Міністерство освіти і науки України

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Олеся Гончара

Факультет фізики, електроніки та комп’ютерних систем

Кафедра електронних обчислювальних машин

Звіт

з лабораторної роботи №1

Виконав студент групи КІ-21-2 Браіла О.А.

Керівник: Скуратовський І. А.

м. Дніпро

2024

Завдання до лабораторної роботи:

1. Написати програму в java, яка виконує наступне:

2. Створити колекцію із 10 мільйонів випадкових цілих значень в діапазоні від 1 до 100

3. за допомогою Stream API (кожне завдання робити окремим методом для того, щоб використати його в бенчмарку п. 4):

      - знайти суму елементів в колекції

      - знайти середнє значення елементів колекції

      - знайти стандартне відхилення

      - помножити на 2 кожен елемент вихідної колекції

      - відфільтрувати елементи колекції таким чином, щоб залишились лише парні значення, які діляться на 3 націло

4. Виконати ті самі дії за допомогою parallel stream

5. Оцінити затрачений час на кожну задачу із пп. 2-3 за допомогою JMH (https://github.com/openjdk/jmh). Бенчмарк методи не робити void, повертати з них розраховані значення (або якимось іншим способом запобігти оптимізації невикористовуваних значень).

6. Порівняльну таблицю роботи послідовного Stream API і parallel stream навести у звіті

7. У звіт додати посилання на Git репозиторій з кодом.

Результат виконання програми:

A screenshot of a computer

Description automatically generated

Рис.1. – Паралельний бенчмарк

A screen shot of a computer

Description automatically generated

Рис.2. – Послідовний бенчмарк

A screenshot of a computer

Description automatically generated

Рис.3. – Результат виконання програми

Таблиця:

|  |  |  |  |
| --- | --- | --- | --- |
| Операція | Паралельний потік (ops/s) | Послідовний потік (ops/s) | Прискорення (разів) |
| Середнє значення (average) | 1,924 | 31,091 | ~ 16,16 |
| Фільтрація (filter) | 31,091 | 232,186 | ~ 7,46 |
| Множення (multiply) | 403,399 | 82,228 | ~ -0,2 |
| Стандартне відхилення | 4,685 | 47,560 | ~ 10,15 |
| Сума (sum) | 2,034 | 6,926 | ~ 3,4 |

Код файлу BenchmarkStart.java

package com.dnu.ffecs;  
import java.io.IOException;  
public class BenchmarkStart {  
 public static void main(String[] args) throws IOException {  
 org.openjdk.jmh.Main.*main*(args);  
 }  
}

Код файлу ParallelStreamBenchmark.java

package com.dnu.ffecs;  
  
import org.openjdk.jmh.annotations.\*;  
import org.openjdk.jmh.infra.Blackhole;  
  
import java.util.List;  
import java.util.Random;  
import java.util.concurrent.TimeUnit;  
import java.util.stream.Collectors;  
  
@Fork(value = 1)  
@State(Scope.*Thread*)  
@Warmup(iterations = 1)  
@Measurement(iterations = 1)  
public class ParallelStreamBenchmark {  
  
 private List<Long> numbers;  
  
 @Setup(Level.*Trial*)  
 public void setup() {  
 numbers = new Random().longs(10\_000\_000, 1, 101)  
 .boxed().collect(Collectors.*toList*());  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public long computeSum(Blackhole blackhole) {  
 long sum = numbers.parallelStream().mapToLong(Long::longValue).sum();  
 blackhole.consume(sum);  
 return sum;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public double computeAverage(Blackhole blackhole) {  
 double average = numbers.parallelStream().mapToLong(Long::longValue)  
 .average().orElseThrow(RuntimeException::new);  
 blackhole.consume(average);  
 return average;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public double calculateStdDev(Blackhole blackhole) {  
 double mean = computeAverage(blackhole);  
 double stdDev = Math.*sqrt*(numbers.parallelStream().mapToDouble(num -> Math.*pow*(num - mean, 2))  
 .average().orElseThrow(RuntimeException::new));  
 blackhole.consume(stdDev);  
 return stdDev;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public List<Long> doubleValues(Blackhole blackhole) {  
 List<Long> doubledValues = numbers.parallelStream().map(num -> num \* 2).collect(Collectors.*toList*());  
 blackhole.consume(doubledValues);  
 return doubledValues;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public List<Long> evenMultiples(Blackhole blackhole) {  
 List<Long> multiples = numbers.parallelStream().filter(num -> num % 2 == 0 && num % 3 == 0).collect(Collectors.*toList*());  
 blackhole.consume(multiples);  
 return multiples;  
 }  
}

Код файлу SequentialStreamBenchmark.java

package com.dnu.ffecs;  
  
  
import org.openjdk.jmh.annotations.\*;  
import org.openjdk.jmh.infra.Blackhole;  
  
import java.util.List;  
import java.util.Random;  
import java.util.concurrent.TimeUnit;  
import java.util.stream.Collectors;  
  
@Fork(value = 1)  
@State(Scope.*Thread*)  
@Warmup(iterations = 1)  
@Measurement(iterations = 1)  
public class SequentialStreamBenchmark {  
  
 private List<Long> numbers;  
  
 @Setup(Level.*Trial*)  
 public void setup() {  
 numbers = new Random().longs(10\_000\_000, 1, 101)  
 .boxed().collect(Collectors.*toList*());  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public long calculateSum(Blackhole blackhole) {  
 long sum = numbers.stream().mapToLong(Long::longValue).sum();  
 blackhole.consume(sum);  
 return sum;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public double calculateAverage(Blackhole blackhole) {  
 double average = numbers.stream().mapToLong(Long::longValue)  
 .average().orElseThrow(RuntimeException::new);  
 blackhole.consume(average);  
 return average;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public double computeStdDev(Blackhole blackhole) {  
 double mean = calculateAverage(blackhole);  
 double stdDev = Math.*sqrt*(numbers.stream().mapToDouble(num -> Math.*pow*(num - mean, 2))  
 .average().orElseThrow(RuntimeException::new));  
 blackhole.consume(stdDev);  
 return stdDev;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public List<Long> doubleNumbers(Blackhole blackhole) {  
 List<Long> doubledValues = numbers.stream().map(num -> num \* 2).collect(Collectors.*toList*());  
 blackhole.consume(doubledValues);  
 return doubledValues;  
 }  
  
 @Benchmark  
 @BenchmarkMode(Mode.*AverageTime*)  
 @OutputTimeUnit(TimeUnit.*MILLISECONDS*)  
 public List<Long> selectMultiples(Blackhole blackhole) {  
 List<Long> multiples = numbers.stream().filter(num -> num % 2 == 0 && num % 3 == 0).collect(Collectors.*toList*());  
 blackhole.consume(multiples);  
 return multiples;  
 }  
}

Посилання на Git репозиторій з кодом

<https://github.com/brailaalexx/ABP-lab1>