Филиал федерального государственного бюджетного образовательного учреждения высшего образования «Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

Направление: 09.04.01. «Информатика и вычислительная техника»

Профиль: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Лабораторная работа №3

«**Вычисление суммы членов ряда с помощью MPI-программы**»

по курсу:

«Вычислительные системы»

Студент: Старостенков А.А.

Группа: ВМ-22(маг)

Вариант: 19

Преподаватель: Федулов А.С.

Смоленск, 2023

Задание

1. Написать на языке Си, скомпилировать, отладить и запустить на гибридном вычислительном кластере (ГВК) СФМЭИ **последовательную** программу вычисления суммы числового ряда: $\sum_{n=1}^{N} a_n$, где a_n - общий член ряда. Вариант задания (общий член ряда) выбрать в таблице (по номеру журнала). Предусмотреть вывод результата. В теле цикла выделить вычисление общего члена ряда и накапливающее суммирование членов ряда.
2. Выполнить проверку правильности вычисления суммы членов ряда, например, с помощью математических пакетов.
3. Предусмотреть замер времени вычисления суммы членов ряда. Число членов ряда N выбрать таким, чтобы время вычисления в последовательной программе было порядка 2- 5 сек.
4. На основе последовательной программы отладить **параллельную** MPI-программу вычисления суммы числового ряда. Использовать то же значение числа членов ряда N, что и в последовательной программе. Предусмотреть замер времени.

   Вычисление отдельных членов ряда в цикле необходимо **самостоятельно** распределить по процессам, по возможности равномерно. Необходимо учитывать то, что в отличие от OpenMP, в MPI **нет средств автоматического распределения** итераций цикла по процессам. Для решения этой задачи можно использовать подход, представленный на слайдах 46- 47 в лекции 4 по курсу «Вычислительные системы».

   Для суммирования частичных сумм, вычисленных всеми процессами, после завершения цикла необходимо использовать функцию **MPI_Reduce**.
5. Запустить отлаженную параллельную программу на одном, двух и трех узлах с максимальным числом процессов.
6. Сравнить результаты и время вычисления последовательной и параллельной программ.
7. Все выполненные задания оформить в виде отчета.

$$\frac{3}{10n^2 - 2n - 3}$$

Ход работы

1. Написать на языке Си, скомпилировать, отладить и запустить на гибридном вычислительном кластере (ГВК) СФМЭИ последовательную программу вычисления суммы числового ряда: $\sum_{n=1}^{N} a_n$, где a_n - общий член ряда. Вариант задания (общий член ряда) выбрать в таблице (по номеру журнала). Предусмотреть вывод результата. В теле цикла выделить вычисление общего члена ряда и накапливающее суммирование членов ряда.

2. Выполнить проверку правильности вычисления суммы членов ряда, например, с помощью математических пакетов.

3. Предусмотреть замер времени вычисления суммы членов ряда. Число членов ряда N выбрать таким, чтобы время вычисления в последовательной программе было порядка 2- 5 сек.

Код:

```c
#include <stdio.h>
#include <omp.h>
#include <unistd.h>
int main()
{
    double start_time, end_time, tick, summa;
    start_time = omp_get_wtime();
    sleep(1);
    int i;
    double s;
    for (i = 0; i <= 550000000; i++)
    {
        s = 3.0 / ((10 * i * i) - (2 * i) - 3);
        summa += s;
    }
    end_time = omp_get_wtime();
    tick = omp_get_wtick();
    printf("Time range %E\n", end_time - start_time);
    printf("Accuracy of timer %E\n", tick);
    printf("Sum = %E\n", summa);
    return 0;
}
```

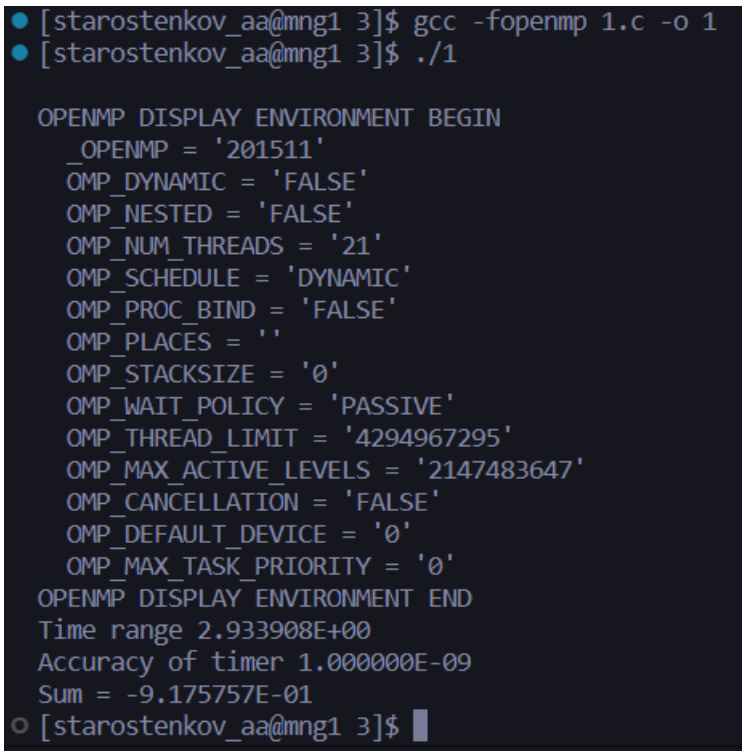

Рисунок 1 – Результат последовательной программы

4. На основе последовательной программы отладить параллельную MPI-программу вычисления суммы числового ряда. Использовать то же значение числа членов ряда N, что и в последовательной программе. Предусмотреть замер времени.

Код:

```c
#include <stdio.h>
#include <mpi.h>

int main()
{
    int num_procs, rank;
    double start_time, end_time, tick, summa = 0.0;
    int N = 550000000;

    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    char name[MPI_MAX_PROCESSOR_NAME];
    int len;
    MPI_Get_processor_name(name, &len);

    printf("Максимальное число процессов: %d\n", num_procs);
    printf("Узел: %s\n", name);
```

```
    int local_N = N / num_procs;      // Число итераций для каждого
процесса
    int start_index = rank * local_N; // Начальный индекс для каждого
процесса

    double local_sum = 0.0;
    double s;
    int i;

    start_time = MPI_Wtime();

    // Вычисление частичной суммы для каждого процесса
    for (i = start_index; i < start_index + local_N; i++)
    {
        s = 3.0 / ((10 * i * i) - (2 * i) - 3);
        local_sum += s;
    }

    // Сбор частичных сумм со всех процессов
    MPI_Reduce(&local_sum, &summa, 1, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);

    end_time = MPI_Wtime();
    tick = MPI_Wtick();
    printf("Time range %E\n", end_time - start_time);
    printf("Accuracy of timer %E\n", tick);

    if (rank == 0)
    {
        printf("Sum = %E\n", summa);
    }
    MPI_Finalize();

    return 0;
}
```

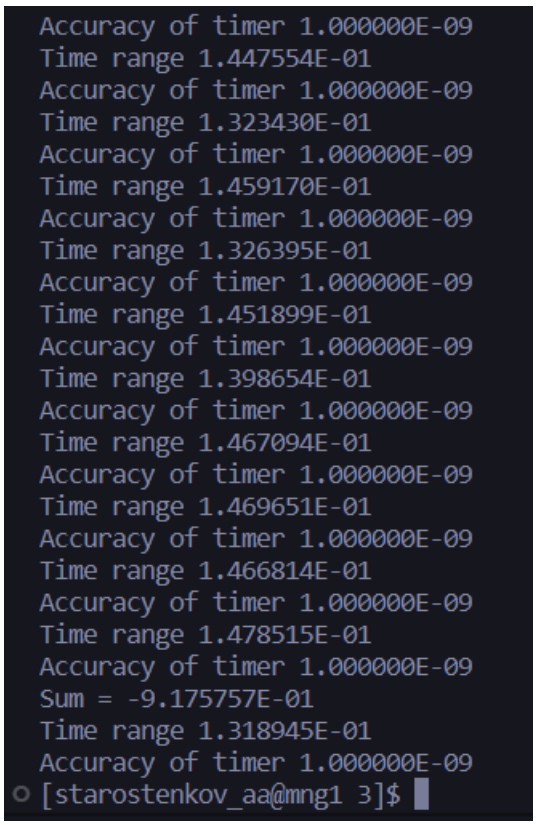

Рисунок 2 - Результат работы программы параллельной вычисления суммы числового ряда

Критерии	Последовательная программа	Параллельная программа
Временной диапазон	2.930924E+00	1.486458E-01
Точность таймера	1.000000E-09	1.000000E-09
Результат	-9.175757E-01	-9.175757E-01

5. Запустить отлаженную параллельную программу на одном, двух и трех узлах с максимальным числом процессов.

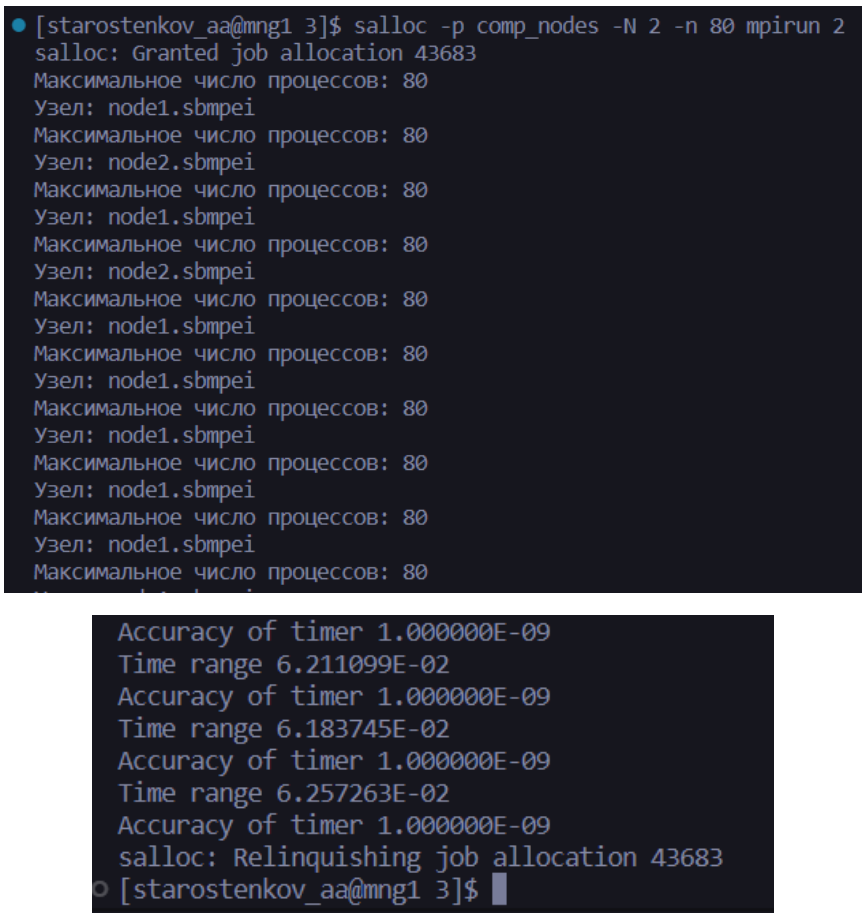

Рисунок 3 – Запуск на двух узлах

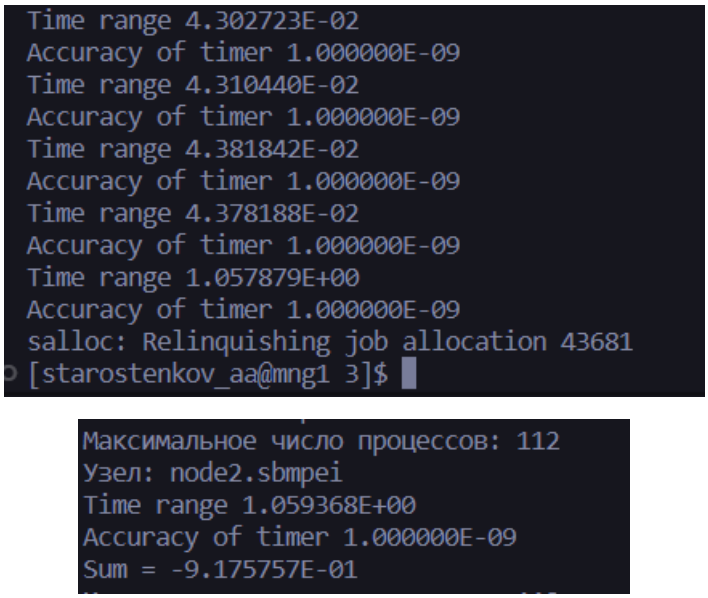

Рисунок 4 – запуск на 3х узлах

Программа запускается на двух и трех узлах. Параллельная программа считается быстрее чем на одном узле. Результат верный.

6. Сравнить результаты и время вычисления последовательной и параллельной программ.

Вывод: Мы видим, что параллельная программа показывает значительное улучшение во времени выполнения по сравнению с последовательной программой. Временной диапазон параллельной программы составляет всего 1.486458E-01, что гораздо меньше (в ~20 раз), чем время выполнения последовательной программы, которое составляет 2.930924E+00. Точность таймера для обеих программ одинаковая и составляет 1.000000E-09. Оба варианта программы дают одинаковый результат, который равен -9.175757E-01.

При вычислении на двух и трех узлах время значительно сократилось, но между собой оно сопоставимо.

Таким образом, параллельная программа демонстрирует лучшую производительность и более быстрое время выполнения по сравнению с последовательной программой.