|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 2 | | |
| по дисциплине «Введение в архитектуру компьютера» | | |
| **Совместный доступ нескольких потоков к данным в общей памяти** | | |
|  | | |
|  | Бригада 3 | пм-23 Нос алексей |
| ПМ-23-24-25 | пм-24 параскун иван |
| Вариант 4 | пм-24 царева виктория |
|  | пм-25 таджибаев завкиддин |
|  | пм-23 лужанский александр |
|  | пм-24 кузнецова алиса |
| Преподаватели | Тракимус Юрий Викторович |
|  |  |
| Новосибирск, 2023 | | |

1. **Цель работы**

1. Написание многопоточной̆ программы вычисления с использованием OpenMP.

2. Изучение способов измерения времени работы программы.

3. Изучение приёмов повышения скорости работы программы.

4. Построение графика зависимости времени работы программы вычисления от количества членов ряда для лучшей̆ реализации и многопоточной̆ реализации.

1. **Вариант задания**

Алгоритм вычисления функции с помощью разложения в ряд по первым членам этого ряда:

Область сходимости ряда:

1. **Текст программы с алгоритмической оптимизацией**

#include <math.h>

#include <stdio.h>

#include <time.h>

double \_sin(unsigned N, double x);

int main()

{

double x, sin\_x;

unsigned N;

scanf("%u %lf", &N, &x);

clock\_t stime = clock();

sin\_x = \_sin(N, x);

clock\_t etime = clock();

printf("sin(x) = %.31lf\nTime = %.2lf\n", sin\_x,

(double)(etime - stime) / CLOCKS\_PER\_SEC \* 1000);

return 0;

}

int cast(double \*x)

{

int sgn = x < 0 ? -1 : 1;

\*x = fmod(fabs(\*x), M\_PI \* 2);

if (\*x > M\_PI)

{

sgn \*= -1;

\*x -= M\_PI;

}

\*x = \*x > M\_PI\_2 ? M\_PI - \*x : \*x;

return sgn;

}

inline double \_sin(unsigned N, double x)

{

int sgn = cast(&x);

double sin = x, ds = x;

for (unsigned n = 2; n <= N; n++)

{

ds \*= -x \* x;

ds /= 2 \* (n - 1) \* (2 \* n - 1);

sin += ds;

}

return sin \* sgn;

}

1. **Текст программы с использованием OpenMP**

#include <math.h>

#include <omp.h>

#include <stdio.h>

#include <time.h>

double \_sin(double, unsigned);

int main()

{

double x, sin\_x, start, end;

unsigned N;

scanf("%u %lf", &N, &x);

start = omp\_get\_wtime();

sin\_x = \_sin(x, N);

end = omp\_get\_wtime();

printf("sin(x) = %.31lf\nTime = %.2lf\n", sin\_x, (end - start) \* 1000);

return 0;

}

char cast(double \*);

double fact(unsigned);

double

\_sin(double x, unsigned N)

{

double ds, part, sin\_x;

char f, sgn = cast(&x);

#pragma omp parallel private(ds, part, f) shared(sin\_x)

{

ds = part = sin\_x = 0;

f = 0;

#pragma omp for

for (unsigned long long i = 1; i <= N; i++)

{

if (f)

{

ds \*= -x \* x;

ds /= (i \* (i + 1));

}

else

{

ds = pow(x, i \* 2 - 1) / fact(i \* 2 - 1) \* (i % 2 == 0 ? -1 : 1);

f = 1;

}

part += ds;

}

#pragma omp critical

{

sin\_x += part;

}

}

return sin\_x \* sgn;

}

char cast(double \*x)

{

char sgn = x < 0 ? -1 : 1;

\*x = fmod(fabs(\*x), M\_PI \* 2);

if (\*x > M\_PI)

{

sgn \*= -1;

\*x -= M\_PI;

}

\*x = \*x > M\_PI\_2 ? M\_PI - \*x : \*x;

return sgn;

}

double

fact(unsigned x)

{

double r = 1;

for (unsigned i = 2; i <= x; i++)

r \*= i;

return r;

}

1. **Точность вычисления числа**

|  |  |  |
| --- | --- | --- |
| **N** |  | |
| **Без OMP** | **OMP (4 потока)** |
| 4 | **0,8414**682539682539763603585925011 | **0,8414**682539682539763603585925011 |
| 8 | **0,84147098480789**37293180956658034 | **0,841**2526902776176207510161475511 |
| 16 | **0,84147098480789650**48756572286948 | **0,84**65304690524199093815127525886 |
| 32 | **0,84147098480789650**48756572286948 | **0,84**65503844847508396753710258054 |
| 106 | **0,84147098480789650**48756572286948 | **0,84**65503844862531934722937876359 |
| 107 | **0,84147098480789650**48756572286948 | **0,84**65503844862531934722937876359 |
| Калькулятор  Windows | **0,84147098480789650**66525023216303 | |

1. **График зависимости времени работы программы от количества членов ряда**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **Время работы c алгоритмической оптимизацией с ключом 02, мс** | | | | | | |
| **Без OMP** | **С OMP** | | | | | |
| **2 потока** | % | **4 потока** | % | **8 потоков** | % |
| 108 | 354,11 | 276,06 | 78 | 322,51 | 91 | 287,84 | 81 |
| 2\*108 | 668,40 | 575,03 | 86 | 579,45 | 87 | 614,1 | 92 |
| 3\*108 | 1021,75 | 843,32 | 83 | 877,56 | 86 | 1108,86 | 109 |
| 4\*108 | 1313,91 | 1110,24 | 84 | 1141,97 | 87 | 1172,72 | 89 |
| 5\*108 | 1678,73 | 1360,39 | 81 | 1390,36 | 83 | 1615,26 | 96 |
| 6\*108 | 1985,83 | 1659,82 | 84 | 1856,18 | 93 | 2026,43 | 102 |
| 7\*108 | 2317,78 | 1954,63 | 84 | 2095,23 | 90 | 2133 | 92 |
| 8\*108 | 2676,30 | 2309,38 | 86 | 2460,22 | 92 | 2499,26 | 93 |
| 9\*108 | 3061,52 | 2577,51 | 84 | 2571,91 | 84 | 2946,8 | 96 |

1. **Вывод**

В ходе исследования было выявлено:

1. При использовании OpenMP на домашнем ПК время выполнения программы уменьшается на 15-20% при распараллеливании на 2 потока, 10-15% при распараллеливании на 4 потока, 5-10% при распараллеливании на 8 потоков.

2. Точность вычисления с разделением на потоки при количестве членов больше 6 уменьшается, остается неизменной при N > 16 .

3. Точность в программе с использованием OpenMP снижается из-за использования функций pow() и fact(), также выигрыш по времени по сравнению с программой без использования OpenMP будет выше без использования данных функций.

4. Наиболее точный и при этом быстрый результат на данной архитектуры можно получить при использовании OpenMP с распараллеливанием на 2 потока, запуская программу с ключом 02.

1. **Архитектура компьютера**

|  |  |
| --- | --- |
| **Характеристика** | **Домашний ПК** |
| **Компилятор** | g++ (Debian 12.2.0-14) 12.2.0 |
| **Операционная система** | Debian GNU/Linux 12 (Linux 6.1.0-13-amd64) |
| **Процессор** | AMD Ryzen 5 3500U, 2100 MHz, x86\_64 |
| **Число ядер** | 4 |
| **Число логический процессоров** | 8 |
| **Объем кэша** | L1d: 128 KiB (4 inst) L1i: 256 KiB (4 inst) L2: 2 MiB (4 inst) L3: 4 MiB (1 inst) |
| **Установленная ОЗУ** | 8 GB, SODIMM-DDR4, 2400 MT/s |
| **stress-ng - bogo ops/s (double, 180s)** | 5542.89 |
| **stress-ng - bogo ops/s (matrixprod, 180s)** | 2736.06 |