Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Організація високопродуктивних обчислень

Лабораторна робота 2

Виконав

студент групи ІМ-21мн

Бурдейний В.О.

Київ – 2023

Варіант 3

Завдання: розробити паралельну програму чисельного інтегрування функції $log_2 x^3$ методом правих прямокутників за допомогою не блокуючих викликів MPI_Isend та MPI_Irecv з точністю 0,0001 у межах $[5;\ 1500]$.

Хід роботи: робота була виконана у середовищі Linux з використанням бібліотеки Open MPI, що реалізує технологію передачі повідомлень між задачами з метою розпаралелювання обчислень. Для інтегрування функції логарифму $log_2 x^3$ використано метод правих прямокутників. У такому випадку кожен вузол буде виконувати обрахунок значення інтегралу функції на власному, рівному іншим шматку інтервалу, кожен з яких надалі буде розбиватись на менші підінтервали вже безпосередньо на самих вузлах з метою досягнення бажаної точності. Загальне значення інтегралу функції на всьому проміжку буде дорівнювати сумі значень порахованих за формулою методу правих прямокутників на кінечних дрібних підінтервалах.

Програма, що реалізована на мові програмування C++, зчитує 3 значення виразу з файлу *input.txt*: початок інтервалу, кінець інтервалу, точність обчислення. Результат виконання виводиться як в стандартний потік, так і зберігається у файл *output.txt*.

Посилання на GitHub з роботою:
https://github.com/volvinbur1/OpenMPI-programming

Результат виконання: програма виконували обрахунки для інтервалу $[5; 1500]$ з точністю 0,0001 на 1-4 ядрах з використанням 1, 2 та 4 процесів. Значення часу виконання виражене в мілісекундах та вказує безпосередньо на тривалість отримання результатів при обрахунках хостом.

1. 1 ядро
    a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0 ~/local/bin/mpirun -np 1 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0441393362

[HOST - 0] Execution time: 7834[ms]
[HOST - 0] Execution time: 7834876[μs]
[HOST - 0] Execution time: 7834876257[ns]

real    0m8.406s
user    0m7.860s
sys     0m0.020s
```

b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0 ~/local/bin/mpirun -np 2 lab2
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0439879949

[HOST - 0] Execution time: 8903[ms]
[HOST - 0] Execution time: 8903148[μs]
[HOST - 0] Execution time: 8903148175[ns]

real    0m9.478s
user    0m17.827s
sys     0m0.037s
```

c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0 ~/local/bin/mpirun -np 4 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 2    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 3    Total nodes count: 4
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0436837929

[HOST - 0] Execution time: 9604[ms]
[HOST - 0] Execution time: 9604932[μs]
[HOST - 0] Execution time: 9604932417[ns]

real    0m9.981s
user    0m38.396s
sys     0m0.053s
```

2. 2 ядра

a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1 ~/local/bin/mpirun -np 1 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0441393362

[HOST - 0] Execution time: 7825[ms]
[HOST - 0] Execution time: 7825474[μs]
[HOST - 0] Execution time: 7825474589[ns]

real    0m8.408s
user    0m7.825s
sys     0m0.051s
```

b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1 ~/local/bin/mpirun -np 2 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0439879949

[HOST - 0] Execution time: 8220[ms]
[HOST - 0] Execution time: 8220248[μs]
[HOST - 0] Execution time: 8220248133[ns]

real    0m8.850s
user    0m16.456s
sys     0m0.065s
```

c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1 ~/local/bin/mpirun -np 4 lab2
[MPI_INIT_INFO] Current node rank: 3    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 2    Total nodes count: 4
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0436837929

[HOST - 0] Execution time: 4674[ms]
[HOST - 0] Execution time: 4674423[µs]
[HOST - 0] Execution time: 4674423333[ns]

real    0m5.248s
user    0m18.639s
sys     0m0.040s
```

3. 3 ядра
    a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1,2 ~/local/bin/mpirun -np 1 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0441393362

[HOST - 0] Execution time: 7831[ms]
[HOST - 0] Execution time: 7831535[µs]
[HOST - 0] Execution time: 7831535597[ns]

real    0m8.406s
user    0m7.857s
sys     0m0.026s
```

    b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1,2 ~/local/bin/mpirun -np 2 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0439879949

[HOST - 0] Execution time: 8247[ms]
[HOST - 0] Execution time: 8247874[µs]
[HOST - 0] Execution time: 8247874708[ns]

real    0m8.624s
user    0m16.525s
sys     0m0.051s
```

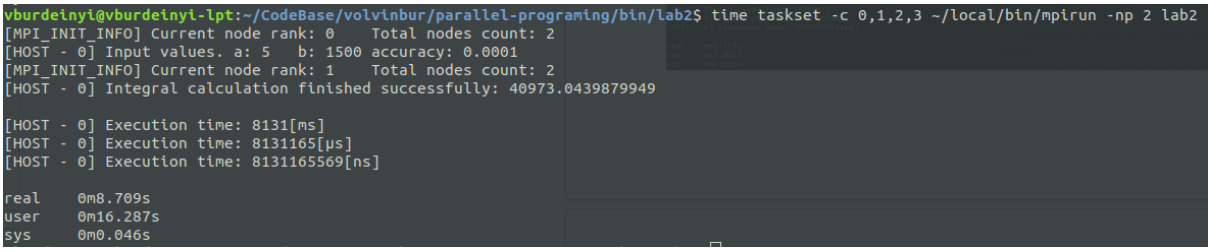
    c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1,2 ~/local/bin/mpirun -np 4 lab2
[MPI_INIT_INFO] Current node rank: 2    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 4
[MPI_INIT_INFO] Current node rank: 3    Total nodes count: 4
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0436837929

[HOST - 0] Execution time: 4547[ms]
[HOST - 0] Execution time: 4547446[µs]
[HOST - 0] Execution time: 4547446359[ns]

real    0m5.122s
user    0m18.214s
sys     0m0.053s
```

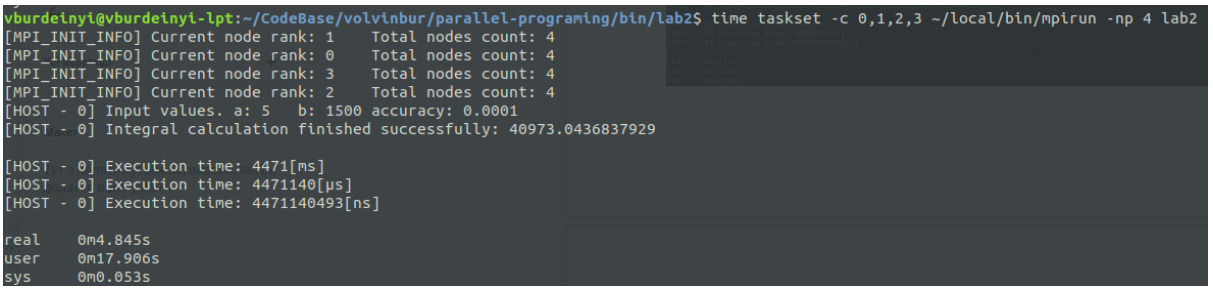
4. 4 ядра
    a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab2$ time taskset -c 0,1,2,3 ~/local/bin/mpirun -np 1 lab2
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] Input values. a: 5   b: 1500 accuracy: 0.0001
[HOST - 0] Integral calculation finished successfully: 40973.0441393362

[HOST - 0] Execution time: 7808[ms]
[HOST - 0] Execution time: 7808050[µs]
[HOST - 0] Execution time: 7808050481[ns]

real    0m8.174s
user    0m7.842s
sys     0m0.020s
```

b. 2 процеси



c. 4 процеси



Таблиця з результатами.

ms	**1 ядро**	**2 ядра**	**3 ядра**	**4 ядра**
1 процес	*7834*	*7825*	*7831*	*7808*
2 процеси	*8903*	*8220*	*8247*	*8131*
4 процеси	*9604*	*4674*	*4547*	*4471*

Висновок: у ході виконання лабораторної роботи було отримано навички зі застосування технології MPI для розпаралелювання процесу інтегрування функцій. Проаналізувавши отримані дані можна побачити, що найшвидше опрацювання було при використанні 4 ядер та процесів, а найповільніше для 1 ядра та 4 процесі. На мою думку, такий результат є досить очікуваним та логічним. Це набагато ефективніше використовувати відповідність процес-ядро, тому що процес буде виконувати менше перемикань між різними процесам для асинхронного виконання. Що стосується інших запусків для одного 1 процеса, то значення де факто однокові (незначні коливання можна рахувати за похибку). Для двох

процесів та 1 ядра ситуація схожа до 4 процесів – надлишкові витрати на очікування перемикання процесора між процесами.

Лістинг коду:

```
#include <mpi.h>
#include <fstream>
#include <iostream>
#include <sstream>
#include <cmath>
#include <iomanip>
#include <array>
#include <string>
#include <vector>
#include <chrono>

using params_array = std::array<double, 3>;

params_array read_params_from_file() {
    params_array params;
    try {
        std::ifstream file("input.txt");

        for (auto i = 0; i < 3; i++) {
            std::string line;
            std::getline(file, line);
            params.at(i) = std::stod(line);
        }

        file.close();
    } catch (std::exception& e) {
        std::cout << "[FATAL] " << e.what() << std::endl;
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    return params;
}

params_array get_parameters(int curr_rank) {
    params_array params;
    if (curr_rank == 0) {
        params = read_params_from_file();
        std::cout << "[HOST - 0] Input values. a: " << params.at(0)
            << "\tb: " << params.at(1)
```

```
            << "\taccuracy: " << params.at(2)
            << std::endl;
    }

    MPI_Bcast(params.data(), 3, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    return params;
}

bool validate_input(params_array params) {
    if (params.at(0) > params.at(1)) {
        std::cout << "[ERROR] Parameter A is greater then B. Exit..." <<
std::endl;
        return false;
    }

    if (params.at(2) < 0 || params.at(2) > 1) {
        std::cout << "[ERROR] Accuracy parameter in out of (0,1] range.
Exit..." << std::endl;
        return false;
    }

    return true;
}

bool is_accurate_enough(double new_val, double prev_val, double
accuracy) {
    return fabs(new_val - prev_val) / 3. < accuracy;
}

double integral_function(double x) {
    return log2(pow(x, 3));
}

double integrate_right_rectangle(double a_edge, double b_edge, double
accuracy) {
    int itr_cnt = 1;
    double new_res = -1, prev_res = 0;

    while (!is_accurate_enough(new_res, prev_res, accuracy)) {
        itr_cnt *= 2;
        prev_res = new_res;
        new_res = 0;
        auto h = (b_edge - a_edge) / itr_cnt;
```

```
        for (auto i = 1; i < itr_cnt; i++) {
            new_res += integral_function(a_edge + i * h) * h;
        }
    }
    return new_res;
}

double calculate_integral(int curr_rank, int nodes_cnt, params_array params) {
    auto rank_step = (params.at(1) - params.at(0)) / nodes_cnt;
    auto rank_a_edge = params.at(0) + curr_rank * rank_step;
    auto rank_b_edge = params.at(0) + (curr_rank + 1) * rank_step;
         auto rank_result = integrate_right_rectangle(rank_a_edge, rank_b_edge, params.at(2));

    auto total_integrall_value = rank_result;
    std::vector<double> nodes_results(nodes_cnt - 1);
    if (curr_rank != 0) {
        MPI_Request request = MPI_REQUEST_NULL;
                MPI_Isend(&rank_result, 1, MPI_DOUBLE, 0, curr_rank, MPI_COMM_WORLD, &request);
        MPI_Waitall(1, &request, nullptr);
    } else {
        MPI_Request requests[nodes_cnt - 1];
        for (auto i = 1; i < nodes_cnt; i++) {
             MPI_Irecv(nodes_results.data() + (i - 1), 1, MPI_DOUBLE, i, i, MPI_COMM_WORLD, &requests[i - 1]);
        }
        MPI_Waitall(nodes_cnt - 1, requests, nullptr);

        for (auto res : nodes_results) {
            total_integrall_value += res;
        }
    }

    return total_integrall_value;
}

void save_to_file(int curr_rank, double value) {
    if (curr_rank != 0) {
        return;
    }
```

```
    try {
        std::ofstream file("output.txt");
            file << std::fixed << std::setprecision(10) << value <<
std::endl;
        file.close();
    } catch (std::exception& e) {
        std::cout << "[FATAL] " << e.what() << std::endl;
        MPI_Abort(MPI_COMM_WORLD, 1);
    }
}

int main(int argc, char *argv[]) {
    MPI_Init(&argc, &argv);

    int curr_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &curr_rank);

    int nodes_cnt;
    MPI_Comm_size(MPI_COMM_WORLD, &nodes_cnt);

     std::cout << "[MPI_INIT_INFO] Current node rank: " << curr_rank <<
"\tTotal nodes count: " << nodes_cnt << std::endl;

    auto params = get_parameters(curr_rank);
    if (!validate_input(params)) {
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    auto start_time = std::chrono::steady_clock::now();
    auto result = calculate_integral(curr_rank, nodes_cnt, params);
    auto finish_time = std::chrono::steady_clock::now();

    save_to_file(curr_rank, result);

    MPI_Finalize();

    if (curr_rank == 0) {
            std::cout << "[HOST - 0] Integral calculation finished
successfully: " << std::fixed << std::setprecision(10) << result <<
std::endl;

        std::cout << std::endl;
```

```
                std::cout << "[HOST - 0] Execution time: " <<
std::chrono::duration_cast<std::chrono::milliseconds>(finish_time -
start_time).count() << "[ms]" << std::endl;
                std::cout << "[HOST - 0] Execution time: " <<
std::chrono::duration_cast<std::chrono::microseconds>(finish_time -
start_time).count() << "[µs]" << std::endl;
                std::cout << "[HOST - 0] Execution time: " <<
std::chrono::duration_cast<std::chrono::nanoseconds>(finish_time -
start_time).count() << "[ns]" << std::endl;
    }

    return 0;
}
```