Porównanie wydajnościowe prob. Prod. I Kons. Active Object vs. 3 Locki

Mateusz Mazur

1. Cel ćwiczenia.

Celem ćwiczenia było porównanie wydajnościowe rozwiązania problemu Producent/Konsument za pomocą Active Object oraz 3 Locków.

2. Specyfikacja sprzętowa.

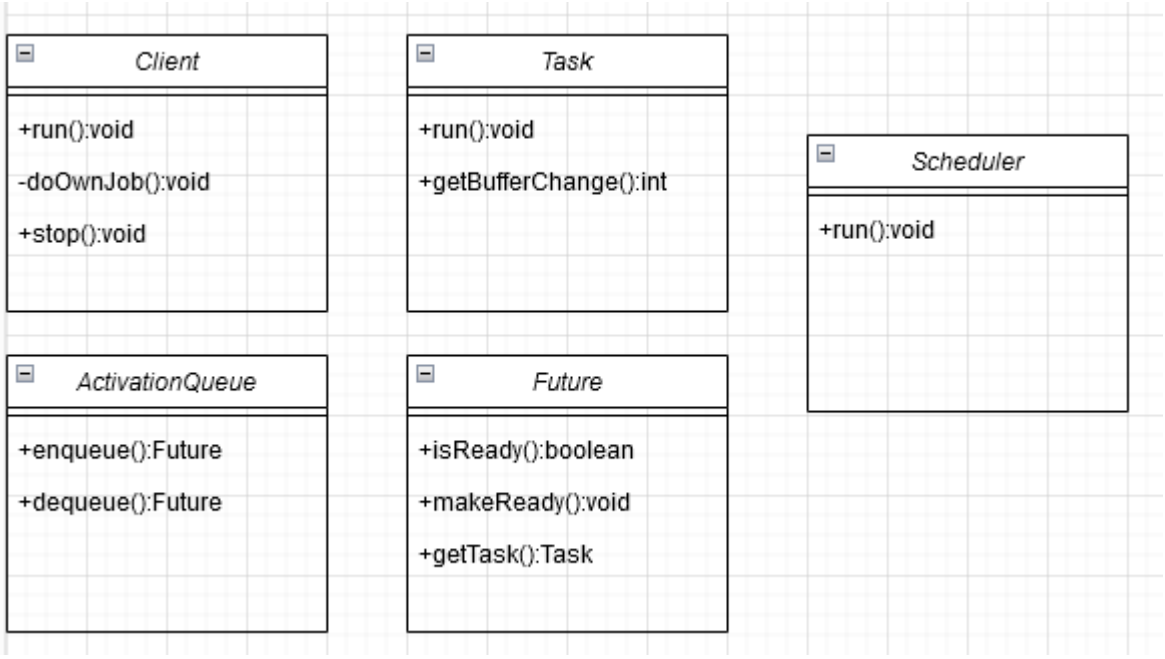
Procesor: Ryzen 1700 – 8 rdzeni, 16 procesorów logicznych, taktowanie 3GHz
Pamięć RAM: 16GB

3. Sposób przeprowadzania eksperymentu

Dla każdego ze sposobów wykonałem pomiary dla kolejno 2,4,6,10,16,100,2000 wątków klientów przez okres 5 minut. Po tym czasie zatrzymywałem wątki i odczytywałem liczbę wykonanych operacji na buforze oraz średni czas dostępu do procesora. Dla Active Object mierzyłem również ilość pracy wykonanej przez wątki w czasie oczekiwania na odpowiedź.

Następnym krokiem było sprawdzenie zachowania się programów pod różnym nakładem pracy (liczenie sinusów) oraz porównanie ich działania przy dużo większym buforze w porównaniu do tworzonych/pobieranych porcji. W tym wypadku testowałem programy przez 2 minuty każdy.

4. Implementacja wzorca Active Object

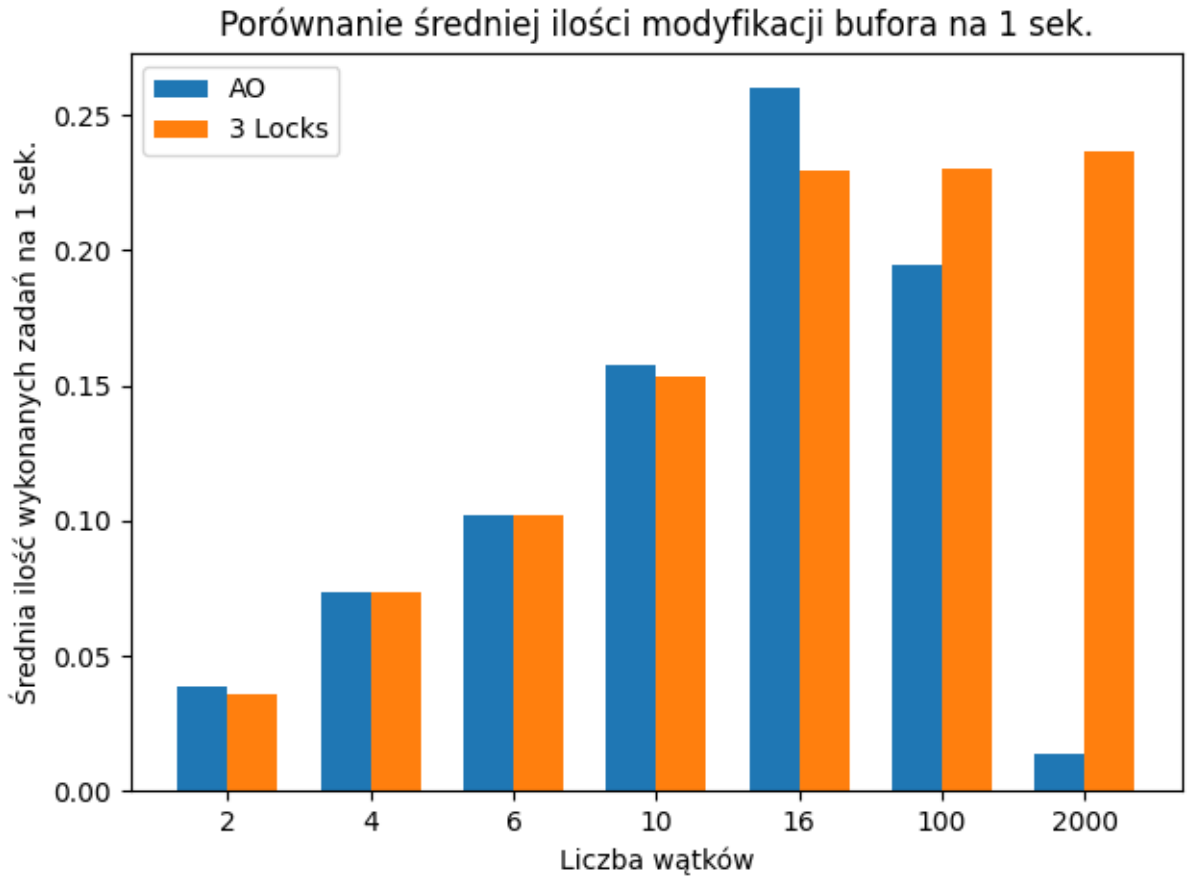


Rysunek 1. Zaimplementowane klasy.

Implementacja obejmuje 5 klas (rys. 1):

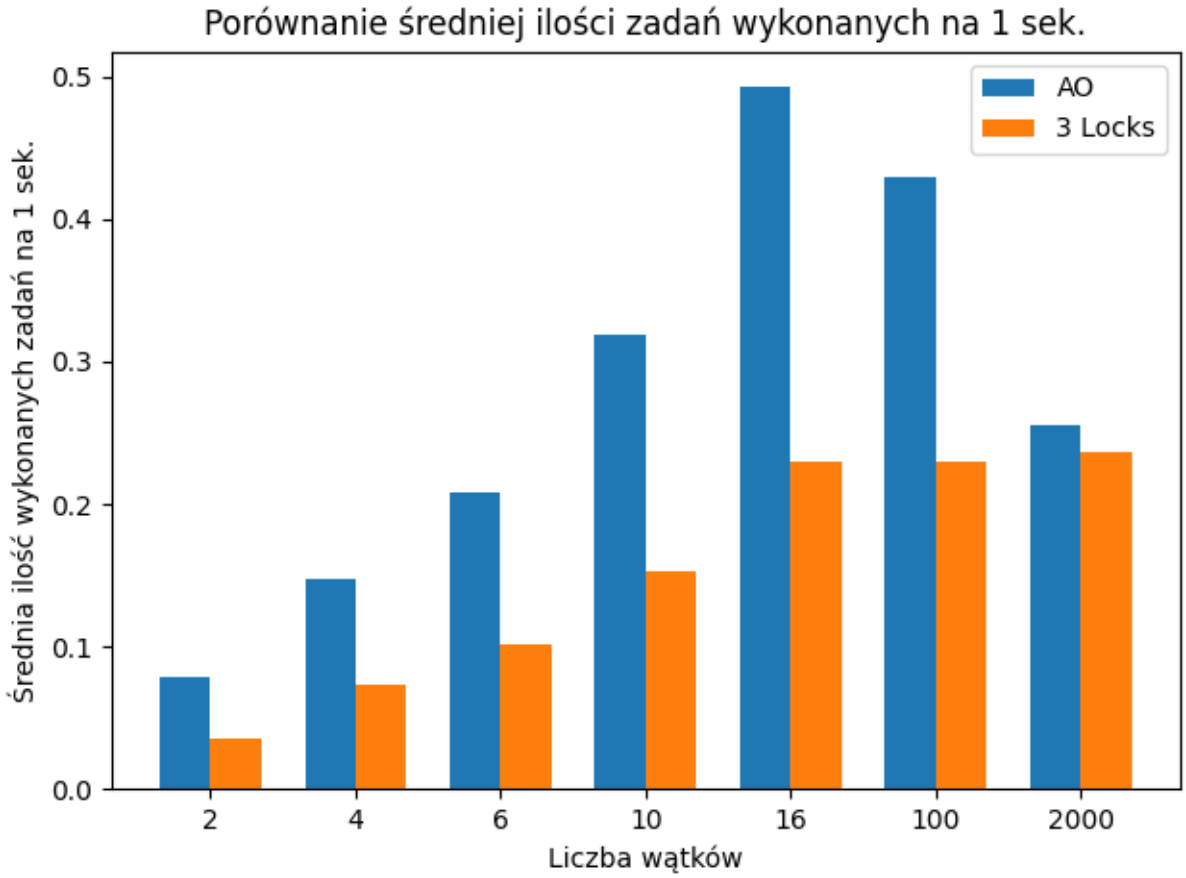
1. **Client** – klasa klientów, wykonuje w pętli tworzenie instancji zadań **Task** i odkłada je do kolejki w **Activation Queue**. W przeciwnym wypadku wykonuje własne zadania i czeka na zakończenie wykonywania Taska poprzez **Future.**
2. **Task** – klasa przechowująca zadanie do wykonania
3. **Future** – przechowuje **Task** i zwraca informacje o jego wykonaniu
4. **ActivationQueue** – przechowuje kolejkę zadań do wykonania przez **Schedulera** i zapewnia jej synchronizację poprzez zastosowanie locka oraz condition.
5. **Scheduler** – pobiera instancję Future z **ActivationQueue** i wykonuje zadania na buforze. Zapewnia brak zagładzania poprzez użycie dodatkowej kolejki priorytetowej.

5. Wyniki



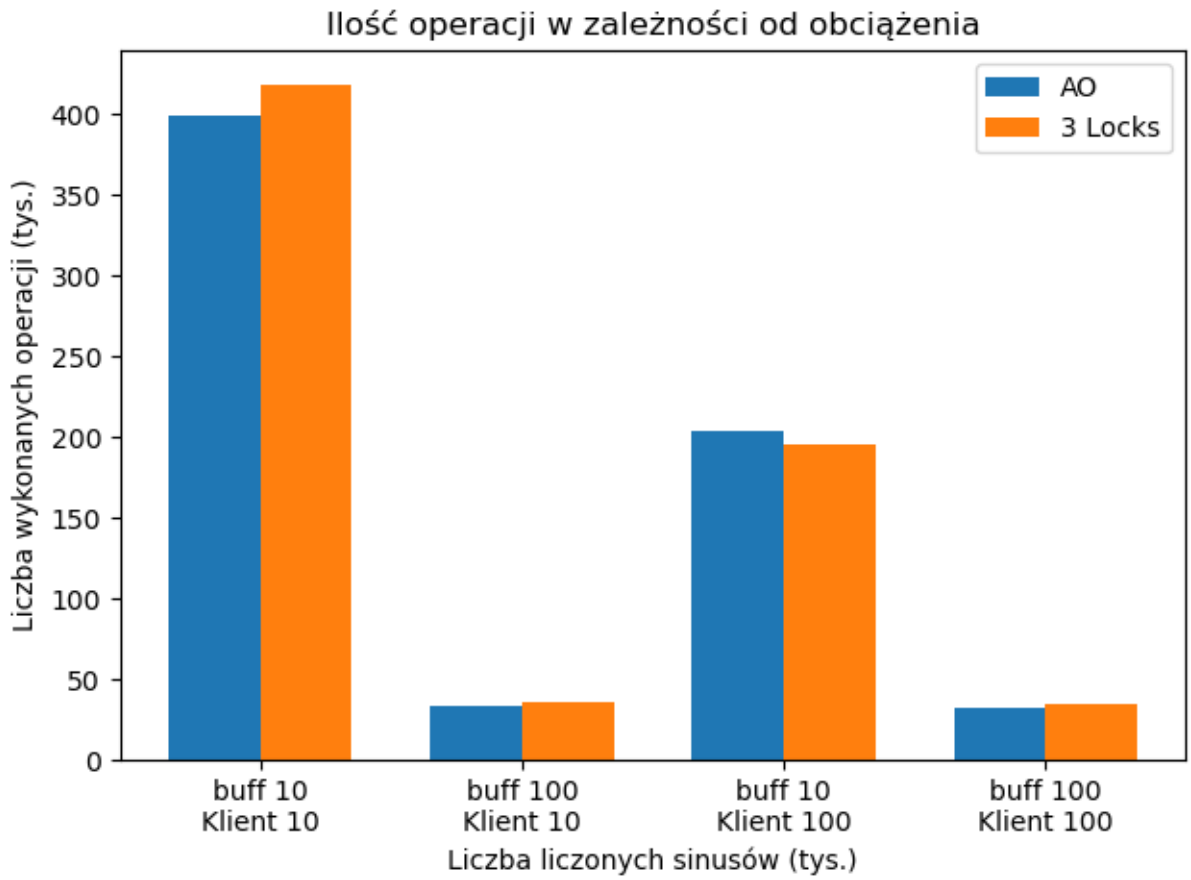
Rysunek 2. Porównanie średniej ilości zadań na 1 sekundę dla różnych ilości wątków

Otrzymane wyniki wskazują na niewielką przewagę rozwiązania na **Active Object** dla ilości wątków nie przekraczającej ilość procesorów logicznych (w moim wypadku 16). Dla większej ilości wątków postępuje zdecydowane pogorszenie się tej implementacji w stosunku do rozwiązania na 3 Lockach, które utrzymuje się na stale wysokim poziomie. Może to być spowodowane tym, że przy wzrastającej ilości wątków klientów, Scheduler coraz rzadziej dostaje dostęp do procesora.



Rysunek 3. Porównanie średniej ilości wszystkich wykonanych zadań na 1 sek.

W czasie oczekiwania na wykonanie zadania przez Scheduler wątek klienta wykonuje własne zadania. Wykres na rys.3 przedstawia średnią wszystkich zadań wykonanych przez wątki. Jest to o tyle istotna zależność, że dla rozwiązania na 3 Lockach wątki bezczynnie oczekują na dostęp do bufora, gdy klienci w AO w czasie oczekiwania mogą wykonywać inne zadania.



Rysunek 4. Porównanie ilości operacji w zależności od obciążenia.

Test przeprowadziłem na 6 wątkach klientów wraz ze zwiększoną pojemnością buforu, by ta nie wpływała znacząco na wyniki. Podczas obciążania zarówno buforów (w AO wątek Schedulera i Klienta osobno wykonują dodatkową pracę, w 3 Lockach wątek Klienta wykonuje swoją pracę oraz „pracę buforu”). Przy równomiernym obciążeniu obu stron możemy zauważyć przewagę rozwiązania na 3 lockach, która w ogólnym przypadku może być szybsza ponieważ wykonuje mniej postronnych operacji (jest prostsza w budowie). Gdy odpowiednio obciążymy wątek klienta lepszy wynik daje nam natomiast AO, który operacje na buforze wykonuje (już po zleceniu) niezależnie od wątku klienta.

6. Wnioski

Oba rozwiązania mają swoje „za i przeciw”. W moim wypadku AO sprawiał się lepiej dla ilości wątków nie przekraczającej ilość procesorów logicznych, ale dla dużo większej ilości klientów jego jakość drastycznie spada z powodu niewielkiego dostępu do zasobów dla wątku Schedulera, dlatego rozw. na 3 Lockach sprawdza się wtedy lepiej.

Dla tej samej liczby wątków, a pod różnym obciążeniem w 3 z 4 przypadków lepiej sprawdziło się rozwiązanie na 3 Lockach, dzięki małej ilości dodatkowego kodu. W przypadku, gdy Klient jest bardziej obciążony AO wykonuje więcej operacji na buforze, więc w takim wypadku lepiej jest zastosować ten wzorzec.

Przewagą Active Object jest również możliwość wykonywania przez klientów innych zadań w czasie wykonywania zadań przez Scheduler.