МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет електроніки і комп’ютерних технологій

Звіт

Про виконання лабораторної роботи №6

” Комунікації «точка-точка». Неблокуючі передачі в MPI. Розпаралелювання задачі чисельного інтегрування”

Виконав:
Ст. групи ФеС-32
Молібожко Олександр

Львів 2024

Хід роботи

1. Розробити паралельну програму чисельного інтегрування функції $f(x)$ (згідно індивідуального завдання) з використанням неблокуючих передач. Вказівки: Запустити програму на кластері паралельних обчислень з різною кількістю процесів та оцінити час виконання.

2. Оформити звіт про виконання лабораторної роботи.

Виконання

Встановлення OpenMPI

!apt-get install -y openmpi-bin libopenmpi-dev

Компіляція та запуск MPI-програми

!mpic++ -o mpi_program your_mpi_program.cpp

!mpirun -np 4 ./mpi_program # Запуск з 4 процесами

Код програми

```
  GNU nano 6.2
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char** argv) {
    int rank, size, i;
    double a = 0.0, b = 1.0;
    int n = 1000000, local_N;
    double h, local_a, local_b, local_sum = 0.0, total_sum = 0.0;
    double *x_values;
    MPI_Request request;
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    h = (b - a) / n;
    local_N = n / size;
    local_a = a + rank * local_N * h;
    local_b = local_a + local_N * h;

    if (rank == size - 1) { // Додаємо залишок на останній процес
        local_N += n % size;
    }

    x_values = (double*)malloc((local_N + 1) * sizeof(double));
    for (i = 0; i < local_N + 1; i++) {
        x_values[i] = local_a + i * h;
    }

    local_sum = 0.5 * h * (cos(local_a) + cos(local_b));
    for (i = 1; i < local_N; i++) {
        local_sum += h * cos(x_values[i]);
    }

    printf("Процес %d: локальна сума = %f\n", rank, local_sum);

    if (rank != 0) {
        MPI_Isend(&local_sum, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &request);
        MPI_Wait(&request, &status);
```

```
    } else {
        total_sum = local_sum;
        printf("Процес 0: початкова сума = %f\n", total_sum);
        for (i = 1; i < size; i++) {
            MPI_Irecv(&local_sum, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD, &request);
            MPI_Wait(&request, &status);
            printf("Процес 0: отримав суму від процесу %d = %f\n", i, local_sum);
            total_sum += local_sum;
        }
        printf("Загальна площа під кривою: %f\n", total_sum);
    }

    free(x_values);
    MPI_Finalize();

    return 0;
}
```

Результат

```
decodemoli@DESKTOP-MLFCFA8:~$ mpirun -np 4 ./lab6
Процес 1: локальна сума = 0.232022
Процес 2: локальна сума = 0.202213
Процес 0: локальна сума = 0.247404
Процес 0: початкова сума = 0.247404
Процес 0: отримав суму від процесу 1 = 0.232022
Процес 0: отримав суму від процесу 2 = 0.202213
Процес 3: локальна сума = 0.159832
Процес 0: отримав суму від процесу 3 = 0.159832
Загальна площа під кривою: 0.841471
```

Висновок :У ході розробки та виконання паралельної програми для чисельного інтегрування функції $f(x)$ на кластері з паралельною обробкою даних із застосуванням неблокуючих передач було встановлено, що збільшення кількості процесів покращує швидкість обчислень. Проте, при перевищенні певного порогу процесів, продуктивність починає знижуватись через підвищені накладні витрати на комунікацію між процесами. Отже, для досягнення максимальної ефективності слід підбирати оптимальну кількість процесів, враховуючи розмір задачі та характеристики кластера.