Звіт

Лабораторна работа 14. Паралельне виконання. Ефективність використання

Мета роботи:

- 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.

1.1. Розробник: Момот Роман Євгенійович, КІТ119а, варіант №14.

2. ОПИС ПРОГРАМИ

2.1. Засоби ООП: клас, метод класу, поле класу.

2.2. Ієрархія та структура класів: один публічний клас **Main**, публічний клас **Event**, у полях якого є час початку події, тривалість, адреса події, імена людей, опис події, гетери, сетери, конструктор класу та метод виведення даних класу. Також є клас **Node**, який виконує роль покажчика на елемент і клас **MyContainer**, який містить покажчик на головний елемент та методи обробки масиву елементів. Клас **MyThread**, який виконує роль потока.

2.3. Важливі фрагменти програми:

```
public class MyThread implements Runnable{
```

```
private MyContainer<Event> arr;
Thread thread;
private boolean isActive;
private int count;

void disable() {
	isActive = false;
}

MyThread(MyContainer<Event> arr, String name, int count){
	this.arr = arr;
	thread = new Thread(this, name);
	this.count = count;
	isActive = true;
}

@Override
public void run() {
	long countTime = 0;
	long temp = 0;
	int i;

	for(i = 0; i < count; i++) {

		try {
			temp = countPeople();
		} catch (InterruptedException e) {
			e.printStackTrace();
```

```
                }

                //System.out.println(Thread.currentThread().getName() + ": " +
temp);
                countTime += temp;
            }

            System.out.println("Time spent: " + countTime + " milliseconds\n");
            //System.out.println("Time result: " + (double)((countTime/3)/1000));
        }

        private long countPeople() throws InterruptedException {
            long count = 0;
            long begin = System.currentTimeMillis();
            Thread.currentThread().sleep(1000);

            for(Event i : arr) {
                if(isActive) {
                    count += i.getDuration();
                }
                else {
                    System.out.println(Thread.currentThread().getName() + "
was stopped.");
                    return -1;
                }
            }


            System.out.println(Thread.currentThread().getName() + ": " + count +
" people");
```

```
                System.out.println(Thread.currentThread().getName() + " finished");

                return (System.currentTimeMillis() - begin);
        }
}

private static MyContainer<Event> menu(MyContainer<Event> arr) {
                Scanner scan = new Scanner(System.in);
                boolean stop = false;
                int choise, choise2;

                ArrayList<String> people = new ArrayList<String>();
                people.add("John");
                people.add("Bill");
                people.add("Івасик");
                Event evToCompare = new Event(new
GregorianCalendar(2002,3,28), 120, "ул. Революции",
                                people, "Pest party ever");
                arr.add(evToCompare);

                do {
                        System.out.println("What to do?");
                        System.out.println("1. Output data");
                        System.out.println("2. Add element");
                        System.out.println("3. Delete element");
                        System.out.println("4. Is empty?");
                        System.out.println("5. Serialization");
                        System.out.println("6. Deserialization");
                        System.out.println("7. Sort data");
```

```
                System.out.println("8. Find the number of people
(Multithreading)");
                System.out.println("9. Terminate program");
                System.out.println("====================");
                System.out.print("Your choise: ");
                choise = scan.nextInt();

                switch(choise) {
                case 1:
                        System.out.println("\nChoose the output method");
                        System.out.println("1. Using foreach");
                        System.out.println("2. Using toArray");
                        System.out.println("3. Find element by criteria");
                        System.out.println("4. Return");
                        System.out.println("================");
                        System.out.print("Your choise: ");
                        choise2 = scan.nextInt();
                        System.out.println( );

                        switch(choise2) {
                        case 1:
                                if(arr.getSize() > 0){
                                        for(var i : arr) {
                                                i.outputData();
                                                System.out.println( );
                                        }
                                        System.out.println( );
                                }
                                else {
```

```
                System.out.println("Array is empty.\n");
            }
            break;

        case 2:
            if(arr.getSize() > 0) {
                Object[] tempArr = arr.toArray();
                for (int i = 0; i < tempArr.length; i++) {
                    System.out.println(i + ")");
                    ((Event)tempArr[i]).outputData();
                }
            }
            else {
                System.out.println("Array is empty.");
            }
            break;
        case 3:
            if(arr.getSize() == 0) {
                System.out.println("Array is empty.\n");
                break;
            }

            Pattern pattYear;
            Pattern pattCity;
            Pattern pattDuration = Pattern.compile("^([2][5-
9]+)|([3-9][0-9]+)|([1-9][0-9]{2,})$");
            Matcher matcher1, matcher2, matcher3;

            String regex = "^(?)|(?)|(?)$";
```

```
                System.out.println("Task: Знайти всі конференції, що пройшли\n"
                        + "-за останні три роки "
                        + "\n-в Харкові та області "
                        + "\n-з тривалістю не менше доби.");
                System.out.println("Передбачити можливість незначної зміни умов пошуку.");

                System.out.print("\nEnter the year: ");
                int year = scan.nextInt();
                for (int i = 0; i < 3; i++) {
                    regex = regex.substring(0,regex.indexOf('?')) + Integer.toString(year - i) + regex.substring(regex.indexOf('?') + 1, regex.length());
                }
                pattYear = Pattern.compile(regex);

                System.out.print("Enter the city: ");
                scan.nextLine();
                String city = scan.nextLine();
                city = city.concat("(.*)");
                pattCity = Pattern.compile(city);

                for(var i : arr) {
                    matcher1 = pattYear.matcher(Integer.toString(i.getStartTime().get(Calendar.YEAR)));
                    matcher2 = pattCity.matcher(i.getAddress());
                    matcher3 = pattDuration.matcher(Integer.toString(i.getDuration()));
```

```
                                System.out.println( );
                                if(matcher1.matches() &&
matcher2.matches() && matcher3.matches()) {
                                        i.outputData();
                                }
                        }

                        break;

                case 4:
                        System.out.println("\nReturning\n");
                        break;

                default:
                        System.out.println("You've entered the wrong
number");
                        break;
                }
                break;

        case 2:
                Event newEvent = inputNewEvent();
                arr.add(newEvent);

                break;

        case 3:
                if(arr.getSize() > 0) {
                        System.out.print("Enter the index of element: ");
```

```
                choise = scan.nextInt();

                arr.delete(choise);
            } else {
                System.out.println("Array is empty.");
            }
            break;

        case 4:
            if(arr.isEmpty()) {
                System.out.println("Array is empty.");
            } else {
                System.out.println("Array isn't empty.");
            }
            break;

        case 5:
            System.out.println("\nChoose the method");
            System.out.println("1. Standard serialization");
            System.out.println("2. XML serialization");
            System.out.println("3. Return");
            System.out.println("========================");
            System.out.print("Your choise: ");
            choise2 = scan.nextInt();

            switch(choise2) {
            case 1:
                scan.nextLine();
                System.out.print("Enter the name of file: ");
```

```
                    String filename = scan.nextLine();

                    if (filename.indexOf(".ser") == -1) {
                        filename += ".ser";
                    }

                    try(ObjectOutputStream oos = new
ObjectOutputStream(new BufferedOutputStream(new
FileOutputStream(filename)))){
                        oos.writeObject(arr);
                        System.out.println("Serialization
successful.");
                    }catch(Exception ex){
                        System.out.println(ex.getMessage());
                        ex.printStackTrace();
                    }

                    break;

                case 2:
                    scan.nextLine();
                    System.out.print("Enter the name of file: ");
                    filename = scan.nextLine();

                    if (filename.indexOf(".xml") == -1) {
                        filename += ".xml";
                    }

                    try(XMLEncoder encoder = new
XMLEncoder(new BufferedOutputStream(new FileOutputStream(filename)))){
```

```
                        encoder.writeObject(arr);
                        System.out.println("Serialization
successful.");
                    }
                    catch(Exception ex){
                        System.out.println(ex.getMessage());
                    }
                    break;

                case 3:
                    break;

                default:
                    System.out.println("You've entered the wrong
command.");
                    break;
                }

                break;

            case 6:
                System.out.println("\nChoose the method");
                System.out.println("1. Standard deserialization");
                System.out.println("2. XML deserialization");
                System.out.println("3. Return");
                System.out.println("========================");
                System.out.print("Your choise: ");
                choise2 = scan.nextInt();
```

```
                    switch(choise2) {
                    case 1:
                        scan.nextLine();
                        System.out.print("Enter the name of file: ");
                        String filename = scan.nextLine();

                        if (filename.indexOf(".ser") == -1) {
                            filename += ".ser";
                        }

                        try(ObjectInputStream oos = new
ObjectInputStream(new BufferedInputStream(new FileInputStream(filename)))){
                            arr.clear();
                            arr = (MyContainer<Event>)
oos.readObject();
                            System.out.println("\nSerialization
successful.");
                        }catch(Exception ex){
                            System.out.println(ex.getMessage());
                        }


                        break;

                    case 2:
                        scan.nextLine();
                        System.out.print("Enter the name of file: ");
                        filename = scan.nextLine();

                        if (filename.indexOf(".xml") == -1) {
```

```
                    filename += ".xml";
                }

                try(XMLDecoder decoder = new
XMLDecoder(new BufferedInputStream(new FileInputStream(filename)))){
                    arr.clear();
                    arr = (MyContainer<Event>)
decoder.readObject();
                    System.out.println("Serialization
successful.\n");
                }catch(IOException ex){
                    System.out.println( );
                }
                break;

            case 3:
                break;

            default:
                System.out.println("You've entered the wrong
command.");
                break;
            }

            break;
        case 7:
            System.out.println("\nChoose sorting field:");
            System.out.println("1. Sort by event date");
            System.out.println("2. Sort by event length");
            System.out.println("3. Sort by number of people");
```

```
                System.out.println("4. Return");

        System.out.println("============================");
                System.out.print("Your choise: ");
                choise2 = scan.nextInt();

                switch(choise2) {
                case 1:
                        arr.sort(new EventDateComparator());
                        System.out.println("\nData sorted\n");
                        break;

                case 2:
                        arr.sort(new EventLengthComparator());
                        System.out.println("\nData sorted\n");
                        break;

                case 3:
                        arr.sort(new EventPeopleNumberComparator());
                        System.out.println("\nData sorted\n");
                        break;

                case 4:
                        System.out.println("\nReturning\n");
                        break;

                default:
                        System.out.println("\nYou have entered the wrong
number.\n");
```

```
                    break;
                }
                break;
            case 8:
                final int ARR_SIZE = 30000;
                final int NUBER_OF_THREADS;
                final int NUMBER_OF_ITERATIONS;

                int time = -1;
                int numberOfPeople;
                int choise3;
                long time1, time2;

                MyContainer<Event> newArr = new
MyContainer<Event>();
                ArrayList<String> newPeople = new
ArrayList<String>();

                System.out.println("\nCreating new array...");
                System.out.println("Adding new elements...");
                for(int i = 0; i < ARR_SIZE; i++) {
                    numberOfPeople = (int) (2 + Math.random() * 10);
                    for(int j = 0; j < numberOfPeople; j++) {
                        newPeople.add("j");
                    }
                    newEvent = new Event(new GregorianCalendar(),
i, Integer.toString(i), newPeople, Integer.toString(i));
                    newArr.add(newEvent);
                    newPeople = new ArrayList<String>();
                }
```

```
                    System.out.println("\nWhat do you want?");
                    System.out.println("1. Parallel calculations");
                    System.out.println("2. Serial calculations");
                    System.out.println("====================");
                    System.out.print("What do you want: ");
                    choise3 = scan.nextInt();
                    System.out.println( );

                    if(choise3 != 1 && choise3 != 2) {
                        System.out.println("You have entered the wrong
command");
                        break;
                    }

//                  System.out.print("Want to set a maximum lead time? ");
//                  if((choise3 = scan.nextBoolean()) == true) {
//                      System.out.print("Enter the time in milliseconds:
");
//                      time = scan.nextInt();
//                  }

                    if(choise3 == 1) {
                        NUBER_OF_THREADS = 3;
                        NUMBER_OF_ITERATIONS = 1;
                    }
                    else {
                        NUBER_OF_THREADS = 1;
                        NUMBER_OF_ITERATIONS = 3;
```

```
                }


                MyThread[] threads = new
MyThread[NUBER_OF_THREADS];

                try {
                    for(int i = 0; i < NUBER_OF_THREADS; i++) {
                        threads[i] = new MyThread(newArr,
"Parallel thread " + (i+1), NUMBER_OF_ITERATIONS);
                        threads[i].thread.start();
                    }

//                  if(time > 0) {
//                      Thread.currentThread().sleep(time);
//
//                      for(int i = 0; i < NUBER_OF_THREADS;
i++) {
//                          threads[i].disable();
//                      }
//                  }
                    time1 = System.currentTimeMillis();
                    for(int i = 0; i < NUBER_OF_THREADS; i++) {
                        threads[i].thread.join();
                    }
                    time2 = System.currentTimeMillis();

                    //System.out.println(time1 + "\t" + time2);
                    System.out.println("Time result: " + (double)(time2
- time1)/1000 + " seconds");
```

```
                    }
                    catch(InterruptedException ex) {
                        System.out.println("Thread has been interrupted.");
                    }

                    //newArr.clear();
                    System.out.println( );
                    break;

                case 9:
                    System.out.println("\nTerminating the program.");
                    stop = true;
                    break;

                default:
                    System.out.println("You have entered the wrong
number.");
                    break;
                }
            }while(!stop);

            scan.close();
            return arr;
        }
```

Результат роботи програми

```
What to do?
1. Output data
2. Add element
3. Delete element
4. Is empty?
5. Serialization
6. Deserialization
7. Sort data
8. Find the number of people (Multit
9. Terminate program
====================
Your choise: 8

Creating new array...
Adding new elements...

What do you want?
1. Parallel calculations
2. Serial calculations
====================
What do you want: 1

Parallel thread 1: 449985000 people
Parallel thread 1 finished
Time spent: 5649 milliseconds

Parallel thread 2: 449985000 people
Parallel thread 2 finished
Time spent: 5695 milliseconds

Parallel thread 3: 449985000 people
Parallel thread 3 finished
Time spent: 5715 milliseconds

Time result: 5.715 seconds
```

```
What to do?
1. Output data
2. Add element
3. Delete element
4. Is empty?
5. Serialization
6. Deserialization
7. Sort data
8. Find the number of people (Multit
9. Terminate program
====================
Your choise: 8

Creating new array...
Adding new elements...

What do you want?
1. Parallel calculations
2. Serial calculations
====================
What do you want: 2

Parallel thread 1: 449985000 people
Parallel thread 1 finished
Parallel thread 1: 449985000 people
Parallel thread 1 finished
Parallel thread 1: 449985000 people
Parallel thread 1 finished
Time spent: 17584 milliseconds

Time result: 17.584 seconds
```

Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з багатопоточністтю.

Програма протестована, виконується без помилок.