**ФГАОУ ВО «Национальный исследовательский университет ИТМО»**

**Факультет программной инженерии и компьютерной техники**

**Параллельные вычисления**

**Лабораторная работа №1**

**Автоматическое распараллеливание программ**

**Студент: Александров Ю.В. гр. Р4114**

**Преподаватель: Жданов А. Д.**

**Санкт-Петербург**

**2023**

Александров Юрий Витальевич

11 4 10

1. **Этап Generate**

srand(i);

double\* M1 = (double\*)malloc(N \* sizeof(double));

double\* M2 = (double\*)malloc(N / 2 \* sizeof(double));

// 1 Этап

// Заполение массива M1

for (int k = 0; k < N; k++) {

double value = 1 + rand\_r(&first\_random) % A;

M1[k] = value;

//printf("%.2f\n", value);

};

// Заполение массива M2

for (int k = 0; k < N / 2; k++) {

double value = 1 + rand\_r(&second\_random) % (A \* 10);

M2[k] = value;

//printf("%.2f\n", value);

}

1. **Этап Map**

6 вариантов

Согласно моему варианту

7 вариантов

Согласно моему варианту

4 вариант

void func\_map\_m1(int sizeArrayM1, double M1[]) {

for (int k = 0; k < sizeArrayM1; k++) {

M1[k] = cosh(sqrt(M1[k]))/sinh(sqrt(M1[k]));

}

}

2 вариант

void func\_map\_m2(int sizeArrayM2, double M2[]) {

double cloneArrayM2[sizeArrayM2];

for (int k = 0; k < sizeArrayM2; k++) {

cloneArrayM2[k] = M2[k];

}

for (int k = 0; k < sizeArrayM2; k++) {

if (k == 0) {

M2[k] = fabsf(cos(M2[k]));

continue;

}

M2[k] = fabsf(cos(M2[k] + cloneArrayM2[k - 1]));

}

}

1. **Этап Merge**

6 вариант

void merge(double M1[], double M2[], int sizeArrayM2) {

for (int k = 0; k < sizeArrayM2; k++) {

M2[k] = fabsf(M1[k] - M2[k]);

}

}

1. **Этап Сортировка**

Stupid sort – 4 вариант

int isSorted(double M2[], int sizeArrayM2)

{

while (--sizeArrayM2 > 1)

if (M2[sizeArrayM2] < M2[sizeArrayM2 - 1])

return 0;

return 1;

}

void swap(double\* x, double\* y)

{

double tmp = \*x;

\*x = \*y;

\*y = tmp;

}

// To generate permutation of the array

void shuffle(double M2[], int sizeArrayM2)

{

for (int i = 0; i < sizeArrayM2; i++)

swap(&M2[i], &M2[rand() % sizeArrayM2]);

}

// Sorts array a[0..n-1] using Bogo sort

void stupid\_sort(double M2[], int sizeArrayM2)

{

// if array is not sorted then shuffle

// the array again

while (!isSorted(M2, sizeArrayM2))

shuffle(M2, sizeArrayM2);

}

1. **Этап Reduce**

double reduce(double M2[], int sizeArrayM2) {

double amount = 0;

double min = 0;

for (int i = 0; i < sizeArrayM2; i++) {

if (M2[i] != 0) {

min = M2[i];

break;

}

}

for (int k = 0; k < sizeArrayM2; k++) {

if (((long)(M2[k] / min) % 2) == 0) {

amount += sin(M2[k]);

}

}

return amount;

}

**Анализ параллельных вычислений**

Глупая сортировка портит проведение исследования, убираем сортировку.

N1 = 1, N2 = 600 001

**GCC – компилятор**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **GCC** | | | | |
| **N** | **seq** | **K = 1** | **K = 2** | **K = 4** | **K = 6** |
| 1 | 0,063 | 0,053 | 0,029 | 0,05 | 0,039 |
| 60001 | 0,988 | 0,904 | 0,858 | 0,822 | 0,756 |
| 120001 | 1,224 | 1,267 | 1,219 | 1,222 | 1,299 |
| 180001 | 1,825 | 1,829 | 1,839 | 1,98 | 1,786 |
| 240001 | 2,353 | 2,353 | 2,364 | 2,559 | 2,358 |
| 300001 | 2,924 | 2,917 | 2,922 | 3,072 | 2,923 |
| 360001 | 3,452 | 3,484 | 3,509 | 3,686 | 3,48 |
| 420001 | 4,01 | 4,052 | 4,043 | 4,157 | 4,017 |
| 480001 | 4,563 | 4,607 | 4,596 | 4,642 | 4,572 |
| 540001 | 4,953 | 4,97 | 4,982 | 5,095 | 5,13 |
| 600001 | 5,665 | 5,625 | 5,679 | 5,863 | 5,595 |

**CLANG - компилятор**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **CLANG** | | | | |
| **N** | **seq** | **K = 1** | **K = 2** | **K = 4** | **K = 6** |
| 1 | 0,075 | 0,033 | 0,065 | 0,023 | 0,029 |
| 60001 | 0,865 | 0,891 | 0,877 | 0,695 | 0,703 |
| 120001 | 1,224 | 1,236 | 1,222 | 1,214 | 1,247 |
| 180001 | 1,814 | 1,776 | 1,844 | 1,785 | 1,804 |
| 240001 | 2,297 | 2,302 | 2,321 | 2,292 | 2,282 |
| 300001 | 2,852 | 2,85 | 2,927 | 2,837 | 2,835 |
| 360001 | 3,422 | 3,416 | 3,495 | 3,397 | 3,467 |
| 420001 | 3,959 | 3,974 | 4,027 | 3,938 | 3,935 |
| 480001 | 4,529 | 4,543 | 4,561 | 4,471 | 4,464 |
| 540001 | 5,069 | 5,068 | 4,955 | 5,011 | 5,081 |
| 600001 | 5,605 | 5,621 | 5,637 | 5,564 | 5,546 |

**Вывод:** в ходе выполнения лабораторной работы было проведено последовательное и автоматическое распраллеливание процесса на разных компиляторах: GCC, CLANG.В результате исследования разница времени между компиляторами выполнения была невелика, а распараллеливание не получилось на 2 компиляторах.

**Приложение**

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include <math.h>

#include <time.h>

// 2 Этап

// Применение операции соответсвенно 4 варианту к массиву М1

void func\_map\_m1(int sizeArrayM1, double M1[]) {

for (int k = 0; k < sizeArrayM1; k++) {

M1[k] = cosh(sqrt(M1[k]))/sinh(sqrt(M1[k]));

}

}

// Применение операции соответсвенно 2 варианту к массиву М2

void func\_map\_m2(int sizeArrayM2, double M2[]) {

double cloneArrayM2[sizeArrayM2];

for (int k = 0; k < sizeArrayM2; k++) {

cloneArrayM2[k] = M2[k];

}

for (int k = 0; k < sizeArrayM2; k++) {

if (k == 0) {

M2[k] = fabsf(cos(M2[k]));

continue;

}

M2[k] = fabsf(cos(M2[k] + cloneArrayM2[k - 1]));

}

}

// 3 Этап

// Merge согласно 6 варинту

void merge(double M1[], double M2[], int sizeArrayM2) {

for (int k = 0; k < sizeArrayM2; k++) {

M2[k] = fabsf(M1[k] - M2[k]);

}

}

// 4 Этап

// Сортировка stupid sort согласно 4 варианту

int isSorted(double M2[], int sizeArrayM2)

{

while (--sizeArrayM2 > 1)

if (M2[sizeArrayM2] < M2[sizeArrayM2 - 1])

return 0;

return 1;

}

void swap(double\* x, double\* y)

{

double tmp = \*x;

\*x = \*y;

\*y = tmp;

}

void shuffle(double M2[], int sizeArrayM2)

{

for (int i = 0; i < sizeArrayM2; i++)

swap(&M2[i], &M2[rand() % sizeArrayM2]);

}

void stupid\_sort(double M2[], int sizeArrayM2)

{

while (!isSorted(M2, sizeArrayM2))

shuffle(M2, sizeArrayM2);

}

// Этап reduce

// ПРИХОДИТ ОТСОРТИРОВАННЫЙ МАССИВ!!!

double reduce(double M2[], int sizeArrayM2) {

double amount = 0;

double min = 0;

for (int i = 0; i < sizeArrayM2; i++) {

if (M2[i] != 0) {

min = M2[i];

break;

}

}

for (int k = 0; k < sizeArrayM2; k++) {

if (((long)(M2[k] / min) % 2) == 0) {

amount += sin(M2[k]);

}

}

return amount;

}

int main(int argc, char\* argv[]) {

const int A = 440;

int N;

struct timeval T1, T2;

long delta\_ms;

if (argc < 2) {

printf("Need to enter arguments (sizeArray)");

return -1;

}

N = atoi(argv[1]);

gettimeofday(&T1, NULL);

unsigned int first\_random = 0;

unsigned int second\_random = 0;

// 100 экспериментов

for (int i = 0; i < 100; i++) {

srand(i);

double\* M1 = (double\*)malloc(N \* sizeof(double));

double\* M2 = (double\*)malloc(N / 2 \* sizeof(double));

// 1 Этап

// Заполение массива M1

for (int k = 0; k < N; k++) {

double value = 1 + rand\_r(&first\_random) % A;

M1[k] = value;

//printf("%.2f\n", value);

};

// Заполение массива M2

for (int k = 0; k < N / 2; k++) {

double value = 1 + rand\_r(&second\_random) % (A \* 10);

M2[k] = value;

//printf("%.2f\n", value);

}

// 2 Этап

func\_map\_m1(N, M1);

func\_map\_m2(N / 2, M2);

// 3 Этап

merge(M1, M2, N / 2);

// 4 Этап

//stupid\_sort(M2, N / 2);

// 5 Этап

int result = reduce(M2, N / 2);

printf("Result: %d\n", result);

free(M1);

free(M2);

}

gettimeofday(&T2, NULL);

delta\_ms = 1000 \* (T2.tv\_sec - T1.tv\_sec) + (T2.tv\_usec - T1.tv\_usec) / 1000;

printf("\nN=%d. Milliseconds passed: %ld\n", N, delta\_ms);

return 0;

}