

Звіт

Автор: Пумня О., КІТ118Б

Дата: 20.02.2020

Лабораторна робота №14

ПАРАЛЕЛЬНЕ ВИКОНАННЯ. ЕФЕКТИВНІСТЬ ВИКОРИСТАННЯ

Мета. Вимірювання часу паралельних та послідовних обчислень. Демонстрація ефективності паралельної обробки.

Вимоги:

1. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.
2. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.
3. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.
4. Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:
 - результати вимірювання часу звести в таблицю;
 - обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

ОПИС ПРОГРАМИ

Опис змінних

`GenericList<Integer> numbers` – список чисел

`Thread thr1` – потік, що шукає кількість парних і непарних чисел масиву

`Thread thr2` – потік, що рахує середнє значення масиву чисел

`Thread thr3` – потік, що шукає мінімальне та максимальне значення масиву чисел

Ієрархія та структура класів

`Pumnya14` – точка входу в програму

ТЕКСТ ПРОГРАМИ

Текст файлу `Pumnya14.java`

```
package labs.pumnya14;
import labs.pumnya09.GenericList;
import labs.pumnya13.FirstThread;
import labs.pumnya13.SecondThread;
import labs.pumnya13.ThirdThread;
import javax.swing.*;
import java.awt.*;
```

```

import java.util.Random;

public class Pumnya14 {
    /** Количество наносекунд в одной миллисекунде. */
    private static final int DIVIDER = 1_000_000;
    /** Приватный конструктор утилитарного класса. */
    private Pumnya14() {
        // Пустое тело
    }

    /**
     * Создаёт таблицу.
     * @param seqTime время последовательной обработки
     * @param concurTime время одновременной обработки
     * @param timeDiff разница во времени обработок
     */
    public static void generateTable(double seqTime, double concurTime, double timeDiff) {
        String[] columnNames = {
            "Одновременная обработка",
            "Последовательная обработка",
            "Во сколько раз последовательная дольше одновременной"
        };
        String[][] data = {
            {Double.toString(seqTime),
             Double.toString(concurTime),
             Double.toString(timeDiff)}
        };
        JTable table = new JTable(data, columnNames);
        JScrollPane scrollPane = new JScrollPane(table);
        JFrame frame = new JFrame("ВЫВОД");
        frame.getContentPane().add(scrollPane);
        frame.setPreferredSize(new Dimension(1100, 200));
        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }

    public static void main(String[] args) throws InterruptedException {
        GenericList<Integer> list = new GenericList<>();
        for (int i = 0; i < 10000000; i++) {
            list.pushBack(new Random().nextInt(20000));
        }
        Thread thr1 = new FirstThread(list, 5000);
        Thread thr2 = new SecondThread(list, 5000);
        Thread thr3 = new ThirdThread(list, 5000);

        double concurrentTime = System.nanoTime();
        thr1.start();
        thr2.start();
        thr3.start();

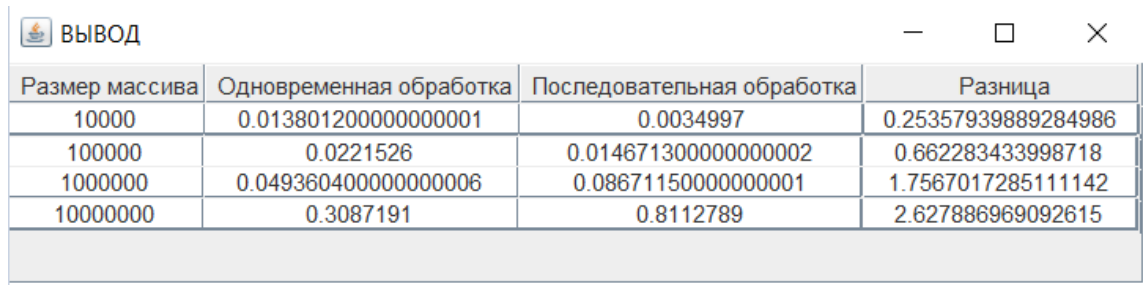
        thr1.join();
        thr2.join();
        thr3.join();
        concurrentTime = (System.nanoTime() - concurrentTime) * 10e-10;
        System.out.print("\n");

        double sequentialTime = System.nanoTime();
        thr1.run();
        thr2.run();
        thr3.run();
        sequentialTime = (System.nanoTime() - sequentialTime) * 10e-10;

        double timeDiff = sequentialTime / concurrentTime;
        Pumnya14.generateTable(concurrentTime, sequentialTime, timeDiff);
    }
}

```

ВАРІАНТИ ВИКОРИСТАННЯ



Размер массива	Одновременная обработка	Последовательная обработка	Разница
10000	0.013801200000000001	0.0034997	0.25357939889284986
100000	0.0221526	0.014671300000000002	0.662283433998718
1000000	0.049360400000000006	0.086711500000000001	1.7567017285111142
10000000	0.3087191	0.8112789	2.627886969092615

Рисунок 1 – Результат роботи програми

ВИСНОВКИ

При виконанні лабораторної роботи детальніше розглянуто матеріал попередньої роботи, набуто практичних навичок вимірювання часу виконання програми засобами Java SE. На рисунку 1 зображено порівняння виконання дій паралельно та послідовно із різною кількістю елементів масиву, з чого випливає результат – паралельне виконання набагато ефективніше при великій кількості елементів. Для порівняння з 100 000 000 елементами недостатньо пам'яті.