Rozproszony bufor w CSP

Ida Ciepiela

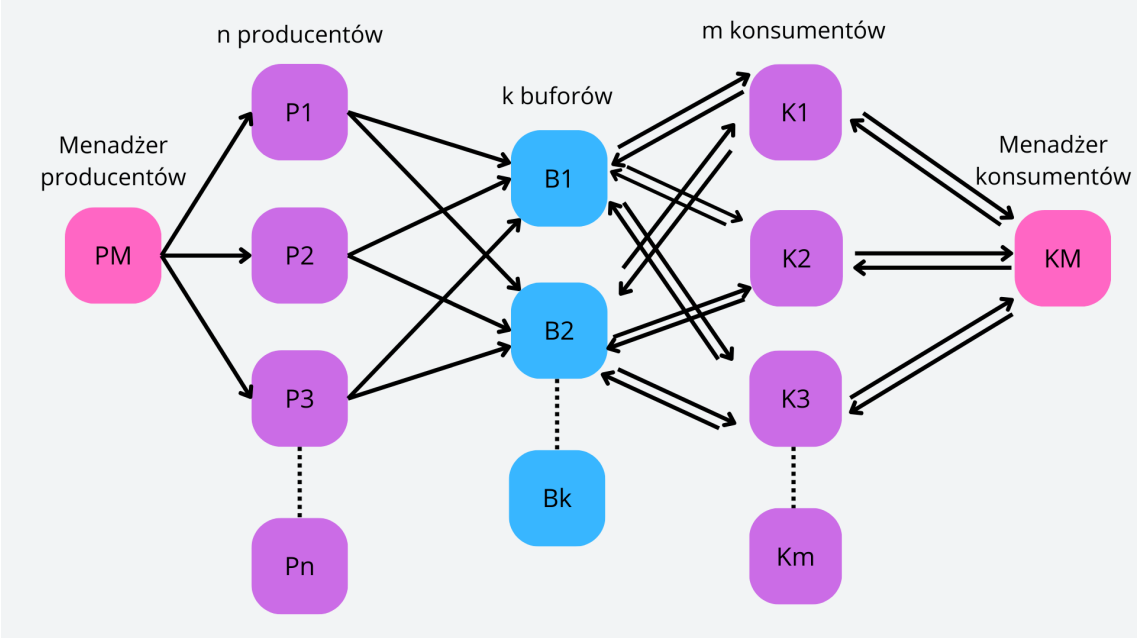
9 grudnia 2024

Spis treści

1 Wstęp

Na potrzeby zadania został zaimplementowany problem producenta i konsumenta z rozproszonym buforem w JCSP.

1.1 Schemat



Rysunek 1: Schemat rozwiązania

1.2 Idea rozwiązania

Moje rozwiązanie bazowało na wprowadzeniu dwóch menadżerów. Pierwszy nadzoruje kierunek wysyłki porcji przez producentów, a drugi zarządza tym, do którego bufora zgłaszają się konsumenci. Obaj menadżerowie stosują algorytm round-robin, aby określić, do którego bufora powinien zgłosić się kolejny producent lub konsument. Przy takiej samej pojemności buforów jest to niezawodna metoda zapewniająca ich równomierne obciążenie.

2 Analiza rozwiązania

2.1 Opis środowiska sprzętowego

1. System operacyjny:
    - Windows 11 Home
2. Procesor CPU:
    - AMD Ryzen 5 5600H with Radeon Graphics
    - 6 rdzeni fizycznych
    - 12 rdzeni logicznych
    - 2.50 GHz - szybkość podstawowa
3. Pamięć RAM:
    - 16 GB
4. Środowisko wykonawcze:
    - IntelliJ IDEA 2023.2.5
    - Oracle OpenJDK 23.0.1

2.2 Przebieg eksperymentów

Aby ocenić wydajność i skalowalność przedstawionego rozwiązania, zrealizowano cztery eksperymenty:

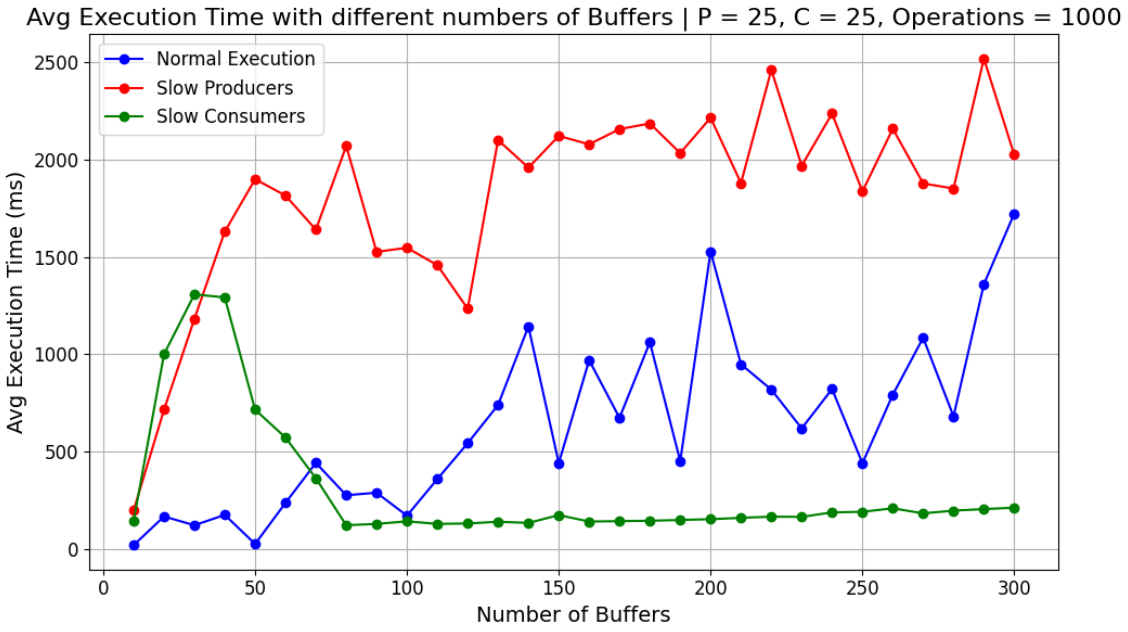
- Ustalono średni czas realizacji 1000 operacji przy różnych liczbach buforów
- Ustalono średni czas realizacji 1000 operacji przy różnych liczbach producentów
- Ustalono średni czas realizacji 1000 operacji przy różnych liczbach konsumentów
- Ustalono średni czas realizacji 1000 operacji dla równej liczby producentów i konsumentów

Każdy z eksperymentów wykonano w trzech wariantach:

- Bez żadnych zmian
- Z wolnymi producentami (symulowany czas produkcji porcji: 1 ms do 3 ms)
- Z wolnymi konsumentami (symulowany czas konsumpcji porcji: 1 ms do 3 ms)

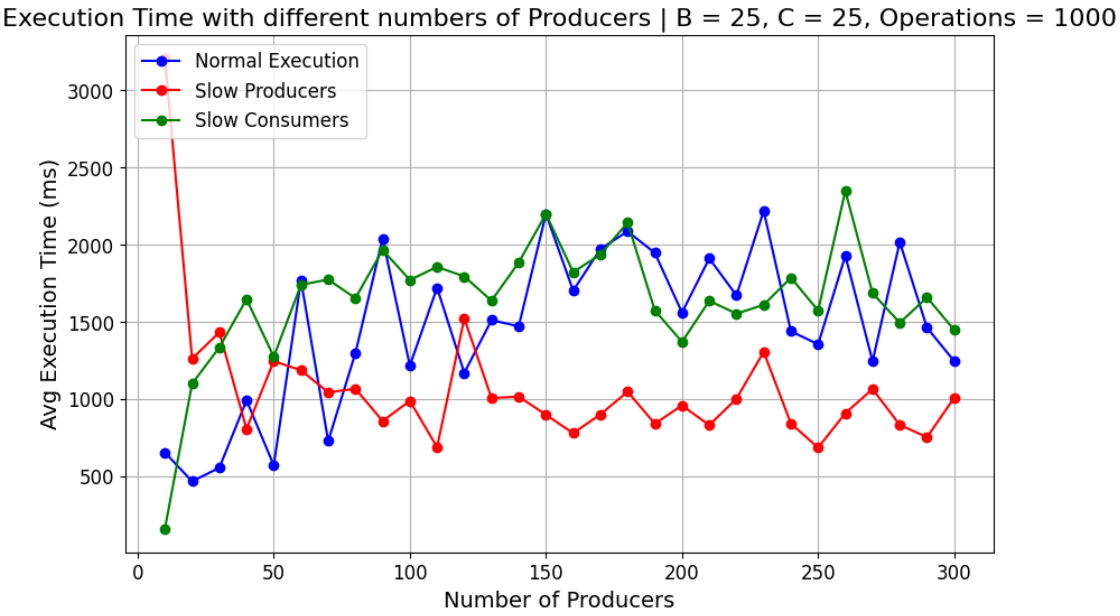
Dla każdego z eksperymentów w każdym z wariantów pomiary zostały zebrane 5 razy, obliczono z nich średnią i zapisano w plikach csv. Na ich podstawie sporządzone zostały wykresy.

2.3 Wyniki pomiarów



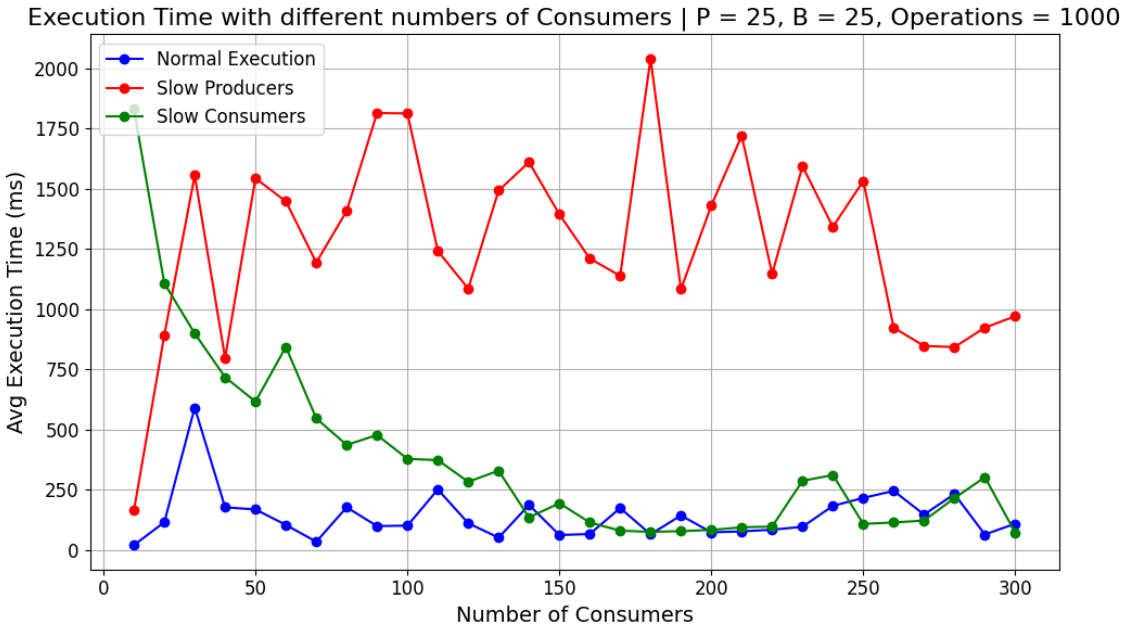
Rysunek 2: Średnie czasy wykonywania 1000 operacji przy różnych liczbach buforów, 25 producentów i 25 konsumentów

W przypadku normalnej pracy system działa najlepiej przy średniej liczbie buforów, a zbyt wiele buforów zwiększa czas z powodu narzutu komunikacyjnego. Wolni producenci znacząco wydłużają czas, szczególnie przy małej liczbie buforów, ponieważ konsumenci czekają na dane. Wolni konsumenci powodują mniejszy wzrost czasu, a większa liczba buforów dobrze niweluje ich wpływ.



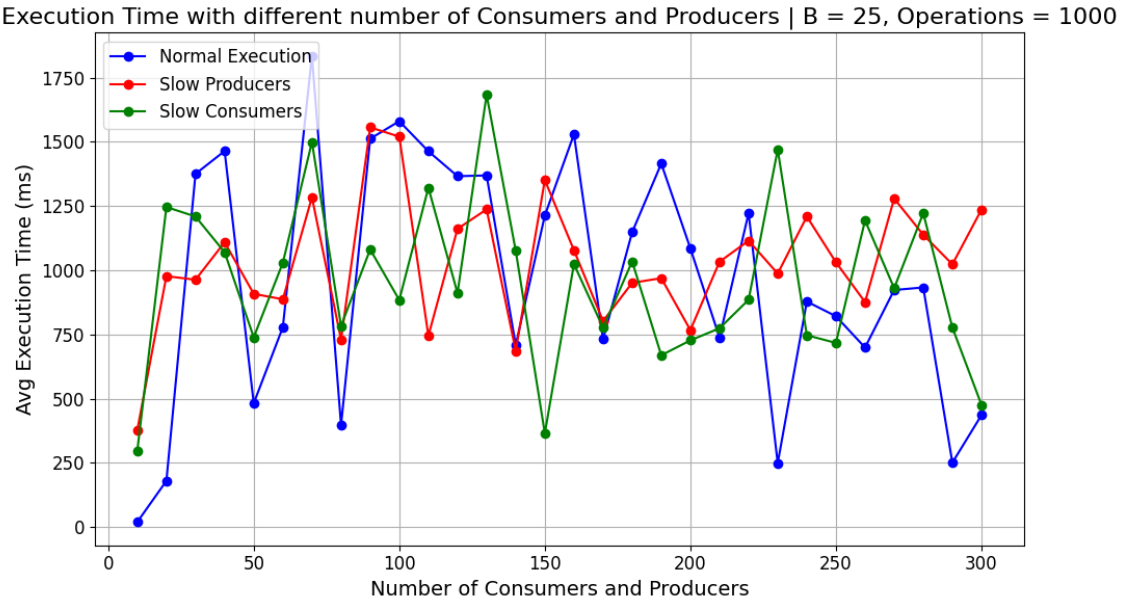
Rysunek 3: Średnie czasy wykonywania 1000 operacji przy różnych liczbach producentów, 25 buforów i 25 konsumentów

Program osiąga stosunkowo wysokie wyniki dla każdej z wariacji. Nie wiem dlaczego, ale najlepsze wyniki osiągane są dla dużej ilości wolnych producentów.



Rysunek 4: Średnie czasy wykonywania 1000 operacji przy różnych liczbach konsumentów, 25 buforów i 25 producentów

Wykres pokazuje, że przy normalnej pracy konsumentów średni czas wykonania stabilizuje się na niskim poziomie przy umiarkowanej liczbie konsumentów. Wolni producenci powodują wysoki czas wykonania, niezależnie od liczby konsumentów. W przypadku wolnych konsumentów zwiększanie ich liczby znacząco poprawia wydajność, aż do momentu zrównoważenia systemu.



Rysunek 5: Średnie czasy wykonywania 1000 operacji przy różnych liczbach konsumentów i producentów przy 25 buforach

W chwili, gdy liczba producentów i konsumentów jest równa, program wykazuje zwykle niestabilność oraz osiąga stosunkowo wysokie czasy działania. W tej sytuacji nie miało znaczenia, czy używani byli wolni producenci lub wolni konsumenci, bo każda z wariacji wykazywała podobną niestabilność.

3 Wnioski

3.1 Ogólne wnioski

Na podstawie wykresów można zauważyć, że moje rozwiązanie najlepiej funkcjonuje, gdy liczba konsumentów przewyższa liczbę buforów i producentów. Również przy dużej liczbie buforów oraz konsumentach, którzy potrzebują czasu na przetworzenie porcji, działa ono idealnie. Zwiększenie liczby producentów zazwyczaj pogarsza wyniki, prowadząc do znacznie dłuższych czasów niezależnie od wariantu.

3.2 Zalety i wady rozwiązania

Zalety:

- Równoważne obciążenie buforów dzięki algorytmowi Round-robin. Dzięki temu zasoby są wykorzystywane efektywnie
- Schemat prosty w implementacji
- Architektura umożliwia łatwe skalowanie liczby producentów, konsumentów i buforów bez konieczności wprowadzania istotnych zmian w logice menedżerów.

Wady:

- Nierównomierne obciążenie komunikacji
- Nadmierna komunikacja po stronie konsumentów
- Przy bardzo dużej liczbie buforów, producentów i konsumentów koordynacja przez dwóch menedżerów może stać się wąskim gardłem systemu

3.3 Skalowalność

Rozwiązanie wykazuje dużą skalowalność, szczególnie w przypadku zwiększania liczby konsumentów i buforów. Architektura systemu umożliwia łatwe dodawanie nowych producentów, konsumentów i buforów bez konieczności wprowadzania istotnych zmian w schemacie. Jednak skalowalność może być ograniczona w przypadku bardzo dużych systemów, gdzie dwaj menedżerowie stają się wąskim gardłem.

3.4 Możliwości optymalizacji

- Zmniejszenie narzutu komunikacyjnego przez lokalne kolejkowanie żądań konsumentów, co ograniczy konieczność ciągłej komunikacji z menedżerem (każdy konsument miałby swojego round-robina)
- Konsument może mieć przydzielane kilka operacji na raz od menedżera, co pozwala zoptymalizować obsługę wielu żądań.
- Hierarchiczna struktura menedżerów. Dla bardzo dużych systemów można wprowadzić wielopoziomową hierarchię menedżerów.