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 1

Виконав

студент групи ІМ-21мн

Бурдейний В.О.

Київ – 2023

Варіант 3

Завдання: виконати розпаралелювання задачі пошуку значення функції квадратного кореня $\sqrt{1+x}$ у певній точці методом обрахування суми ряду з точністю 1e-8.

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 обрахунку значення функції квадратного кореня $\sqrt{1+x}$ використано ряд Тейлора у загальному вигляді, коли кожен доданок в сумі обчислюється окремою задачею MPI (включаючи й хостову). У такому підході правильність знаходження значення функції може гарантуватись при $|x| < 1$. Безпосередньо сама точність обчислень досягається шляхом порівняння значення добутку з мінімальним обмеженням. Якщо перше меньше другого, то виконання можно завершувати.

Програма, що реалізована на мові програмування C++, зчитує значення підкореневого виразу з файлу *input.txt*, від якого далі віднімається одиниця для отримання безпосереднього значення параметру x. 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: для прикладу було виконано обчислення квадратного кореня для числа 1.1462963582 на 1-4 ядрах з використанням 1, 2 та 4 процесів. Значення часу виконання виражене в мікросекундах та вказує безпосередньо на тривалість отримання результатів при обрахунках хостом.

1. 1 ядро
    a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0 ~/local/bin/mpirun -np 1 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 11[µs]
[HOST - 0] Execution time: 11656[ns]

real    0m0.377s
user    0m0.033s
sys     0m0.027s
```

b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0 ~/local/bin/mpirun -np 2 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 24[µs]
[HOST - 0] Execution time: 24115[ns]

real    0m0.375s
user    0m0.039s
sys     0m0.030s
```

c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0 ~/local/bin/mpirun -np 4 lab1
[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] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[MPI_INIT_INFO] Current node rank: 3    Total nodes count: 4
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 76[µs]
[HOST - 0] Execution time: 76478[ns]

real    0m0.378s
user    0m0.048s
sys     0m0.058s
```

Таблиця з результатами

	1 ядро	2 ядра	3 ядра	4 ядра

2. 2 ядра

a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1 ~/local/bin/mpirun -np 1 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 19[µs]
[HOST - 0] Execution time: 19523[ns]

real    0m0.576s
user    0m0.016s
sys     0m0.037s
```

b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1 ~/local/bin/mpirun -np 2 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 22[µs]
[HOST - 0] Execution time: 22113[ns]

real    0m0.373s
user    0m0.043s
sys     0m0.031s
```

c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1 ~/local/bin/mpirun -np 4 lab1
[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] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 55[µs]
[HOST - 0] Execution time: 55109[ns]

real    0m0.377s
user    0m0.032s
sys     0m0.080s
```

3. 3 ядра

a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1,2 ~/local/bin/mpirun -np 1 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 15[µs]
[HOST - 0] Execution time: 15567[ns]

real    0m0.591s
user    0m0.000s
sys     0m0.056s
```

b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1,2 ~/local/bin/mpirun -np 2 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 30[µs]
[HOST - 0] Execution time: 30157[ns]

real    0m0.380s
user    0m0.022s
sys     0m0.053s
```

c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1,2 ~/local/bin/mpirun -np 4 lab1
[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] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 57[µs]
[HOST - 0] Execution time: 57052[ns]

real    0m0.373s
user    0m0.042s
sys     0m0.067s
```

4. 4 ядра

   a. 1 процес

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1,2,3 ~/local/bin/mpirun -np 1 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 1
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 15[µs]
[HOST - 0] Execution time: 15192[ns]

real    0m0.376s
user    0m0.021s
sys     0m0.040s
```

   b. 2 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1,2,3 ~/local/bin/mpirun -np 2 lab1
[MPI_INIT_INFO] Current node rank: 0    Total nodes count: 2
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[HOST - 0] square root calculation started.
[MPI_INIT_INFO] Current node rank: 1    Total nodes count: 2
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 25[µs]
[HOST - 0] Execution time: 25127[ns]

real    0m0.373s
user    0m0.027s
sys     0m0.041s
```

   c. 4 процеси

```
vburdeinyi@vburdeinyi-lpt:~/CodeBase/volvinbur/parallel-programing/bin/lab1$ time taskset -c 0,1,2,3 ~/local/bin/mpirun -np 4 lab1
[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: 3    Total nodes count: 4
[HOST - 0] square root param value: 1.1462963582
[HOST - 0] accuracy: 1e-08
[MPI_INIT_INFO] Current node rank: 2    Total nodes count: 4
[HOST - 0] square root calculation started.
[HOST - 0] break
[HOST - 0] Calculation finished. Square root of '1.1462963582' is equal 1.0706523047

[HOST - 0] Execution time: 53[µs]
[HOST - 0] Execution time: 53845[ns]

real    0m0.364s
user    0m0.059s
sys     0m0.023s
```

Таблиця з результатами.

µs	**1 ядро**	**2 ядра**	**3 ядра**	**4 ядра**
1 процес	*11*	*19*	*15*	*14*
2 процеси	*24*	*22*	*30*	*25*

4 процеси	*76*	*55*	*57*	*53*

Висновок: у ході виконання лабораторної роботи було отримано навички зі застосування технології MPI для розпаралелювання обчислень математичних завдань. Проаналізувавши отримані дані можна побачити, що найшвидше опрацювання було при використанні одного ядра та процесу. На мою думку, це спричинено саме не великою обчислювальною складністю. Також при запуску 1 процесу немає надлишкових витрат часу на комунікацію між ними, що, за таких умов, сприяє пришвидшенню роботи.

Що стосується інших запусків на 2, 3, 4 ядрах, то можна стверджувати, що отримані результати по часу фактично ідентичні один одному. Розбіжності між ними є похибкою.

Лістинг коду:

```
#include <mpi.h>
#include <fstream>
#include <iostream>
#include <sstream>
#include <limits>
#include <cmath>
#include <iomanip>
#include <chrono>

const double accuracy = 1e-8;
const int x_value_tag = 1;
const int seq_number_tag = 2;
const int series_element_value_tag = 3;
const int break_flag_tag = 4;

double read_x_from_file() {
	try {
		std::ifstream file("input.txt");
		std::stringstream input_str_buffer;
		input_str_buffer << file.rdbuf();

		return std::stod(input_str_buffer.str());
```

```
	} catch (std::exception& e) {
		std::cout << "[FATAL] " << e.what() << std::endl;
		MPI_Abort(MPI_COMM_WORLD, 1);
	}
	return 0;
}

double get_x_value(int curr_rank, int nodes_cnt) {
	double input_x_value;
	if (curr_rank == 0) {
		input_x_value = read_x_from_file();
		  std::cout << "[HOST - 0] square root param value: " <<
std::fixed << std::setprecision(10) << input_x_value << std::endl;
			std::cout << "[HOST - 0] accuracy: " <<
std::defaultfloat << accuracy << std::endl;

		input_x_value -= 1;

			for (auto dest_rank = 1; dest_rank < nodes_cnt;
dest_rank++) {
				MPI_Send(&input_x_value, 1, MPI_DOUBLE,
dest_rank, x_value_tag, MPI_COMM_WORLD);
		}
	} else {
		 MPI_Recv(&input_x_value, 1, MPI_DOUBLE, 0, x_value_tag,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
	}
	return input_x_value;
}

bool validate_input(double x_value) {
	if (fabs(x_value) > 1) {
		 std::cout << "[ERROR] Input param value -1 is out of the
range [-1, 1]. Exit..." << std::endl;
		return false;
	}

	return true;
}

double factorial(int value) {
	if (value < 0) {
		return 0;
```

```
	} else if (value == 0) {
		return 1;
	}
	return value * factorial(value - 1);
}

double calc_series_element(double x, int seq_number) {
		auto numerator = pow(-1, seq_number - 1) * factorial(2*seq_number);
		auto denominator = pow(4, seq_number) * pow(factorial(seq_number), 2) * (2 * seq_number - 1);
	return numerator / denominator * pow(x, seq_number);
}

int get_target_seq_number(int curr_rank, int nodes_cnt, int* last_seq_number) {
	int target_seq_number;
	if (curr_rank == 0) {
		target_seq_number = (*last_seq_number)++;
			for (auto dest_rank = 1; dest_rank < nodes_cnt; dest_rank++) {
				MPI_Send(last_seq_number, 1, MPI_INT, dest_rank, seq_number_tag, MPI_COMM_WORLD);
				(*last_seq_number)++;
		}
	} else {
			MPI_Recv(&target_seq_number, 1, MPI_INT, 0, seq_number_tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
	}
	return target_seq_number;
}

bool get_break_flag(int curr_rank, int nodes_cnt, double* series_elements_sum, double series_element_value) {
	auto break_calculation = false;
	if (curr_rank == 0) {
		(*series_elements_sum) += series_element_value;
		if (fabs(series_element_value) < accuracy) {
			break_calculation = true;
			std::cout << "[HOST - 0] break" << std::endl;
		} else {
			for (auto dest_rank = 1; dest_rank < nodes_cnt; dest_rank++) {
```

```
                                    MPI_Recv(&series_element_value, 1,
MPI_DOUBLE,    dest_rank,    series_element_value_tag,    MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
                                                (*series_elements_sum)  +=
series_element_value;
                                    if (fabs(series_element_value) <
accuracy) {
                                    break_calculation = true;
                                          std::cout << "[NODE - " <<
dest_rank << "] break" << std::endl;
                                    break;
                                }
                            }
                        }

                    for (auto dest_rank = 1; dest_rank < nodes_cnt;
dest_rank++) {
                            MPI_Send(&break_calculation, 1, MPI_BYTE,
dest_rank, break_flag_tag, MPI_COMM_WORLD);
                }
        } else {
                    MPI_Send(&series_element_value, 1, MPI_DOUBLE, 0,
series_element_value_tag, MPI_COMM_WORLD);
                        MPI_Recv(&break_calculation, 1, MPI_BYTE, 0,
break_flag_tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        }

        return break_calculation;
}

double calculate_square_root(int curr_rank, int nodes_cnt, double x) {
        if (curr_rank == 0) {
                    std::cout << "[HOST - 0] square root calculation
started." << std::endl;
        }

        auto last_seq_number = 0;
        auto series_elements_sum = 0.0;

        while (true) {
                                        auto   target_seq_number   =
get_target_seq_number(curr_rank, nodes_cnt, &last_seq_number);
```

```
                    auto series_element_value = calc_series_element(x,
target_seq_number);

                        // std::cout << "[CalcNode - "<< curr_rank <<
"][SeqNumber - " << target_seq_number
                                        //                        <<   "]
calculate_square_root::series_element_value: " << series_element_value
<< std::endl;

                                if  (get_break_flag(curr_rank,  nodes_cnt,
&series_elements_sum, series_element_value)) {
                            break;
                }
        }

        return series_elements_sum;
}

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 sqrt_param = get_x_value(curr_rank, nodes_cnt);
        if (!validate_input(sqrt_param)) {
                MPI_Abort(MPI_COMM_WORLD, 1);
        }

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

        save_to_file(curr_rank, result);

        MPI_Finalize();

        if (curr_rank == 0) {
                  std::cout << "[HOST - 0] Calculation finished. Square
root of '" <<
                              std::fixed << std::setprecision(10) << 1 +
sqrt_param << "' is equal " << result << std::endl;

                std::cout << 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;
}
```