Rozproszony Bufor dla problemu Producentów i Konsumentów

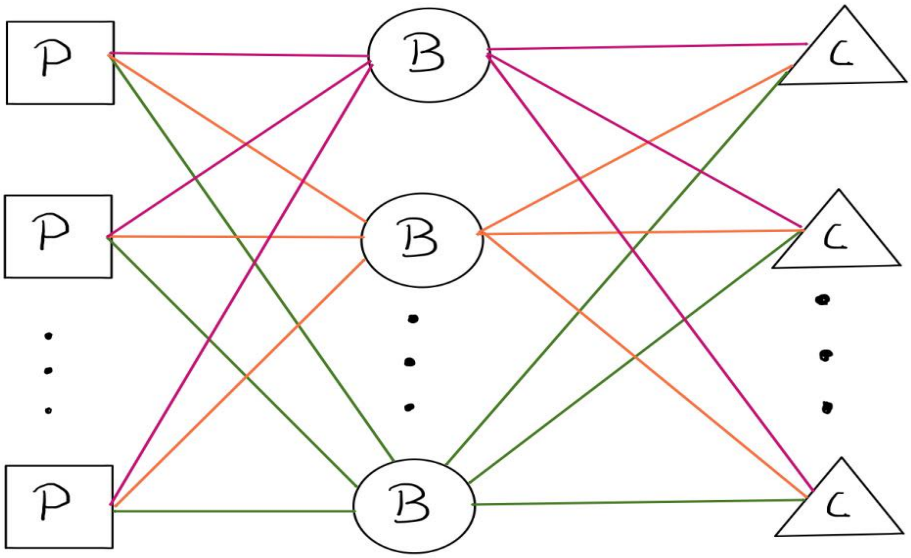
Szymon Twardosz

Grudzień 2023

1 Wprowadzenie

W ramach ćwiczenia zaimplementowany został rozproszony bufor oraz klasy producentów i konsumentów. Całość zadania zrealizowana została w języku Java oraz biblioteki JCSP. Wszelkie pomiary wykonane zostały na laptopie z systemem operacyjnym Windows 10 x64, o procesorze AMD Ryzen 7 3750 2.30GHz, z zainstalowaną pamięcią RAM 8,00GB.

2 Architektura rozwiązania



Rysunek 1: Architektura rozwiązania

3 Idea działania Bufora

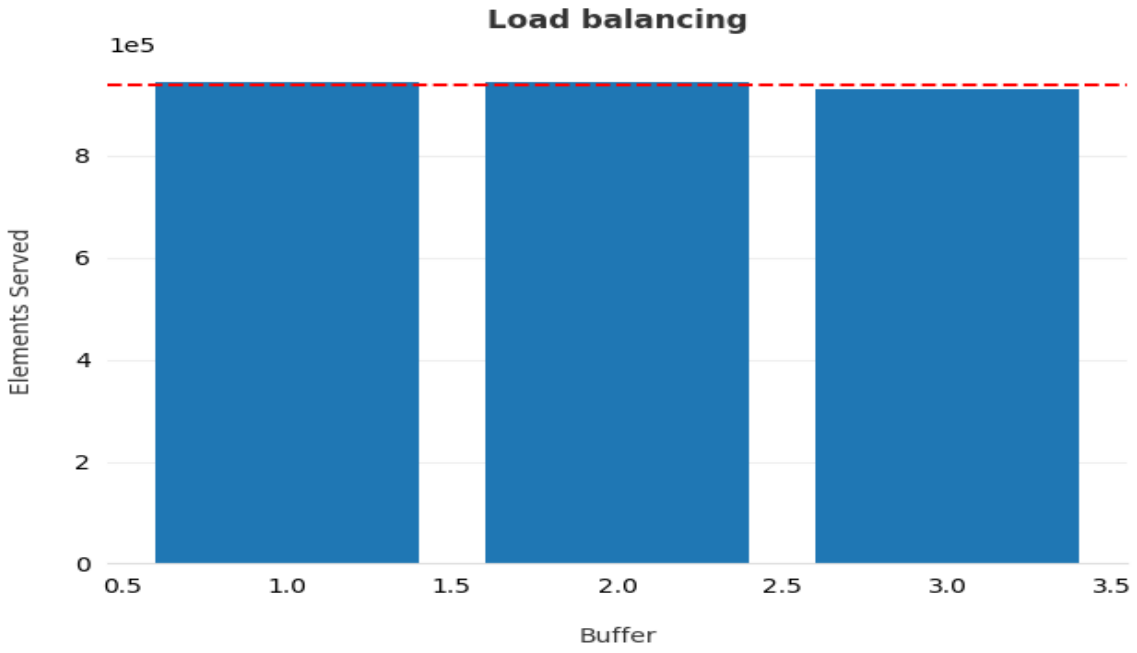
Zaimplementowany Bufor posiada dwa stany, pomiędzy którymi się przełącza w zależności od aktualnej liczby elementów:

- SELL_ONLY - obsługuje tylko Konsumentów
- BUY_ONLY - obsługuje tylko Producentów

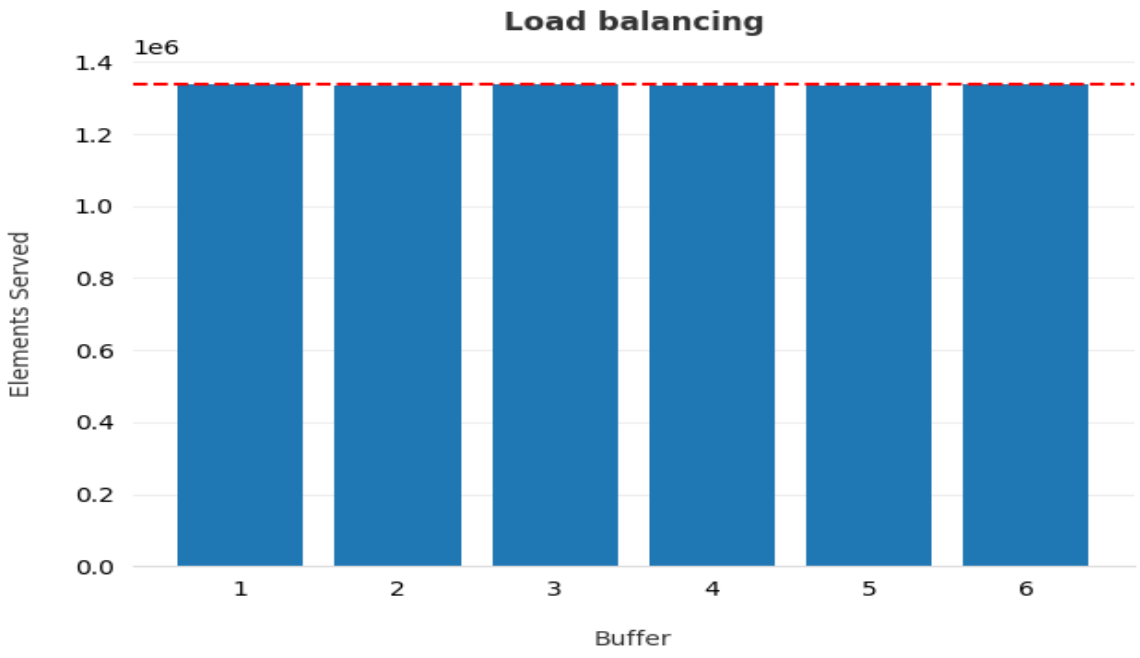
Konsumenci i producenci przed wysłaniem/odebraniem porcji od Bufora, dokonują u niego rezerwacji. Gdy już jej dokonają (i przebiegnie ona pozytywnie) mają pewność, że mogę wysłać/odebrać swoją porcję elementów.

4 Testy

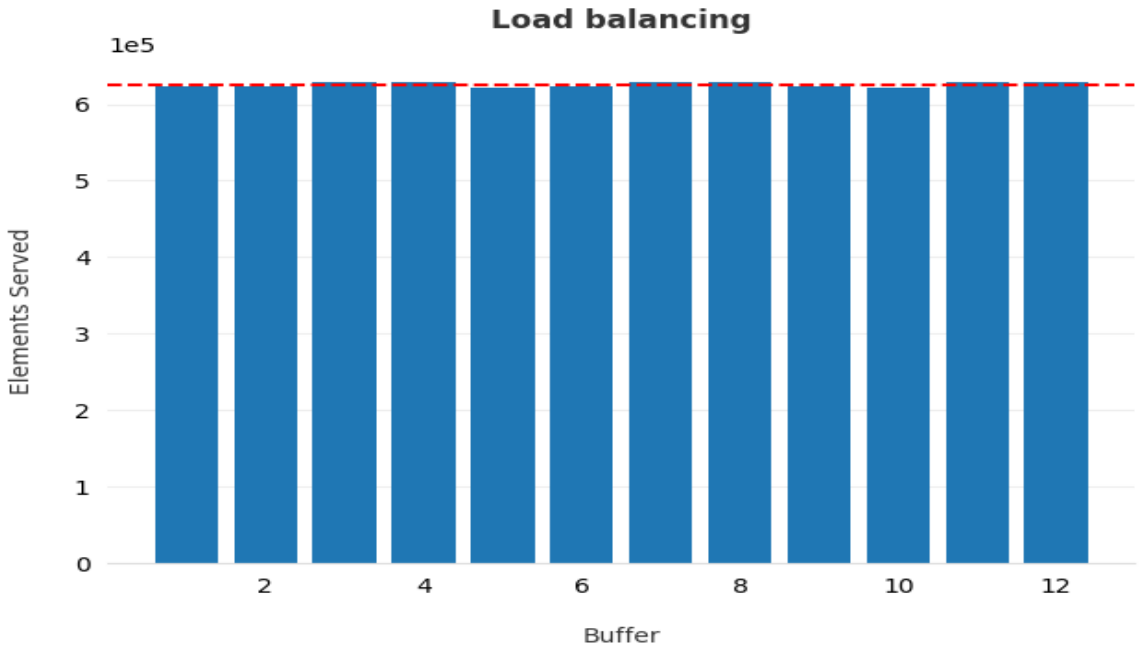
4.1 Test obciążenia



Rysunek 2: Obciążenie buforów dla 6 Konsumentów, 6 Producentów i 3 Buforów

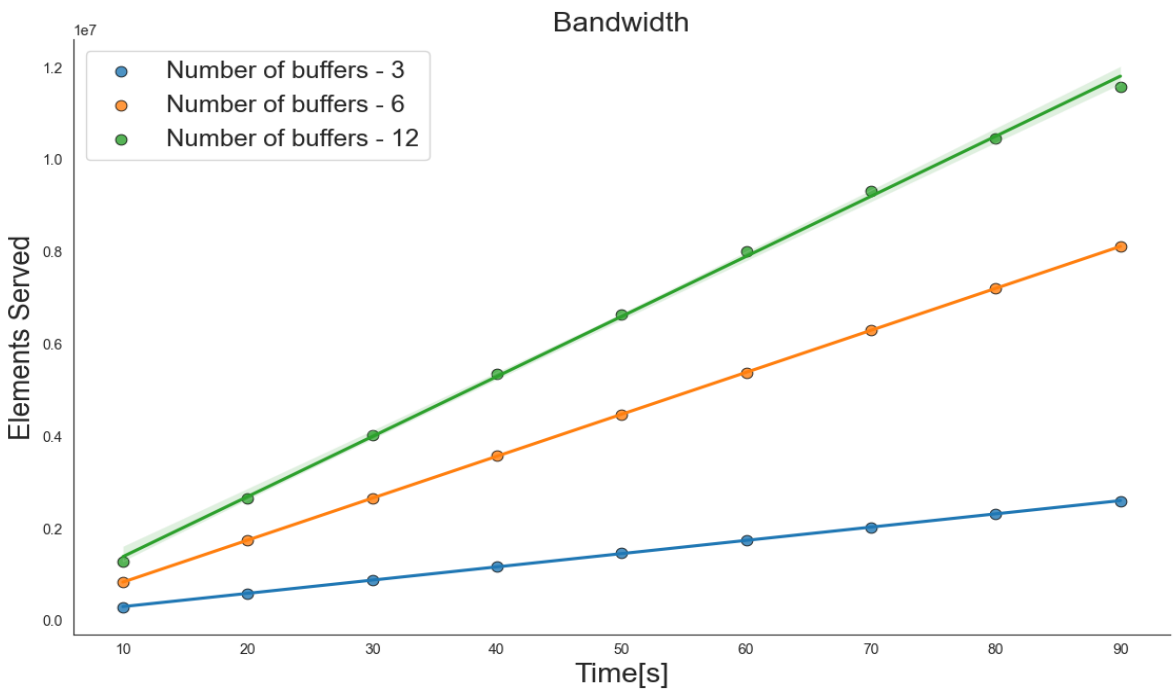


Rysunek 3: Obciążenie buforów dla 6 Konsumentów, 6 Producentów i 6 Buforów



Rysunek 4: Obciążenie buforów dla 6 Konsumentów, 6 Producentów i 12 Buforów

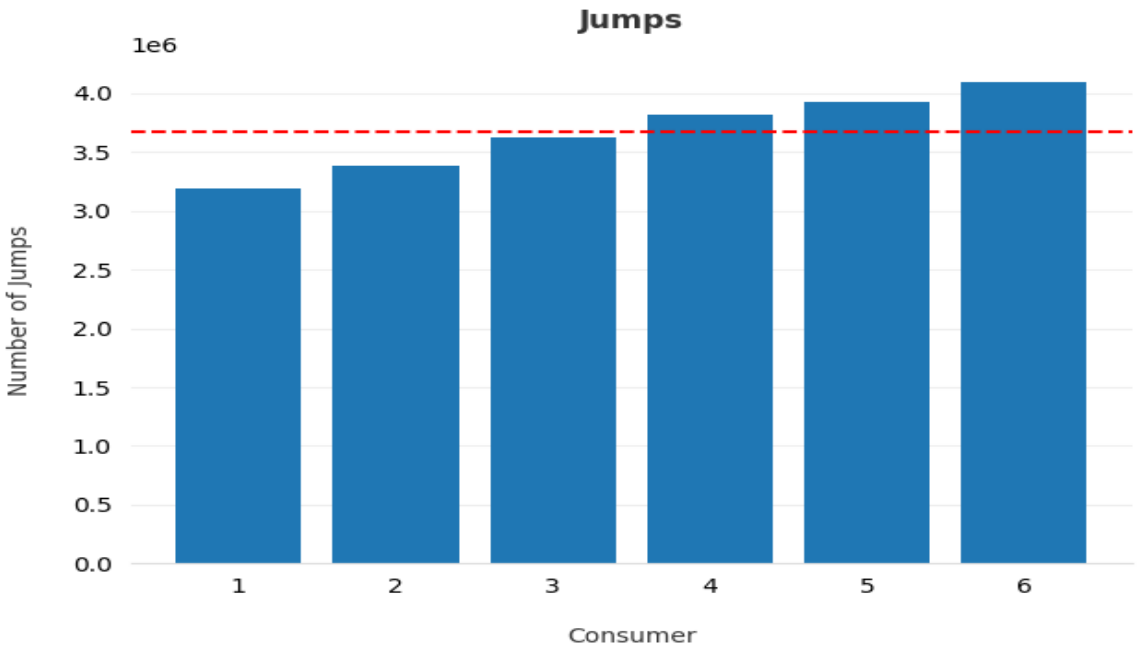
4.2 Test przepustowości



Rysunek 5: Porównanie przepustowosci dla różnych parametrów zadania

4.3 Liczba skoków Konsumentów

Konsumenci podczas wybierania kandydata na dostawce często „skaczą"pomiędzy Buforami wysyłając im swoję prośby. Liczba tych zapytań jest bardzo duża i przedstawiona jest na poniższym rysunku (wariant z dwunastoma buforami). Dla innych wariantów wykresy mają taki sam kształt. Ten sam problem dotyczy Producentów.

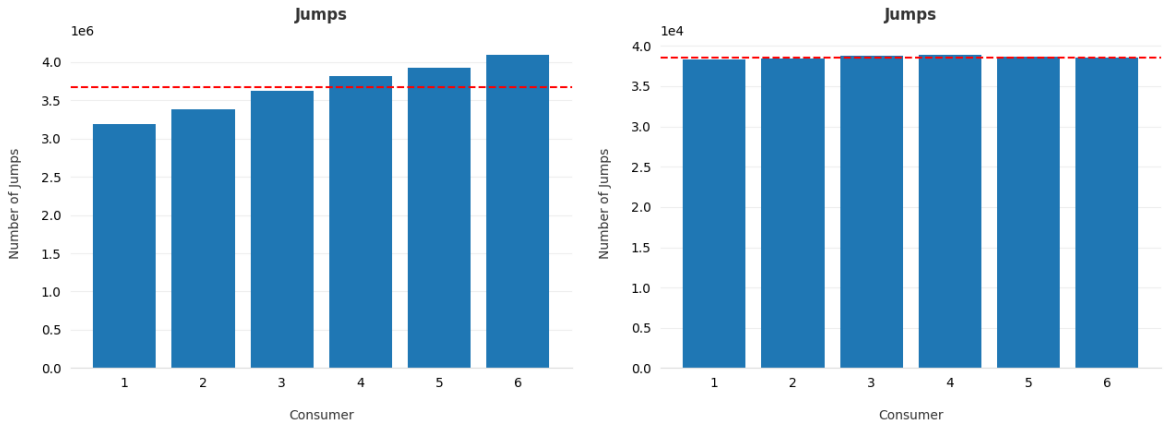


Rysunek 6: Liczba skoków wykonywanych przez Konsumentów

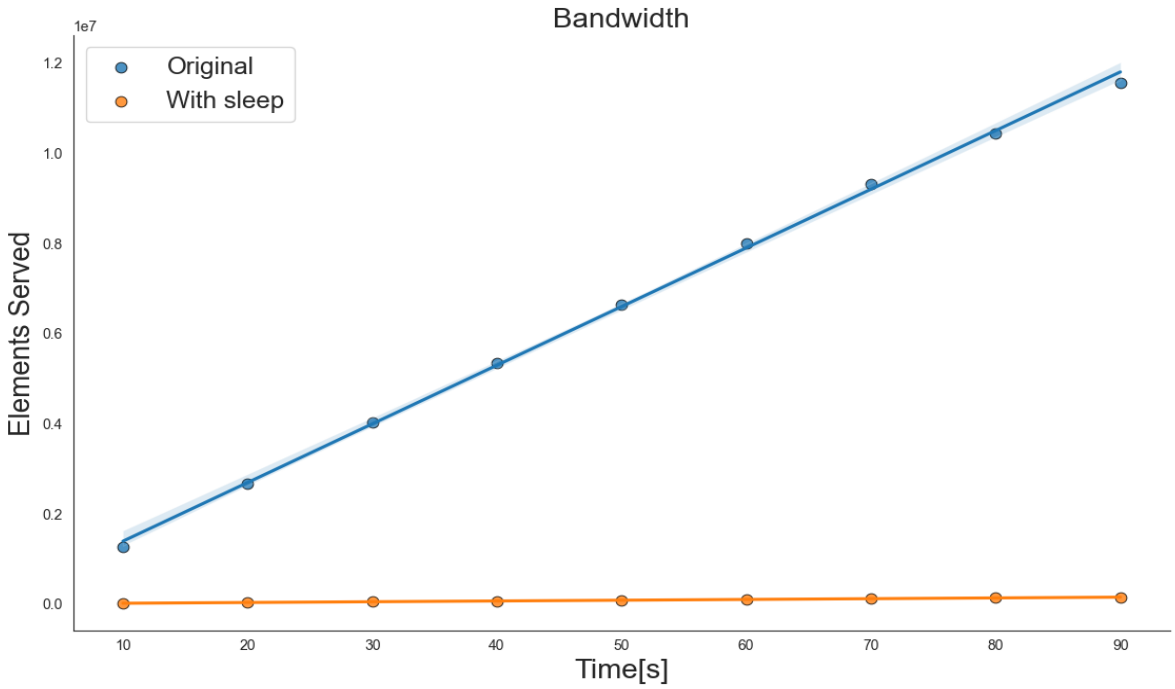
5 Wariant z metodą sleep

Aby ograniczyć liczbę ofert, które wysyłają sobie wątki, wprowadzoną mechanizm usypiania. Jeżeli oferta wątku A zostanie odrzucona przez bufor, to zasypia on na jedną milisekundę. Ma to zapobiec „gorączce próśb", która to występuję gdy wszystkie bufory są w tym samym stanie, bądź są już zarezerwowane.

5.1 Testy



Rysunek 8: Porównanie liczby skoków wariantu bez metody sleep(lewo) oraz z metodą sleep(prawo)



Rysunek 9: Porównanie efektywności metod

6 Zalety i wady

Zalety

- Rozwiązanie działa. Dzięki mechanizmowi zapytań nie zaobserwowano również zakleszczenia.
- Metoda w poprawny sposób równoważy obciążenie pomiędzy wszystkie bufory niezależnie od ich liczby.
- Rozwiązanie pozwala na wysyłanie większej liczby elementów pomiędzy wątkami, pod warunkiem stosowania zasady: Każdy wątek wysyła(dostaje) do(od) bufora conajwyżej n / 2 elementów, gdzie n to pojemność bufora.

Wady

- Liczba kanałów potrzebnych do realizacji architektury rośnię kwadratowo względem liczby buforów. Rozwiązanie słabo się skaluję. Aby rozwiązać ten problem możliwa jest próba zaimplementowania architektury podobnej do wyżej opisanej ale z stałą liczbą połączeń pomiędzy buforami a wątkami. Oznacza, to że każdy bufor posiadałby co najwyżej k kanałów, gdzie k to hiperparametr.
- Liczba zapytań wysyłanych przez producentów i konsumentów, które zostają odrzuconę jest ogromna. Zastosowania metody sleep() znacznie zmniejsza liczbę odrzucanych ofert. Robi to niestety wielkim kosztem efektywności. Optymalne dostosowanie czasu oczekiwania zależy od określonego problemu. Aby rozwiązać ten problem można również zwiększyć liczbę buforów. Nie rozwiązuje to jednak problemu tak gwałtownie jak metoda sleep().