
	Politechnika Bydgoska im. J.J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
<b>Przedmiot:</b>	Programowanie współbieżne	IST Studia stacjonarne Semestr 5, 2021/2022	
<b>Temat:</b>	Silnia		
<b>Numer lab.:</b>	3	<b>Data wykonania:</b>	2021.10.27
<b>Prowadzący</b>	mgr inż. Karol Hartwig	<b>Data oddania:</b>	2022.01.07
<b>Autor:</b>	Mariusz Jackowski	<b>Indeks:</b>	113031

### Instrukcja:

- Napisać dwie funkcje do liczenia silni z wartości 100000 i większych. W funkcji wykorzystać StreamApi. Funkcja pierwsza klasyczny strumień, funkcja druga zrównoleglony (ParallelStream lub stream().parallel() ).
- Zmierzyć i porównać czas wykonywania obu funkcji.
- Sprawdzić jak będzie się zmieniać różnica czasów wykonywania dla coraz większych wartości (100.000, 200.000, 500.000, 1.000.000)
- Sprawdzić ilość wykorzystywanych wątków podczas pracy funkcji.

### Repozytorium:



Aplikacja składa się z strumienia integerów w zakresie 1 do x. Każda wartość jest mapowana do BigInteger z powodu wielkości powstałej liczby.

```

public static void main(String[] args) {
    Instant start;
    Instant end;
    int rangeStart = 1;
    int rangeStop = 100000;
    System.out.println("Silnia: "+rangeStop);
    System.out.println("Stream: ");
    start = Instant.now();
    Silnia(rangeStart, rangeStop);
    end = Instant.now();
    System.out.println("Czas: " + Duration.between(start, end).toMillis()
        + " milliseconds.");
    System.out.println("Parallel stream: ");
    start = Instant.now();
    ParallelSilnia(rangeStart, rangeStop);
    end = Instant.now();
    System.out.println("Czas: " + Duration.between(start, end).toMillis()
        + " milliseconds.");
}

```

Liczenie stream

```

private static Optional<BigInteger> Silnia(int rangeStart, int rangeStop)
{
    return IntStream.rangeClosed(rangeStart, rangeStop) IntStream
        .mapToObj(BigInteger::valueOf) Stream<BigInteger>
        .reduce((x, y) -> x.multiply(y));
}

```

Liczenie paralel stream

```

private static Optional<BigInteger> ParallelSilnia(int rangeStart, int rangeStop) {
    return IntStream.rangeClosed(rangeStart, rangeStop) IntStream
        .parallel()
        .mapToObj(BigInteger::valueOf) Stream<BigInteger>
        .reduce((x, y) -> x.multiply(y));
}

```

Wyniki:

Silnia: 100000	Silnia: 200000
Stream:	Stream:
Czas: 12446 milliseconds.	Czas: 53723 milliseconds.
Parallel stream:	Parallel stream:
Czas: 547 milliseconds.	Czas: 1734 milliseconds.

Silnia: 500000	Silnia: 1000000
Stream:	Stream:
Czas: 398864 milliseconds.	Czas: 1989468 milliseconds, 33 minut.
Parallel stream:	Parallel stream:
Czas: 8711 milliseconds.	Czas: 33547 milliseconds 0 minut.

Wnioski

Stream domyślnie działa na jednym wątku, aby wykorzystać wiele wątków należy użyć `parallelStream()`. Jak też widać w przypadku tego zadania w wyżej wymienionych wynikach gdy jest więcej danych do przetworzenia `parallelStream` jest dużo szybszy nie tak jak w naszym 1 zadaniu o Pizzy. Patrząc na wyniki warto wspomnieć, że robię to na laptopie i zastanawiam się czy aby nie opłacało się uruchomić komputer Desktop by zaoszczędzić na czasie.