Implementacja ActiveObject – sprawozdanie

1) Opis implementacji

W ogólności, moja implementacja, w celu zmaksymalizowania wydajności, jest swego rodzaju połączeniem podejścia asynchronicznego z elementami podejścia asynchronicznego. Konkretnie, wątki producentów i konsumentów, oczekując na wykonanie ich Requesta najpierw wykonują ustalonej wielości operację (tj. wyliczanie określoną liczbę razy wartości funkcji tangens; w dalszej części pracy będziemy to działanie określać mianem „operacja oczekiwania asynchronicznego"), po której, jeśli jej zadanie nie zostanie wykonane, przejdzie do oczekiwania synchronicznego.

Najważniejsze punkty implementacji w moim kodzie to:

a) *Task* – klasa będąca połączeniem Request oraz Future; zapewnia Aktywnemu Obiektowi dostęp do metody, którą ma wykonać na zlecenie podanego wątku, a wątkom producentów i konsumentów daje wgląd do wyniku zleconej przez nich operacji;
b) *ActiveShop* – klasa realizuje Scheduler, wybierając i wykonując zadania dane przez wątki producentów i konsumentów; dodatkowo, zawiera też implementacje Proxy, które pozwala tym wątkom na dodawanie zadań do kolei
c) *MySyncQueue* – kolejka priorytetowa, w której przechowywane są zadania; zapewnia też dodatkowo Schedulerowi metodę pozwalającą na oczekiwanie na przyjście zlecenia odpowiedniego typu;

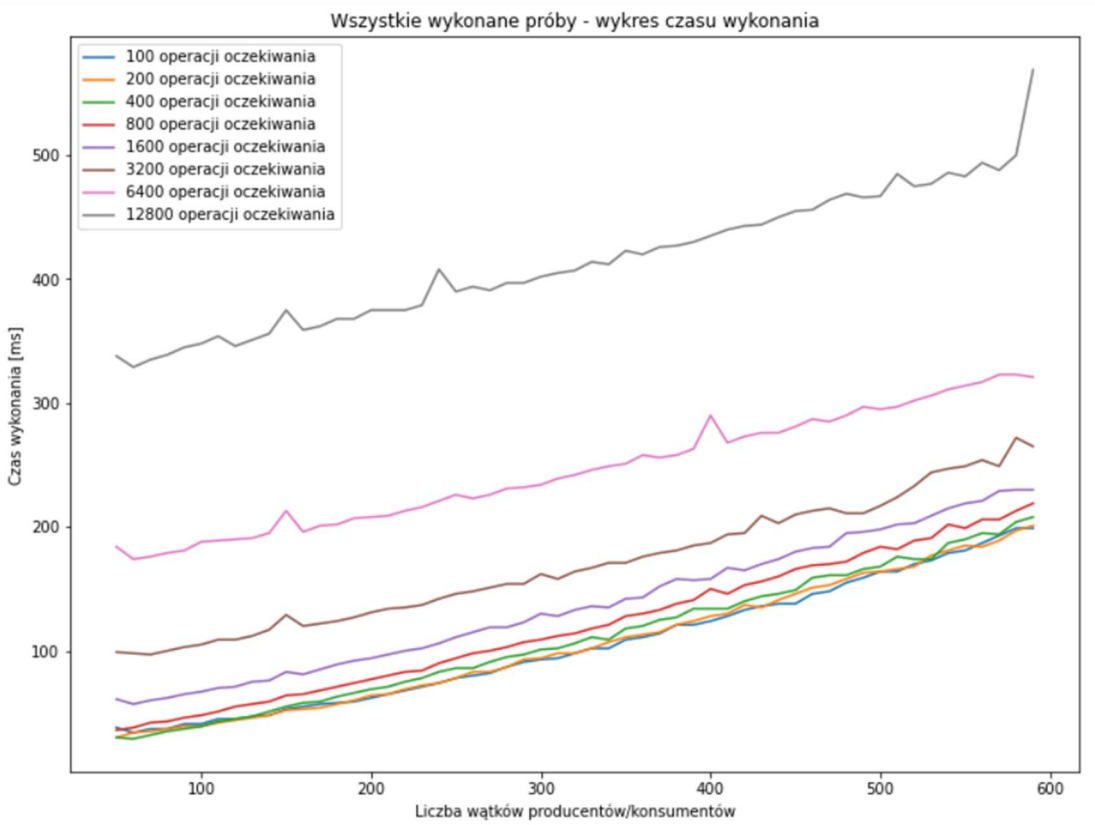
2) Opis eksperymentu

W przeprowadzonym przeze mnie eksperymencie skupię się porównaniu szybkości nowego i starego rozwiązania dla różnej ilości wątków oraz dla różnych długości oczekiwania asynchronicznego (tj. innej ilości operacji.)

3) Wyniki eksperymentu

Uwaga: Dla wszystkich poniższych pomiarów każda wartość była liczona 10 razy, po czym za wartość pomiaru przyjmowałem ich medianę. Zastosowałem to podejście, aby w miarę możliwości zminimalizować wpływ zewnętrznych aplikacji na przebieg eksperymentu.

Zacznijmy od zestawienia ze sobą czasów wykonania dla ośmiu różnych długości oczekiwania asynchronicznego. Dane na poniższym wykresie zestawiłem czas wymagany na wykonanie 5000 tys. operacji produkcji lub konsumpcji w zależności od ilości wątków na buforze 10–cio elementowym dla 8 różnych długości tego oczekiwania.

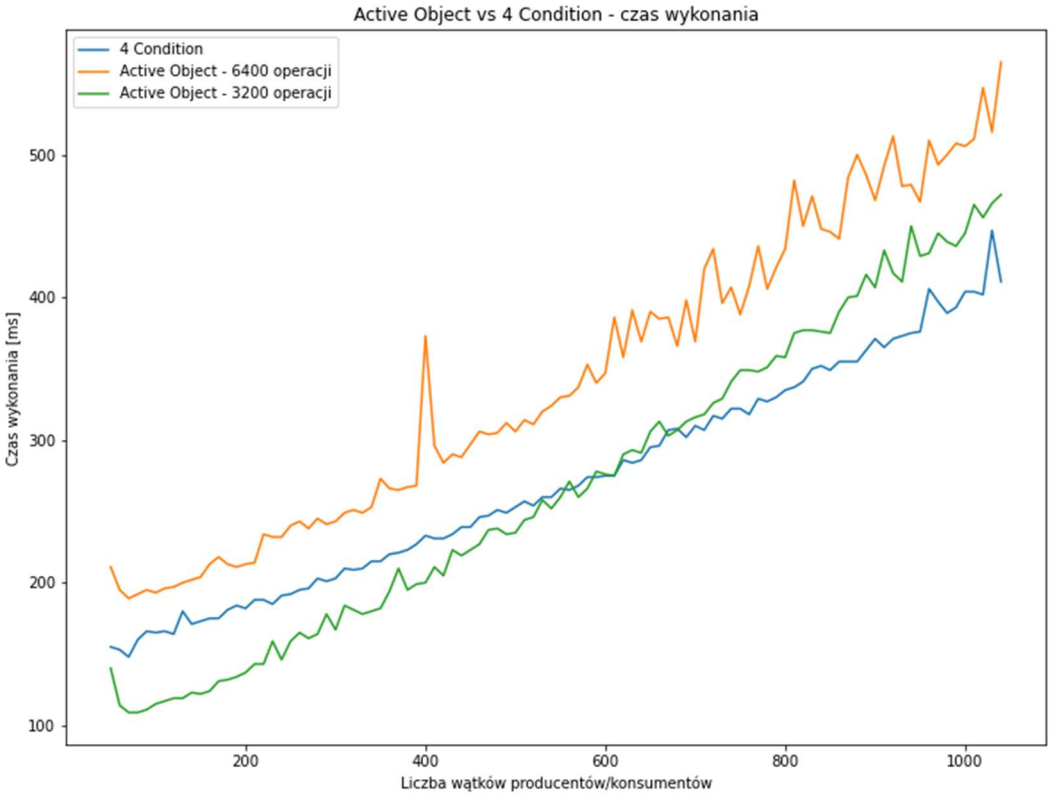


Jak widać na załączonym wykresie, mamy bezpośredni związek między czasem potrzebnym na wykonanie wszystkich operacji, a długością oczekiwania asynchronicznego, tj. im więcej operacji musimy wykonać w oczekiwaniu asynchronicznym, tym dłużej będziemy czekać na ostateczny wynik. Dodatkowo, widzimy, że dla wszystkich wartości oczekiwania, ilość wątków producentów/konsumentów wpłynie na czas potrzebny na wykonanie wszystkich operacji w podobny sposób. Możemy się utwierdzić w tym przekonaniu, patrząc na współczynniki regresji liniowej dla podanych danych:

Liczba operacji oczekiwania	Wartość współczynnika regresji
100	0.310656565656565
200	0.318253968253968
400	0.329191919191919
800	0.338564213564213

1600	0.332525252525252
3200	0.308340548340548
6400	0.282640692640692
12800	0.326847041847042

Porównajmy teraz czas potrzebny na wykonanie przykładowych implementacji ActiveObject z czasem wykonania rozwiązania opartego na 4 zmiennych Condition. Ponownie, dane na poniższym wykresie zestawiłem czas wymagany na wykonanie 5000 tys. operacji produkcji lub konsumpcji w zależności od ilości wątków na buforze 10–cio elementowym dla 8 różnych długości tego oczekiwania.



Jak widzimy, na otrzymanym wykresie, dla pewnej długości oczekiwania asynchronicznego (6400 operacji oczekiwania), czas wykonania wszystkich zadanych operacji jest od początku większy niż czas wymagany przez 4 Condition. Dodatkowo, dla długości oczekiwania, która początkowo daje lepsze wyniki niż 4 Condition (3200 operacji oczekiwania), po przekroczeniu pewnej ilości równolegle pracujących wątków staje się ona wolniejsza. Co więcej, patrząc na współczynnik regresji dla danych z rozwiązania 4 Condition:

$$a = 0.26117671767176714$$

możemy się spodziewać, że dla każdej z zaprezentowanych wartości oczekiwania asynchronicznego będzie hipotetycznie istniała ilość wątków, po której rozwiązanie to będzie wymagało więcej czasu na wykonanie, lecz w praktyce, zwłaszcza dla małych ilości operacji oczekiwania, będą to wartości na tyle duże, że rzadko trafimy na ten problem.

4) Wyniki eksperymentu

a) Dla mniejszych ilości operacji oczekiwania asynchronicznego oraz odpowiednio do nich dobranej ilości wątków producenta i konsumenta, implementacja problemu z zastosowaniem wzorca ActiveObject jest znacznie szybsza niż rozwiązanie oparte na 4 zmiennych Condition.

b) Fakt, że od pewnego progu ilości operacji oczekiwania asynchronicznego rozwiązanie wykorzystujące ActiveObject jest wolniejsze niż 4 Condition wynika najprawdopodobniej bezpośrednio z fizycznej ilości operacji, którą wątki muszą wykonać, zanim sprawdzą czy został wykonany ich Request, co powoduje sytuacje, w której Scheduler nie wykonuje żadnej operacji, bo wszystkie wątki, które mogłyby dać mu operację, którą mógłby wykonać są zajęte;

c) Po przekroczeniu pewnej ilości wątków rozwiązanie ActiveObject przestaje być szybsze niż rozwiązanie 4 Condition. Prawdopodobnie wywoływane jest przez ograniczone zasoby procesora, a co za tym idzie wydłużony czas oczekiwania asynchronicznego dla wątków, jeśli wiele z nich próbuje to robić w tym samym czasie.

d) Na podstawie podanych przykładów widzimy, że implementacja ActiveObject w ogólności będzie przydatna, jeśli w problemie, który będziemy rozwiązywać wątki oprócz wykonywania pracy na pewnym wspólnym buforze będą musiały wykonywać dodatkowe niekolizyjne operacje. Jak widać w załączonych przykładach, oddelegowanie tej pracy do osobnego wątku daje nam czas, w którym możemy wykonywać własne operacje, co podniesie efektywność całego rozwiązania.