Sprawozdanie programowanie asynchroniczne **Active Object** vs synchroniczne **four conditions**

Teoria Współbieżności

Adam Ćwikła

2022/2023

Active Object implementacja

Scheduler posiada dwie kolejki **activeQueue** oraz **waitingQueue:**

1. **activeQueue** - nowo utworzone zadanie zawsze tam trafia. Następnie weryfikowane jest czy można je wykonać w przypadku negatywnym dodajemy do **waitingQueue**.
2. **waitingQueue** - zawsze sprawdzamy czy można wykonać zadanie z tej kolejki aby nie zagłodzić wątków.

Synchronizacja w Scheduler zachodzi przy wstawianiu do kolejki **waitingQueue**, ponieważ została zastosowana **LinkedBlockingQueue** synchronizacja wstawiania dzieje się automatycznie za pomocą metody **put**.
Jeżeli w kolejcę **activeQueue** nie ma żadnej pracy do wykonania Scheduler czeka na wstawienie zadań za pomocą metody **take,** która blokuje do momentu otrzymania elementu.

Metryka doświadczenia

Doświadczenie zostało przeprowadzone na poniższym sprzęcie:

1. Procesor AMD Ryzen 7 3700X 3.6 GHz 8 rdzeni 16 wątków
2. System Windows 10
3. Ram 16GB

Dane, które będą stałe w naszym doświadczeniu to:

- ilość wątków producenta/konsumenta równa trzy
- rozmiar bufora równy 10

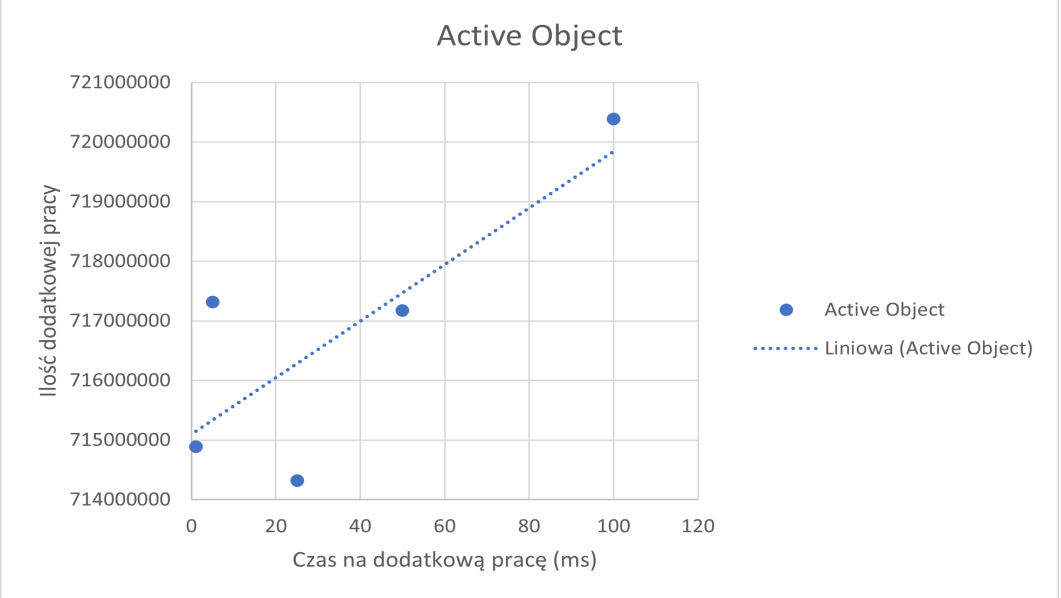
Zmiennymi danymi w naszym doświadczeniu będzie ilość czasu, które przeznaczymy na zadania dodatkowe. W implementacji **Active Object** będą one wykonywały się asynchronicznie, natomiast dla **four conditions** synchronicznie.
Włączamy program na jedną minutę dla każdego czasu i sprawdzamy ile wykonało się dodatkowej pracy.
Ilości czasu zmiennego:

1. 1 / 5 / 20 / 50 / 100 dane podane w ms

Wyniki doświadczenia

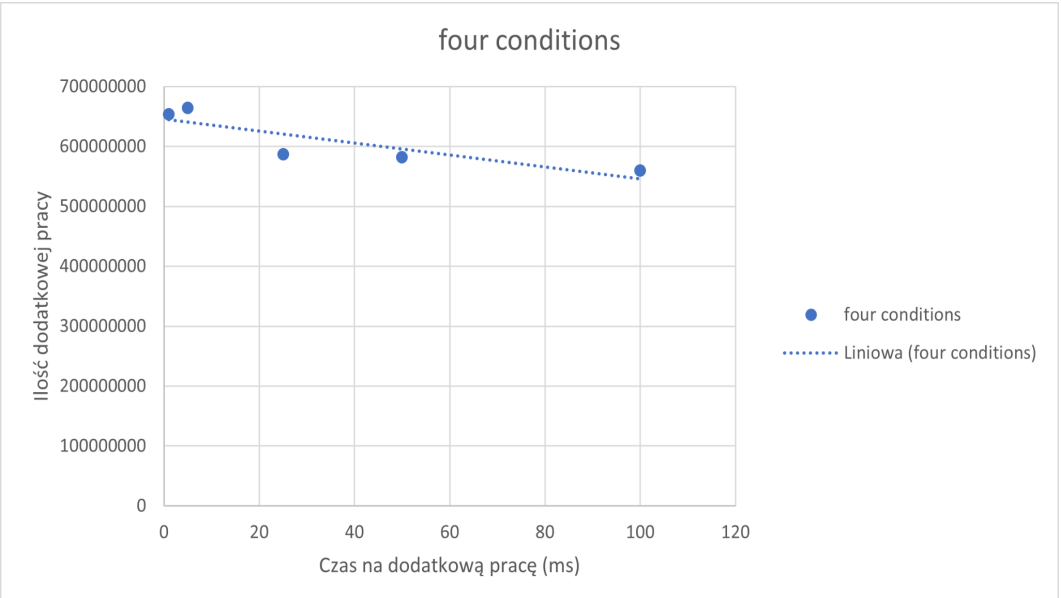
Dla rozwiązania **Active Object**

```
Czas na dodatkowe zadanie 1 ms wykonano dodatkowej pracy: 714884962
Czas na dodatkowe zadanie 5 ms wykonano dodatkowej pracy: 717313389
Czas na dodatkowe zadanie 25 ms wykonano dodatkowej pracy: 714320807
Czas na dodatkowe zadanie 50 ms wykonano dodatkowej pracy: 717171383
Czas na dodatkowe zadanie 100 ms wykonano dodatkowej pracy: 720383801
```



Dla rozwiązania **four conditions**

```
Czas na dodatkowe zadanie 1 ms wykonano dodatkowej pracy: 653765183
Czas na dodatkowe zadanie 5 ms wykonano dodatkowej pracy: 664581415
Czas na dodatkowe zadanie 25 ms wykonano dodatkowej pracy: 586908587
Czas na dodatkowe zadanie 50 ms wykonano dodatkowej pracy: 582049536
Czas na dodatkowe zadanie 100 ms wykonano dodatkowej pracy: 560033324
```



Wnioski

Dla rozwiązania **Active Object** wyniki dodatkowej pracy są znacznie lepsze aniżeli **four conditions**. Dzieje się tak, ponieważ praca w rozwiązaniu **Active Object** wykonuje się asynchronicznie w porównaniu z drugim sposobem. Ilość wykonanej pracy pierwszym sposobem jest w pewnym stopniu stała. Gdy dla **four conditions** im więcej czasu przeznaczyliśmy na wykonanie tej pracy tym mniej jej uzyskaliśmy.

Podsumowanie:
Rozwiązanie **Active Object** jest znacznie lepszę pod względem dodatkowej pracy, którą jesteśmy w stanie wykonać. Jednakże jest ono skomplikowane do zaimplementowania, więc jeśli nie zależy nam na wykonanej pracy w czasie oczekiwania na dostęp do bufora skorzystałbym z rozwiązania **four conditions**.