|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 3 | | |
| по дисциплине «Введение в архитектуру компьютера» | | |
| **Исследование эффективности работы многопоточной  программы сортировки Шелла** | | |
|  | | |
|  | Бригада 3 | пм-23 Нос алексей |
| ПМ-23-24-25 | пм-24 параскун иван |
| Вариант 1 | пм-24 царева виктория |
|  | пм-25 таджибаев завкиддин |
|  | пм-23 лужанский александр |
|  | пм-24 кузнецова алиса |
| Преподаватель | Тракимус Юрий Викторович |
|  |  |
| Новосибирск, 2023 | | |

1. **Цель работы**

Написать на Си многопоточную программу алгоритма сортировки Шелла с помощью OpenMP.

1. **Условие задачи**

Реализовать сортировку алгоритмом Шелла.

1. **Текст программы**

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <locale.h>

#include <omp.h>

int shell\_sort(int\* array, int size, int threads)

{

int i = 0;

omp\_set\_num\_threads(threads);

for (int s = size / 2; s > 0; s /= 2)

#pragma omp parallel for reduction(+: i) //Если убрать директиву reduction(+: i), то упадёт скорость сортировки

for (int i = s; i < size; i++)

#pragma omp critical //Если убрать директиву critical, то сортировка перестанет работать корректно

for (int j = i - s; j >= 0 && array[j] > array[j + s]; j -= s)

{

int temp = array[j];

array[j] = array[j + s];

array[j + s] = temp;

}

return 0;

}

int main()

{

setlocale(0, "");

time\_t start, end;

long long int n;

int thread;

FILE\* file;

printf("Введите количество элементов массива: ");

scanf\_s("%lld", &n);

printf("Введите количество потоков ");

scanf\_s("%d", &thread);

int\* B = new int[n];

srand(1000); //Seed генерации

fopen\_s(&file, "base.txt", "w");

for (int i = 0; i < n; i++)

{

B[i] = rand() % 201;

fprintf(file, "%d ", B[i]);

}

fclose(file);

fopen\_s(&file, "sort.txt", "w");

start = clock();

shell\_sort(B, n, thread);

end = clock();

for (int i = 0; i < n; i++)

fprintf(file, "%d ", B[i]);

fclose(file);

printf("%f <- времени прошло\n", (double)(end - start) / CLOCKS\_PER\_SEC);

delete[]B;

}

1. **Характеристики**

|  |  |  |
| --- | --- | --- |
|  | Домашний компьютер №1 | Домашний компьютер №2 |
| Компилятор | Visual Studio Community 2022 Версия 17.6.10 | Visual Studio Community 2022 Версия 17.6.2 |
| OC | Microsoft Windows 10 Pro x64  Сборка 18363.476 | Microsoft Windows 11 x64  22H2  22621.2283 |
| Процессор | Intel(R) Core(TM) i5-6300U  CPU @ 2.50 GHz  2 ядра | AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz  8 ядер |
| Кэш | L1 - 128 КБ  L2 - 512 КБ  L3 - 3 МБ | L1- 512Кб  L2- 4Мб  L3- 16Мб |
| ОЗУ | 8 ГБ | 16 ГБ |
| Тип системы | 64-разрядная операционная система | 64-разрядная операционная система |

1. **Термины**

* OpenMP — открытый стандарт для распараллеливания программ на языках C, C++ и Фортран. Даёт описание совокупности директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью.
* Директива parallel — определяет параллельный регион, который является кодом, который будет выполняться несколькими потоками параллельно.
* Директива for — вызывает разделение работы в for цикле внутри параллельного региона между потоками.
* Директива reduction - указывает, что одна или несколько переменных, которые являются частными для каждого потока, подлежат операции сокращения в конце параллельной области.
* Директива critical - указывает, что код выполняется только в одном потоке за раз.

1. **Графики зависимости времени работы программы от количества элементов в массиве**

Домашний ПК №1:

|  |  |  |  |
| --- | --- | --- | --- |
| Количество членов ряда | Время работы программы с ключом /O2, OpenMP, c | | |
| 1 поток | 2 потока | 4 потока |
| 10^5 | 0,023 | 0,022 | 0,024 |
| 10^6 | 0,272 | 0,286 | 0,303 |
| 10^7 | 3,924 | 3,335 | 3,525 |
| 10^8 | 38,916 | 41,235 | 41,906 |

Домашний ПК №2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество членов ряда | Время работы программы с ключом /O2, OpenMP, c | | | | |
| 1 поток | 2 потока | 4 потока | 8 потоков | 16 потоков |
| 10^5 | 0,042 | 0,041 | 0,041 | 0,041 | 0,041 |
| 10^6 | 0,692 | 0,523 | 0,510 | 0,138 | 0,138 |
| 10^7 | 6,895 | 6,480 | 6,293 | 6,106 | 5,733 |
| 10^8 | 75,494 | 75,105 | 74,911 | 74,523 | 73,747 |

Общий график с лучшими показателями

График со сравнимыми показателями

1. **Выводы**

На Домашнем ПК №2:

1. У ПК №2 с 2, 4, 8, 16 потоками увеличилась производительность в среднем на 1.17%, 1.64%, 2.78%, 4.16% соответственно, по сравнению с 1 потоком.
2. При каждом увеличении потоков в 2 раза уменьшался прирост скорости (+0.47%, +1.14%, +1.38%).

На домашнем ПК №1:

1. У ПК №1 с 2 потоками увеличилась производительность в среднем на 3.88% по сравнению с 1 потоком.
2. У ПК №1 с 4 потоками увеличилась производительность в среднем на 6.08% по сравнению с 1 потоком.

Общие:

1. ПК №1 с 1 потоком производительнее терминала с 1 потоком в среднем на 48%.
2. ПК №1 с 2 потоками производительнее терминала с 2 потоками в среднем на 45%.
3. ПК №1 с 4 потоками производительнее терминал с 4 потоками в среднем на 44%.
4. ПК №2 с 16 потоками производительнее ПК с 4 потоками в среднем на 42%.

Лучший вариант по результатам тестов — это ПК №1 с 1 потоком.