Teoria Współbieżności

Sprawozdanie 1 - Badania porównawcze dwóch rozwiązań PK z losowością — bez zagłodzenia

Autorka:
Paulina Jędrychowska (grupa 10, śr. 11:20)

1. Dane techniczne urządzenia i wykorzystane narzędzia:

System: Windows 10 x64
Procesor: i5-9300H
Rdzenie: 4
Procesory logiczne: 8
Pamięć RAM: 16GB
Środowisko: IntelliJ IDEA 2021.2.2
Język programowania: Java 16

2. Opis pomiarów:

W eksperymentach był mierzony czas wykonywania programu dla różnych wartości parametrów. Kod po wykonaniu zadeklarowanej ilości operacji (konsumpcji i produkcji) był wyłączany z pomocą funkcji systemowej System.exit(0). Nie był brany pod uwagę rozmiar porcji, jako jedną operację zliczano jedno wykonanie metody produce() lub consume(). Czas mierzony był w milisekundach.

Wszystkie pomiary były wykonywane pięciokrotnie, a następnie uśrednione.

Pomiar czasu CPU zrobiłam niepoprawnie, ponieważ zlicza on czas tylko na wątku zliczającym ilość operacji. Natomiast postanowiłam nie usuwać tych obliczeń, ponieważ można zaobserwować, jak dużo czasu program spędza na operacji zliczania, która jest niejako błędem implementacji i psuje precyzję pomiarów.

3. Eksperymenty:

1. Porównanie czasów dla różnej ilości operacji:

   Ustawienia eksperymentu:

- liczba producentów: 5
- liczba konsumentów: 5
- maksymalna porcja: 10
- rozmiar bufora: 20
- generator liczb losowych: Math.random()

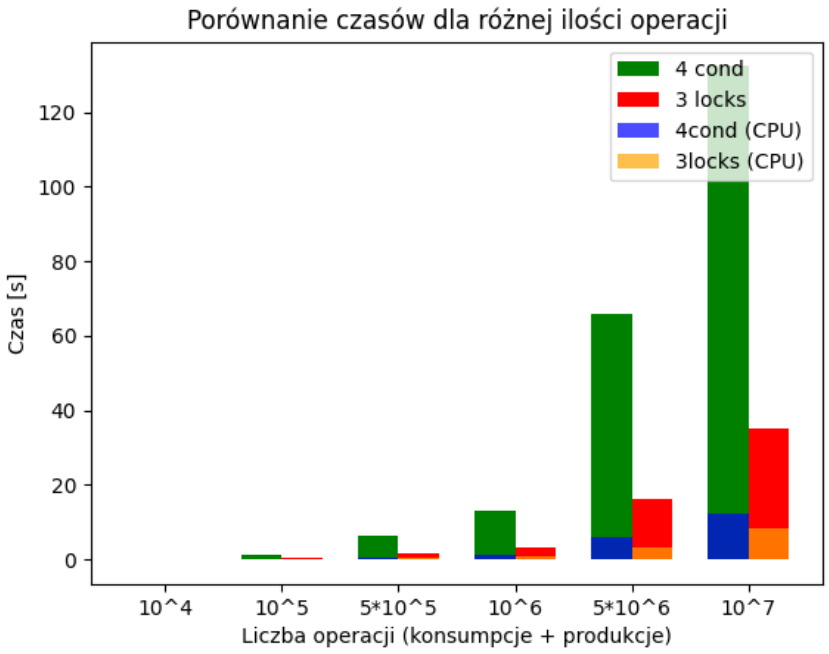
Testowane wartości:

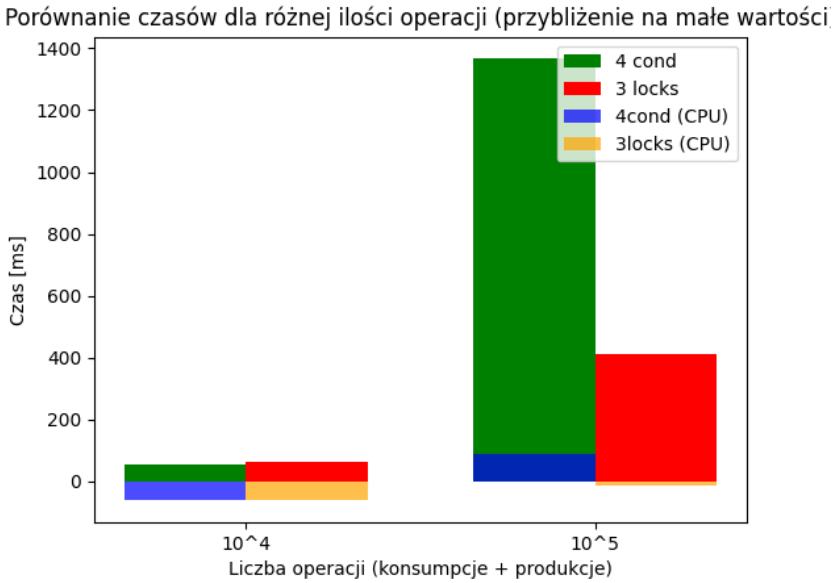
- ilość operacji: [10^4, 10^5, 5*10^5, 10^6, 5*10^6, 10^7]

Wartość mierzona:

- Czas wykonywania
- Czas CPU

Wyniki:





Wnioski:

Spodziewanie rozwiązanie na 4 condition zawsze ma dłuższy czas wykonywania niż to na 3 lockach. Wyjątkiem jest pomiar dla 10^4 operacji, natomiast czas wykonywania jest tak krótki, że mogło to być kwestią niepewności pomiarowej. We wszystkich pozostałych pomiarach czas wykonywania dla 4 condition był około 3.5 raza większy.

Z pomiarów czasu CPU dla operacji zliczania można zauważyć, że stanowi on sporą część pomiaru (około 20% dla 3 locków i 10% dla 4 condition).

2. Porównanie czasów dla różnych generatorów liczb losowych:

Ustawienia eksperymentu:

- liczba producentów: 5
- liczba konsumentów: 5
- maksymalna porcja: 10
- rozmiar bufora: 20
- ilość operacji: 5 * 10^5

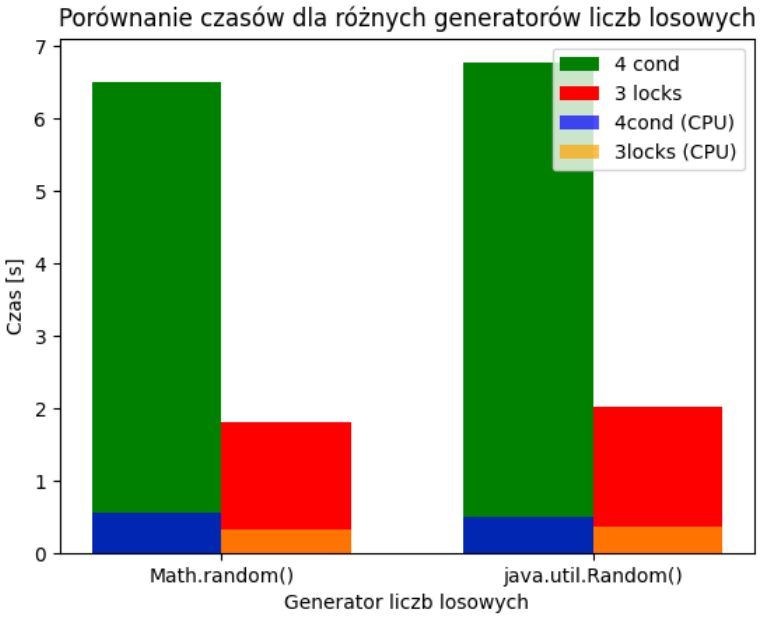
Testowane wartości:

- generator liczb losowych: [Math.random(), java.util.Random()]

Wartość mierzona:

- Czas wykonywania
- Czas CPU

Wyniki:



Wnioski:

Nie zauważono szczególnych różnic pomiędzy dwoma generatorami liczb losowych.

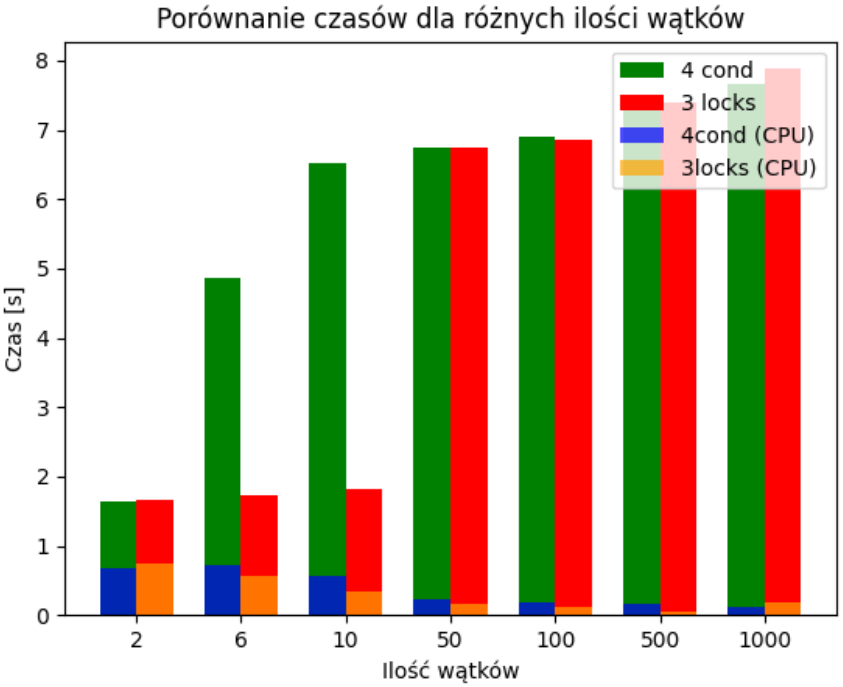
3. Porównanie czasów dla różnych ilości wątków:
   Ustawienia eksperymentu:

   - maksymalna porcja: 10
   - rozmiar bufora: 20
   - ilość operacji: 5 * 10^5
   - generator liczb losowych: Math.random()

   Testowane wartości:
   - ilość wątków: [2, 6, 10, 50, 100, 500, 1000]
     (2 wątki oznaczają 1 producent + 1 konsument)

   Wartość mierzona:
   - Czas wykonywania
   - Czas CPU

   Wyniki:



Wnioski:

Dla bardzo małej ilości wątków (1 producent i 1 konsument) oraz dla bardzo dużej ilości (powyżej 50) algorytmy działają w porównywalnym czasie. Dla małej ilości wątków problem ze współbieżnością nie istnieje, ponieważ wpuszczamy na przemian jednego klienta i producenta i nie ma walki o zasób. Dla bardzo dużej liczby wątków występuje problem z samą implementacją wątków w Javie i to może być tego kwestia.

Dla mniejszej ilości wątków niż 50 lepiej sprawuje się algorytm z 4 lockami, natomiast dla większej wybrany algorytm nie ma znaczenia.

4. Porównanie czasów dla różnych wielkości bufora:

Ustawienia eksperymentu:

- liczba producentów: 5
- liczba konsumentów: 5
- maksymalna porcja: 10
- generator liczb losowych: Math.random()
- ilość operacji: 5 * 10^5

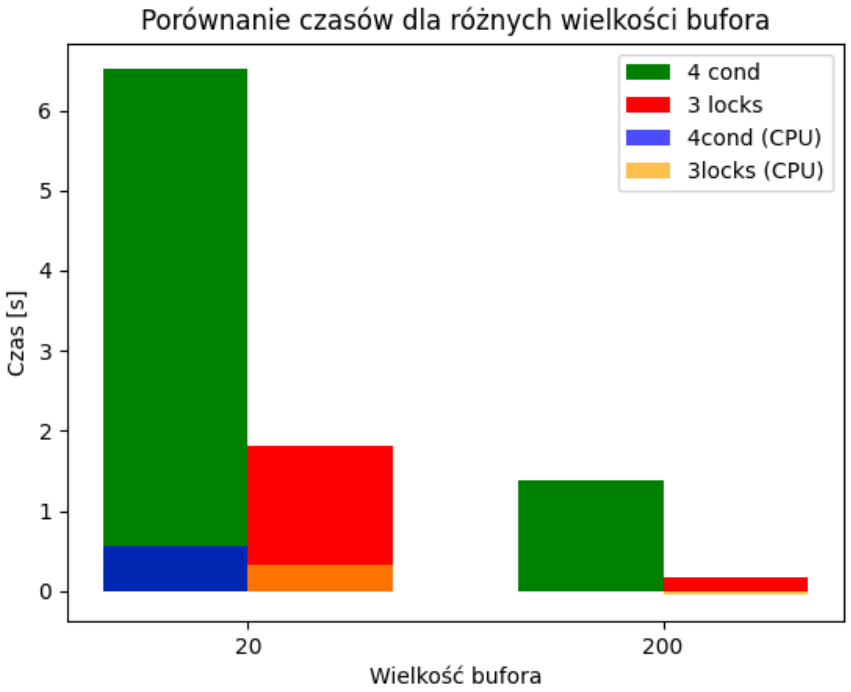
Testowane wartości:

- rozmiar bufora = [20, 200]

Wartość mierzona:

- Czas wykonywania
- Czas CPU

Wyniki:



Wnioski:

Dla znacznie zwiększonego bufora czasy zmalały. Ma to sens, ponieważ nie ma walki o zasób między wątkami, bo bufor jest dużo większy niż porcja. Natomiast mimo to implementacja z 3 lockami działa znacząco szybciej (dla bufora 20 jest 3 razy szybsza, a dla bufora 200 jest 7 razy szybsza).