TW-Filozofowie

Maksymilian Zawiślak

October 2023

1 Wstęp

Problem pięciu filozofów jest jednym z klasycznych problemów teorii współbieżności. Podstawowe sformułowanie problemu jest następujące :

- N filozofów zasiada przy okrągłym stole
- Pomiędzy sąsiednimi filozofami leży widelec (łącznie jest N widelców)
- Każdy filozof działa ciągle według schematu „myślenie - jedzenie - myślenie - jedzenie - ...". Każdy z etapów (myślenie i jedzenie) jest skończony.
- Aby zjeść, filozof musi podnieść oba sąsiadujące widelce

Do rozwiązywania problemu został użyty język programowania Java, wykorzystaniem mechanizmów takich jak: ReentrantLock, Synchronized oraz Semaphore. Aby zapobiec sytuacji, w której jeden z filozofów ciągle podnosi swoje sąsiadujące widelce po odłożeniu ich, filozof jest uśpiony na okres 10 milisekund przed próbą ponownego podniesienia widelców. Symulacja działa przez 5 sekund, potem wszystkie wątki są przerywane. Warianty rozwiązania będą testowane dla N = 5 oraz N = 20. Wyniki pomiarów czasów każdego z filozofów są przedstawione na wykresach pudełkowych w nanosekundach. Oś y jest prezentowana w skali logarytmicznej dla lepszej reprezentacji wyników.

2 Rozwiązania problemu

2.1 Wariant 1 - rozwiązanie naiwne (z możliwością blokady)

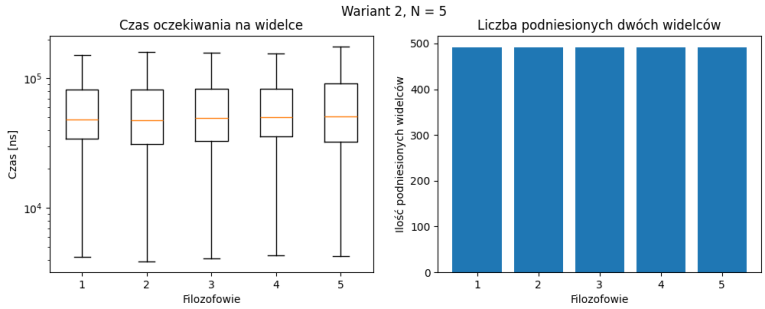
Każdy filozof czeka, aż wolny będzie lewy widelec, a następnie go podnosi (zajmuje), następnie podobnie postępuje z prawym widelcem.

To podejście prowadzi niemal natychmiast do zakleszczenia, niezależnie od liczby filozofów (czy to N = 5, czy N = 20). Każdy filozof podnosi lewy widelec i nie jest w stanie podnieść prawego, co powoduje zakleszczenie w procesie. Przy implementacji został wykorzystany mechanizm Synchronized.

2.2 Wariant 2 - rozwiązanie z możliwością zagłodzenia

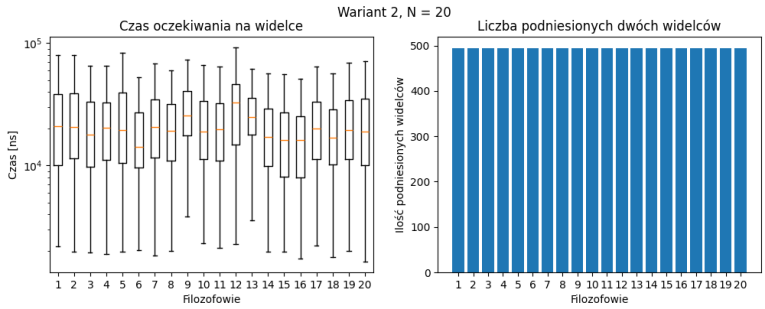
Każdy filozof sprawdza czy oba sąsiednie widelce są wolne i dopiero wtedy zajmuje je jednocześnie. Rozwiązanie to jest wolne od blokady, jednak w przypadku, gdy zawsze któryś z sąsiadów będzie zajęty jedzeniem, nastąpi zagłodzenie, gdyż oba widelce nigdy nie będą wolne.

2.2.1 N = 5



Rysunek 1

2.2.2 N = 20



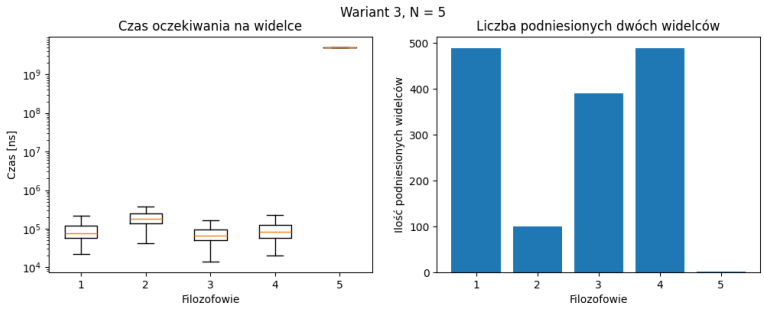
Rysunek 2

Dla testowanych N oraz kilku prób przeprowadzenia symulacji, dzięki wykorzystaniu uśpienia filozofa na 10 milisekund po jedzeniu, nie udało się zaobserwować zagłodzenia żadnego z filozofów. Przy implementacji został wykorzystany mechanizm ReentrantLock.

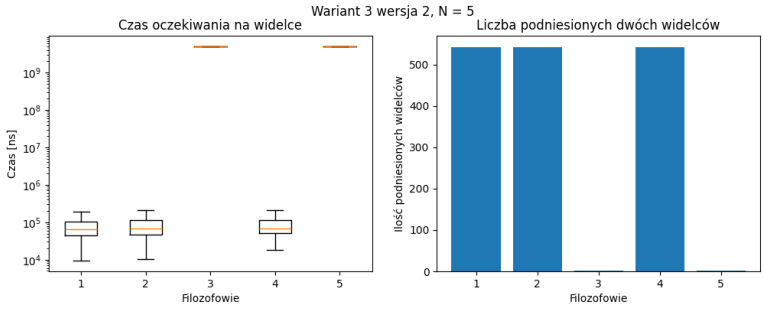
2.3 Wariant 3 - rozwiązanie asymetryczne

Filozofowie są ponumerowani. Filozof z parzystym numerem najpierw podnosi prawy widelec, filozof z nieparzystym numerem najpierw podnosi lewy widelec.

2.3.1 N = 5

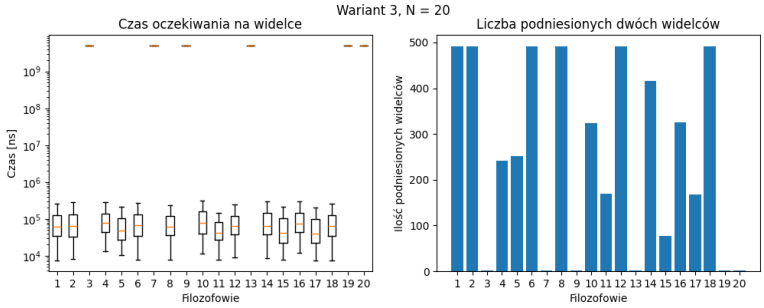


Rysunek 3

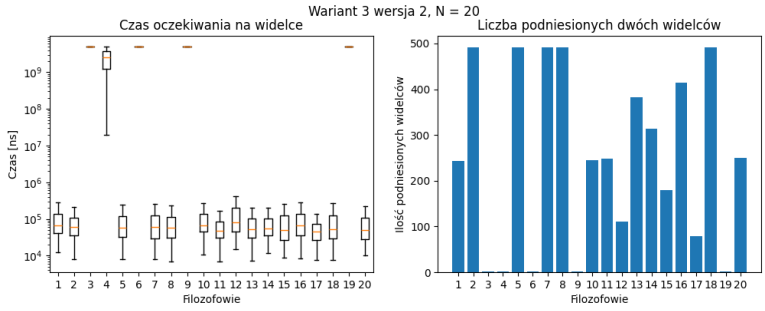


Rysunek 4

2.3.2 N = 20



Rysunek 5



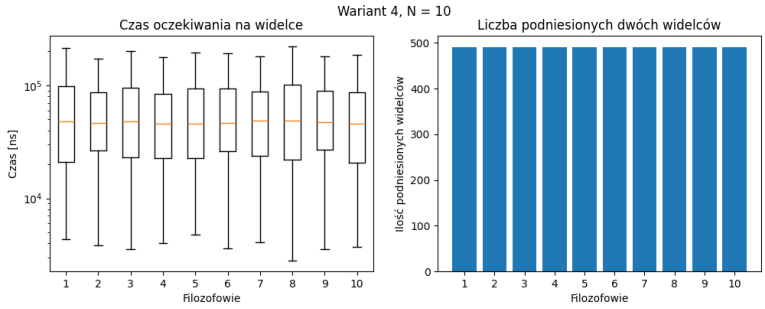
Rysunek 6

Na wykresach dobrze widać że w tym przypadku dochodzi do zagłodzenia, niektórych z filozofów. Przy pierwszym przeprowadzaniu symulacji dla N = 5, następuje zagłodzenie filozofa numer 5. Przy drugiej próbie zagłodzony jest 3 oraz 5 filozof. Analogiczną sytuacje można zaobserwować dla N = 20. Przy implementacji został wykorzystany mechanizm Synchronized.

2.4 Wariant 4 - rozwiązanie stochastyczne

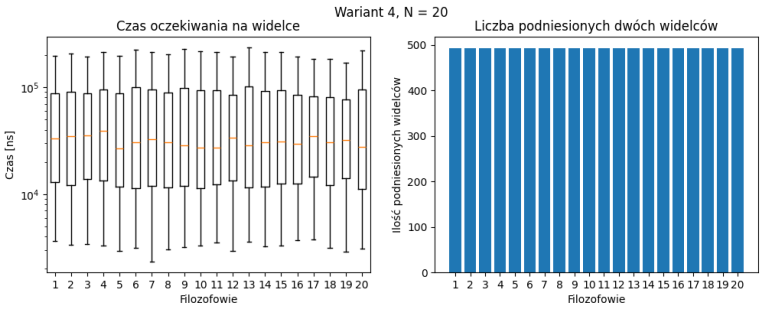
Każdy filozof rzuca monetą tuż przed podniesieniem widelców i w ten sposób decyduje, który najpierw podnieść - lewy czy prawy.

2.4.1 N = 10



Rysunek 7

2.4.2 N = 20



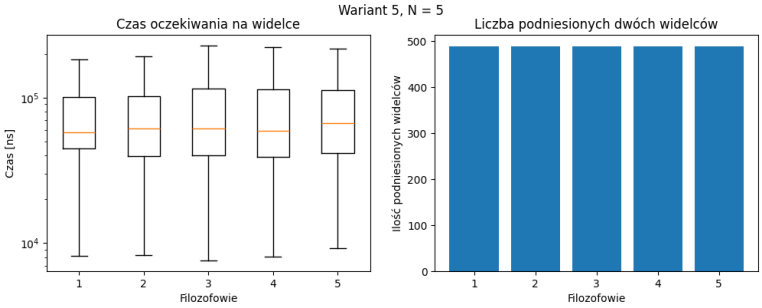
Rysunek 8

W przypadku N = 5 notorycznie dochodziło do zakleszczenia co nie pozwalało wygenerować wykresów. Dopiero wykorzystanie N = 10 nie powodowało takiego problemu. Dla na $N \geqslant 10$ ten wariant działał poprawnie. Niższe N często kończyło się zakleszczeniem w czasie trwania symulacji. Przy implementacji został wykorzystany mechanizm Synchronized oraz Random.

2.5 Wariant 5 - rozwiązanie z arbitrem

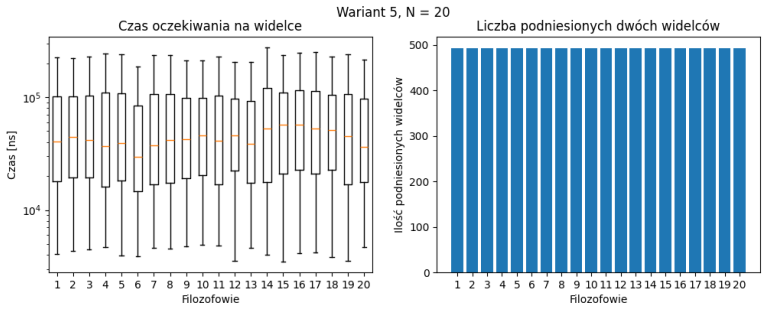
Zewnętrzny arbiter (lokaj, kelner) pilnuje, aby jednocześnie co najwyżej czterech (w ogólnym przypadku N-1) filozofów konkurowało o widelce. Każdy podnosi najpierw lewy a potem prawy widelec. Jeśli naraz wszyscy filozofowie będą chcieli jeść, arbiter powstrzymuje jednego z nich aż do czasu, gdy któryś z filozofów skończy jeść.

2.5.1 N = 5



Rysunek 9

2.5.2 N = 20



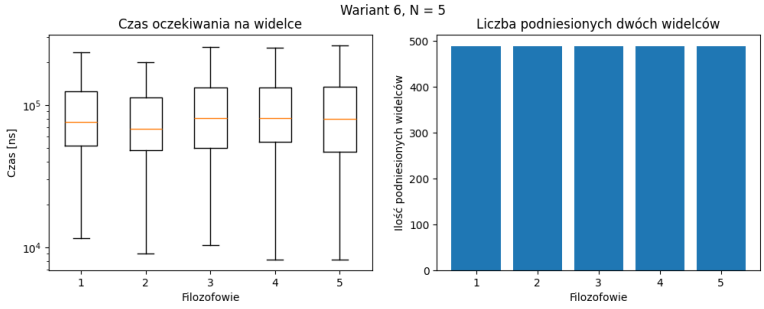
Rysunek 10

Wariant z lokajem działa poprawnie. Żaden filozof nie jest zagłodzony oraz nie pojawia się zakleszczenie. Przy implementacji został wykorzystany mechanizm Synchronized oraz Semaphore.

2.6 Wariant 6 - rozwiązanie z jadalnią

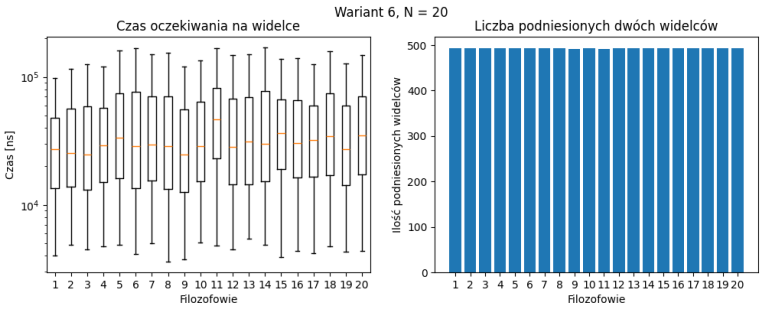
Rozwiązanie jest modyfikacją wersji z arbitrem. Filozof, który nie zmieści się w jadalni (czyli arbiter nie pozwolił mu jeść) je "na korytarzu" podnosząc jednorazowo widelce w odwrotnej kolejności (do reszty filozofów w jadalni).

2.6.1 N = 5



Rysunek 11

2.6.2 N = 20



Rysunek 12

Ten wariant to ulepszona wersja wcześniejszego wariantu. Nie zmusza jednego z filozofów do czekania aż zwolni się dla niego miejsce w semaforze, co pozwala o stałe konkurowanie N filozofów o widelce. Również działa poprawnie. Przy implementacji został wykorzystany mechanizm Synchronized oraz Semaphore.

3 Wnioski

Średni czas oczekiwania, w większości przydatków, wahał się między 10^4 a 10^5 nanosekund, mimo zastosowania różnych podejść rozwiązania problemu oraz innych mechanizmów synchronizacji. Przy takich wartościach ciężko określić, czy któryś z wariantów działa szybciej. Po wykonanych testach warianty 2, 5 oraz 6 okazały się działać poprawnie w sprawdzanych przypadkach. Wariant 1 oczywiście nie jest dobrym rozwiązaniem problemu, natychmiast kończy się zakleszczeniem symulacji. Rozwiązanie asymetryczne prowadzi do zagłodzenia filozofów. Natomiast przy podejściu stochastycznym należy pamiętać, aby używana była odpowiednio wysoka liczba filozofów.