Teoria Współbieżności

Sprawozdanie I

Grupa wtorek 16:40

Michał Nożkiewicz

7 listopada 2023

1 Opis zadania i użyte narzędzia

Zadaniem było zaimplementowanie sześciu rozwiązań do problemu pięciu filozofów. Rozwiązania zostały napisane w Javie. Implementacja znajduje się w folderze src/main/java/org/task. Pomiary czasowe zapisałem do plików csv, a analizę danych przeprowadziłem w pythonie.

2 Opis rozwiązań

Każde rozwiązanie jest zaimplementowanie w oddzielnej klasie o nazwie Soli, gdzie i to numer zadania. Każda klasa dziedziczy z abstrakcyjnej klasy Philosopher i implementuje metody takeForks() i putForks(). Obiekty Soli w trakcie działania stanowią odrębne wątki. Ich najważniejsze atrybuty to "leftFork"(widelec od lewej strony filozofa), "rightFork"(prawy widelec), a także "semaphore", czyli obiekt typu Semafor, który przydaje się w trzech rozwiązaniach.

2.1 Zadania 1, 3 i 4

Te zadania mają bardzo podobne rozwiązania. Każdy widelec posiada atrybut "lock"typu ReentrantLock. Każdy filozof próbuje zdobyć odpowiedni widelec, a jeśli obiekt jest już w posiadaniu innego wątku to czeka na jego zwolnienie.

2.2 Zadanie 5

W tym zadaniu dodatkowo używany jest semafor licznikowy. Jeśli mamy N filozofów to jego początkowa wartość to N-1, dzięki czemu maksymalnie N-1 walczy o widelce przez co można uniknąć potenjcalnego zakleszczenia jak w zadaniu 1.

2.3 Zadanie 6

Tutaj także skorzystałem z semafora. Jego początkowa wartość wynosiła N-1, ale tym razem filozof, który przybył jako N-ty nie rezygnuje z walki o widelce, ale zamiast tego podnosi je w odwrotnej w kolejności od wszystkich pozostałych.

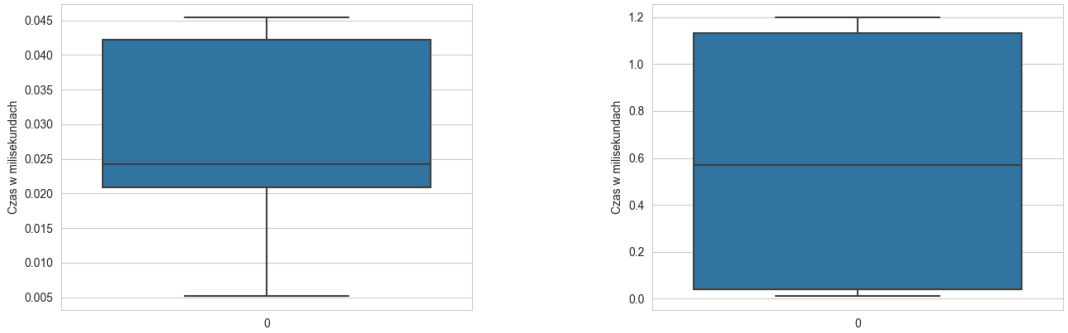
2.4 Zadanie 2

W tym zadaniu użyłem semafora binarnego, który pozwalał tylko jednemu filozofowi naraz sprawdzić czy widelce są zajęte. Jeśli chociaż jeden był zajęty to rezygnował z podnoszenia któregokolwiek i czekał na kolejną próbę.

3 Pomiary czasowe

Celem pomiarów było sprawdzanie ile w danym z rozwiązań każdy z filozofów średnio czeka na to, aby mógł zjeść. Pomiary wykonywałem w dwóch wariantach. W wariancie w którym naraz działało 5 wątków i w wariancie z 20 wątkami. Założyłem, że każda czynność filozofa(to jest myślenie i jedzenie) musi trwać jakiś ustalony okres czasu. Gdy filozofów było pięciu, było to 10 milisekund, ale wykonywali wtedy 1000 cykli myślenie-jedzenie, a w przypadku z 20 filozofami było to 100 milisekund i wykonywanych było 100 iteracji.

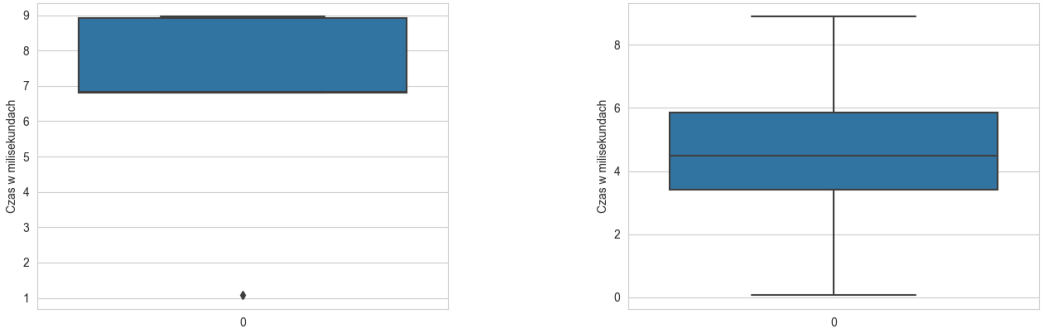
W każdym przypadku zmierzyłem średni czas oczekiwania na widelce i wyniki przedstawiłem na wykresie pudełkowym.



(a) 5 wątków

(b) 20 wątków

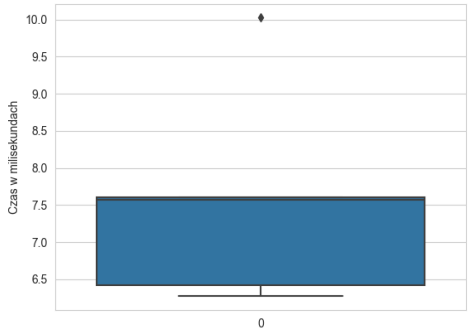
Rysunek 1: Rozwiązanie nr 2



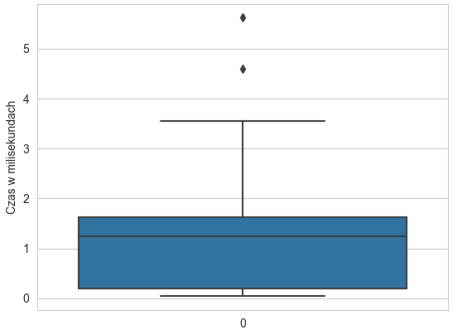
(a) 5 wątków

(b) 20 wątków

Rysunek 2: Rozwiązanie nr 3

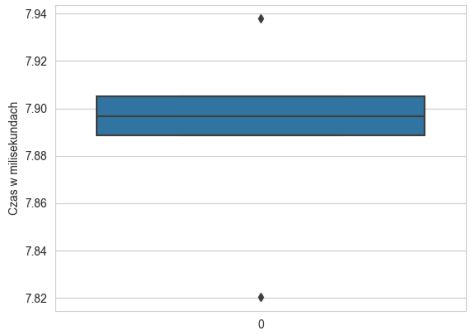


(a) 5 wątków

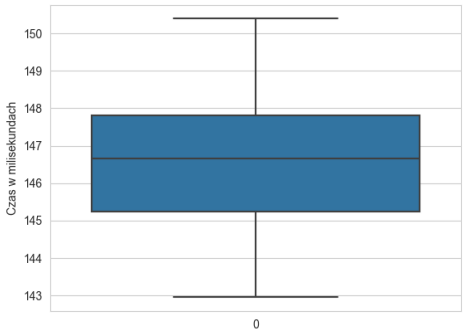


(b) 20 wątków

Rysunek 3: Rozwiązanie nr 4

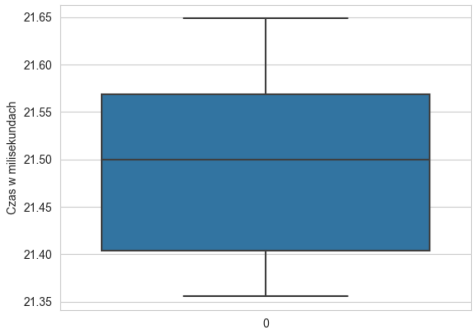


(a) 5 wątków

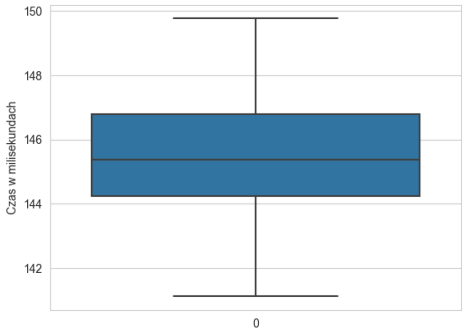


(b) 20 wątków

Rysunek 4: Rozwiązanie nr 5



(a) 5 wątków



(b) 20 wątków

Rysunek 5: Rozwiązanie nr 6

4 Wnioski

Jak widać na przedstawionych wykresach najszybszy i znacznie szybszy okazał się sposób drugi który tak naprawdę wykorzystywał tylko jeden obiekt do synchronizacji wszystkich wątków. Najwolniejsze okazały się sposoby 5 i 6. Były kilka lub kilkanaście razy wolniejsze od innych. Wyniki to zapewne z dużej ilości mechanizmów użytych do synchronizacji, ale były one także najbardziej sprawiedliwe. Różnice czasowe między poszczególnemu wątkami są najmniejsze podczas, gdy w sposobie 2 są one bardzo duże.