МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

„ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”

# Лабораторна робота №1

## з курсу «теорія розподілених систем та паралельних обчислень»

Тема: *«Розробка та реалізація паралельного алгоритму для задач з паралелізмом даних*»

Виконала:

студентка групи ДА-42

Балан К.Г.

Варіант №1

Київ

2017

**Порядок виконання роботи**

Розробити послідовну на багатопоточну програми які реалізують варіант індивідуального завдання. Зберегти час розрахунку. Порівняти правильність виконання, порівнявши послідовний та паралельний розв’язки. Розрахувати коефієнт прискорення та коефіцієнт ефективності для декількох різних значень кількості потоків та розміру векторів або кроків інтегрування.

|  |  |
| --- | --- |
| Варіанти завдання | |
| № з/п | Опис ситуації |
| 1 | Створити 2 вектора з N>=1000 елементами з випадкових чисел. Скласти ці вектори. |

Результати:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 2 потоки | | 3 потоки | | 4 потоки | |
| 1 | N | 1 | N | 1 | N |
| 10 000 | 6 | 5 | 6 | 5 | 6 | 4 |
| 100 000 | 43 | 30 | 40 | 26 | 43 | 21 |
| 1 000 000 | 397 | 287 | 416 | 247 | 409 | 218 |
| 10 000 000 | 9806 | 5864 | 9882 | 4489 | 9936 | 3838 |

Прискорення та ефективність паралельного алгоритму:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S2 | S3 | S4 | E2 | E3 | E4 |
| 10 000 | 1.2 | 1.2 | 1.5 | 0.6 | 0.4 | 0.8 |
| 100 000 | 1.43 | 1.54 | 2.05 | 0.72 | 0.51 | 0.51 |
| 1 000 000 | 1.38 | 1.68 | 1.88 | 0.69 | 0.56 | 0.47 |
| 10 000 000 | 1.67 | 2.2 | 2.59 | 0.84 | 0.73 | 0.65 |

package lab1;  
  
*/\*\*  
 \* Created by Katya on 21.02.2017.  
 \*/*class ThreadCacl extends Thread{  
  
 double vectA[];  
 double vectB[];  
 int startIndex;  
 int endIndex;  
 double[] result;  
  
 public ThreadCacl(double[] vectA, double[] vectB, int startIndex, int endIndex) {  
 this.vectA = vectA;  
 this.vectB = vectB;  
 this.startIndex = startIndex;  
 this.endIndex = endIndex;  
 this.result = new double[endIndex - startIndex];  
 }  
  
 public double[] getResult() {  
 return result;  
 }  
  
 //@Override  
 public void run(){  
 for(int i = 0; i<endIndex - startIndex; i++ ){  
 result[i]= Math.*sin*(vectA[startIndex + i]) + Math.*cos*(vectB[startIndex + i]);  
 }  
 }  
  
}

package lab1;  
import java.lang.Math;  
  
import com.sun.prism.shader.Solid\_ImagePattern\_Loader;  
  
*/\*\*  
 \* Created by Katya on 21.02.2017.  
 \*/*public class ThreadSample {  
 public static int *SIZE* = 10000000;  
 public static int *NUNMBER\_JOBS* = 2;  
 public static void main(String [] args ) throws InterruptedException{  
  
 long startTime = System.*currentTimeMillis*();  
 double vectA[] = new double [*SIZE*];  
 double vectB[] = new double [*SIZE*];  
  
 for(int i =0; i<*SIZE*; i++){  
 vectA[i]=i;  
 vectB[i]=i;  
 }  
  
 double[] serialResult = new double[*SIZE*];  
 for( int i=0; i< *SIZE*; i++){  
 serialResult[i] = Math.*sin*(vectA[i]) + Math.*cos*(vectB[i]);  
 //System.out.println(serialResult[i]);  
 }  
 long timeSpent = System.*currentTimeMillis*() - startTime;  
 System.*out*.println("Програма виконалась послідовно за " + timeSpent + " мілісекунд");  
  
  
 long startTime2 = System.*currentTimeMillis*();  
 ThreadCacl TreadArrray[] = new ThreadCacl[*NUNMBER\_JOBS*];  
  
 for(int i = 0; i < *NUNMBER\_JOBS*; i++){  
  
 TreadArrray[i] = new ThreadCacl(vectA ,vectB,  
 *SIZE*/*NUNMBER\_JOBS* \* i,  
 i==*NUNMBER\_JOBS*-1 ?*SIZE*:*SIZE*/*NUNMBER\_JOBS* \* (i + 1) );  
 TreadArrray[i].start();  
 }  
 for(int i = 0; i < *NUNMBER\_JOBS*; i++){  
 TreadArrray[i].join();  
 }  
 double[] parallelResult = new double[*SIZE*];  
 for(int i = 0; i < *NUNMBER\_JOBS*; i++) {  
 int start = *SIZE* / *NUNMBER\_JOBS* \* i;  
 int end = i == *NUNMBER\_JOBS* - 1 ? *SIZE* : *SIZE* / *NUNMBER\_JOBS* \* (i + 1);  
 double[] part\_res = new double[end - start];  
 part\_res = TreadArrray[i].getResult();  
 for (int j = 0; j < end - start; j++) {  
 parallelResult[start + j] = part\_res[j];  
 //System.out.println(parallelResult[start + j]);  
  
 }  
 }  
 long timeSpent2 = System.*currentTimeMillis*() - startTime2;  
 System.*out*.println("Програма виконалась паралельно за " + timeSpent2 + " мілісекунд");  
  
 boolean equality = true;  
 for( int i=0; i< *SIZE*; i++){  
 if ( parallelResult[i] != serialResult[i] ){  
 equality = false;  
 }  
 }  
 System.*out*.println(equality);  
  
 }  
}