
- count – кількість елементів у повідомленні,
- type – тип елементів у повідомленні,
- dest – номер процесу, від якого отримується повідомлення,
- tag – тег повідомлення, використовується для ідентифікації повідомлення. Значення тегу повинно співпадати при відправці і прийманні.
- comm - група процесів, в рамках якої відбувається передавання.

При заданні номера процесу і тегу при прийманні можна використовувати константи відповідно MPI_ANY_SOURCE та MPI_ANY_TAG, що означають прийняти повідомлення з від довільного процесу та з будь-яким тегом.

Приклад використання функцій:

```
MPI_Send(&ProcRank,1,MPI_INT,0,0,MPI_COMM_WORLD);
```

MPI_Recv(&RecvRank, 1, MPI_INT, MPI_ANY_SOURCE,
MPI_ANY_TAG, MPI_COMM_WORLD, &Status);

Для визначення часу виконання програми з використанням MPI використовується функція MPI_Wtime, яка повертає час поточного виконання програми. Її формат:

double **MPI_Wtime**(void).

Визначивши поточний час виконання програми на її початку, потім в кінці і віднявши ці значення отримаємо час (в секундах) виконання програми. Для підвищення точності заміру часу можна помістити тіло програми (для якої потрібно визначити час виконання) в цикл, скажімо із 10 ітерацій. Результуючий час при цьому треба розділити на 10. Таким чином можна усунути похибку визначення часу.

Для визначення точності визначення часу виконання програми, яка визначається апаратними особливостями комп'ютерної системи, використовується функція MPI_Wtick, формат якої:

double **MPI_Wtick**(void).

Функція повертає час в секундах між двома послідовними тактами апаратного таймера.

3.2 Завдання для виконання роботи

Задано одномірний масив A_i, де $0 \le i \le 1000$ відповідно до варіанту (табл.1). Необхідно написати програму, орієнтовану на 4 процеси, кожен з яких виконує:

- генерацію свого фрагмента вектора A_i;
- обробку фрагмента відповідно до завдання Z1;
- передавання результату обробки процесу з номером 0.

Нульовий процес повинен обробити отримані від інших процесів дані відповідно до завдання Z2 і вивести їх на екран.

Табл. 1.

A_i	Z1	Z2

$\frac{0.1 \cdot \log(i+1}{\cos\ (i)+2}$	Знайти різницю найменшого і найбільшого додатніх елементів	Знайти мінімальне серед отриманих від процесів значень

Програму написати з використанням операцій попарного обміну. Для програми передбачити визначення часу її виконання. Програму виконати на одному, двох, трьох і чотирьох вузлах.