**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: , где  - 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**.

1. Запустить отлаженную параллельную программу на одном, двух и трех узлах с максимальным числом процессов.
2. Сравнить результаты и время вычисления последовательной и параллельной программ.
3. Все выполненные задания оформить в виде отчета.



**Ход работы**

1. Написать на языке Си, скомпилировать, отладить и запустить на гибридном вычислительном кластере (ГВК) СФМЭИ последовательную программу вычисления суммы числового ряда:  , где  - 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:

#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;

}

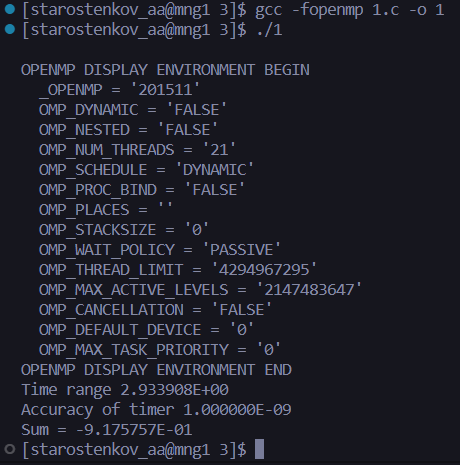


Рисунок 1 – Результат последовательной программы

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:

#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;

}

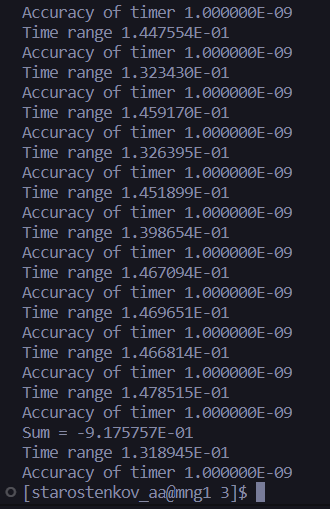
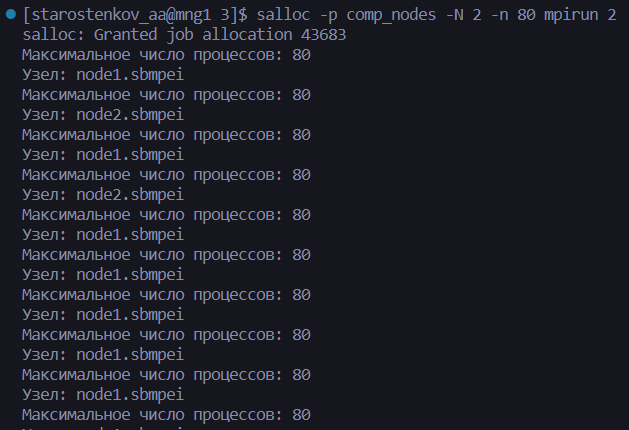


Рисунок 2 - Результат работы программы параллельной вычисления суммы числового ряда

|  |  |  |
| --- | --- | --- |
| 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.



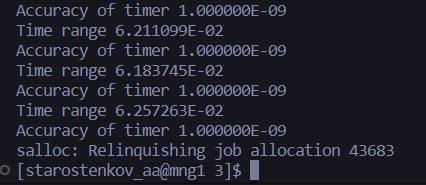
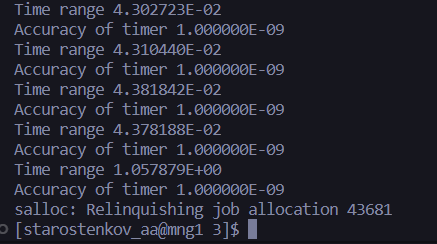


Рисунок 3 – Запуск на двух узлах



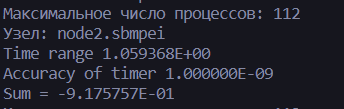


Рисунок 4 – запуск на 3х узлах

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.