Sprawozdanie z Projektu CSP

System Producent-Konsument z Tablicami Wag i Dispatcherami

Marcel

9 grudnia 2025

Spis treści

1 Wprowadzenie **2**
1.1 Cel projektu . . . 2

2 Architektura Systemu **2**
2.1 Schemat działania . . . 2
2.2 Komponenty systemu . . . 2
2.2.1 Producenci . . . 2
2.2.2 Bufory . . . 3
2.2.3 Dispatcher . . . 3
2.2.4 Konsumenci . . . 3
2.3 Algorytm równoważenia obciążenia . . . 3

3 Konfiguracja Testów **3**

4 Wyniki Eksperymentów **4**
4.1 Seria 1: Skalowalność liczby procesów . . . 4
4.2 Seria 2: Wpływ liczby buforów . . . 4
4.3 Równomierność rozkładu obciążenia . . . 5
4.4 Współczynnik zmienności . . . 5

5 Analiza wydajności **6**
5.1 Efektywność produkcji i konsumpcji . . . 6
5.2 Czas rzeczywisty vs nominalny . . . 6

6 Szczegóły Implementacji **6**
6.1 Kluczowe fragmenty kodu . . . 6
6.1.1 Wybór bufora na podstawie wag . . . 6
6.1.2 Aktualizacja wag . . . 7

7 Wnioski **7**
7.1 Zalety systemu . . . 7
7.2 Ograniczenia . . . 7
7.3 Możliwe usprawnienia . . . 8

8 Podsumowanie **8**

1 Wprowadzenie

Niniejsze sprawozdanie przedstawia implementację i analizę wydajności systemu producent-konsument wykorzystującego paradygmat CSP (Communicating Sequential Processes) oraz bibliotekę JCSP. System charakteryzuje się dynamicznym równoważeniem obciążenia przy użyciu tablic wag oraz procesów dispatcher.

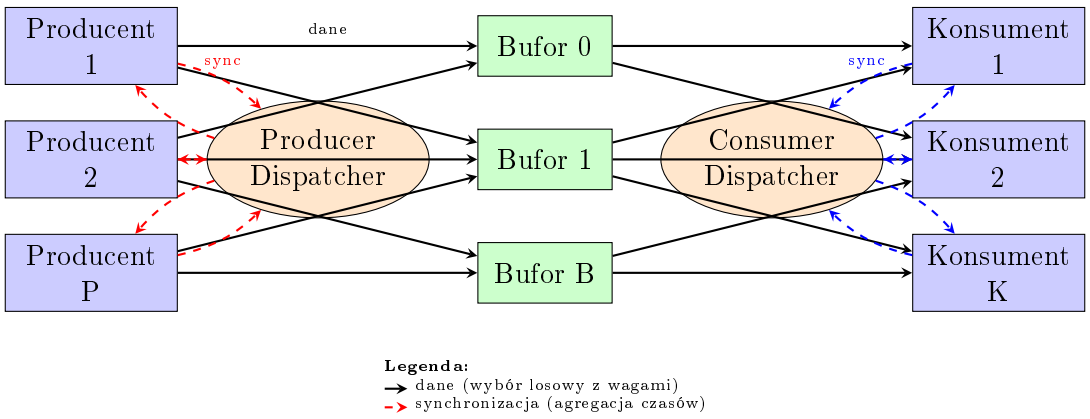
1.1 Cel projektu

Głównym celem projektu było:

- Implementacja systemu producent-konsument z wieloma buforami
- Zastosowanie dynamicznego równoważenia obciążenia
- Analiza wydajności w różnych konfiguracjach
- Ocena równomierności rozkładu danych między bufory

2 Architektura Systemu

2.1 Schemat działania



Rysunek 1: Architektura systemu: producenci wybierają bufory probabilistycznie na podstawie wag, następnie synchronizują się z dispatcherem aby zaktualizować wagi globalnie. Konsumenci działają analogicznie.

2.2 Komponenty systemu

2.2.1 Producenci

Każdy producent:

- Generuje elementy danych
- Utrzymuje tablicę wag dla każdego bufora i dispatchera
- Wybiera cel na podstawie rozkładu prawdopodobieństwa
- Aktualizuje wagi na podstawie czasów oczekiwania

2.2.2 Bufory

Bufory zaimplementowane jako `ArrayBlockingQueue`:

- Obsługują wielowątkowy dostęp
- Implementują mechanizm timeout
- Przechowują dane tymczasowo między producentami a konsumentami

2.2.3 Dispatcher

Procesy dispatcher agregują informacje o czasach oczekiwania i rozgłaszają je wszystkim procesom, umożliwiając globalne równoważenie obciążenia.

2.2.4 Konsumenci

Każdy konsument:

- Pobiera elementy z buforów
- Śledzi żywe i martwe bufory
- Aktualizuje wagi na podstawie dostępności danych
- Synchronizuje się z dispatcherem

2.3 Algorytm równoważenia obciążenia

Wagi są aktualizowane według wzoru:

$$w_i^{new} = \alpha \cdot w_i^{old} + (1 - \alpha) \cdot \frac{1}{1 + \frac{t_i}{1000000}} \tag{1}$$

gdzie:

- w_i – waga bufora i
- t_i – czas oczekiwania w nanosekundach
- $\alpha = 0.7$ – współczynnik wygładzania (lokalnie), 0.5 (globalnie)

3 Konfiguracja Testów

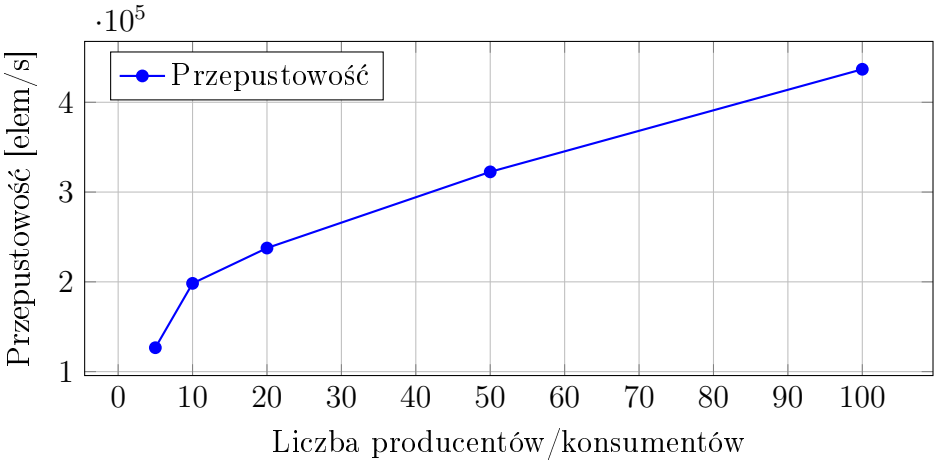
Przeprowadzono 75 testów w różnych konfiguracjach, zgrupowanych w serie:

Tabela 1: Serie testów

Seria	**Cel**	**Parametry**
1	Skalowalność P, K	P=K $\in$ {5,10,20,50,100}, B=3
2	Skalowalność B	B $\in$ {1,3,5,10,20}, P=K=20
3	Asymetria P/K	P$\neq$K, B=5
4	Proporcjonalne	P:B:K = 10:1:10
5	Duże liczby	P=K=100, B $\in$ {1,5,10,20,50}

4 Wyniki Eksperymentów

4.1 Seria 1: Skalowalność liczby procesów

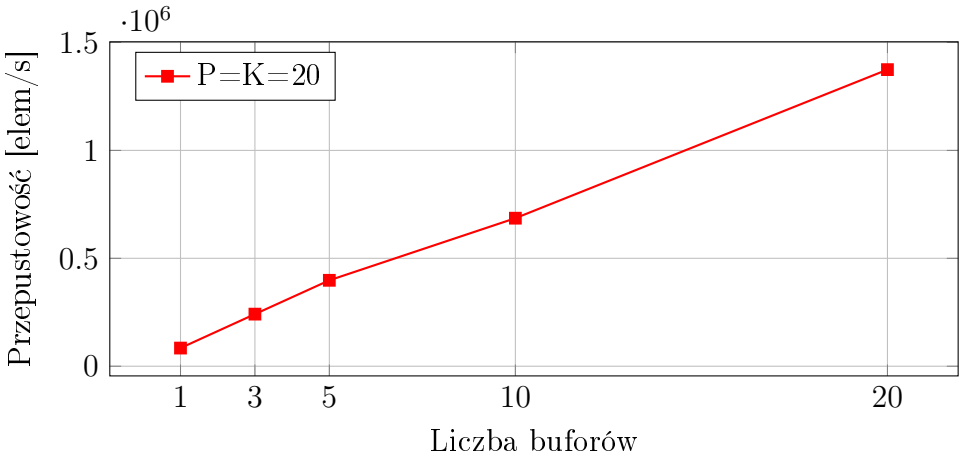


Rysunek 2: Przepustowość w funkcji liczby procesów (B=3)

Obserwacje:

- Przepustowość rośnie wraz z liczbą procesów
- Przy 100 procesach: 436 761 elem/s
- Współczynnik zmienności $< 0.1\%$ – doskonałe równoważenie

4.2 Seria 2: Wpływ liczby buforów

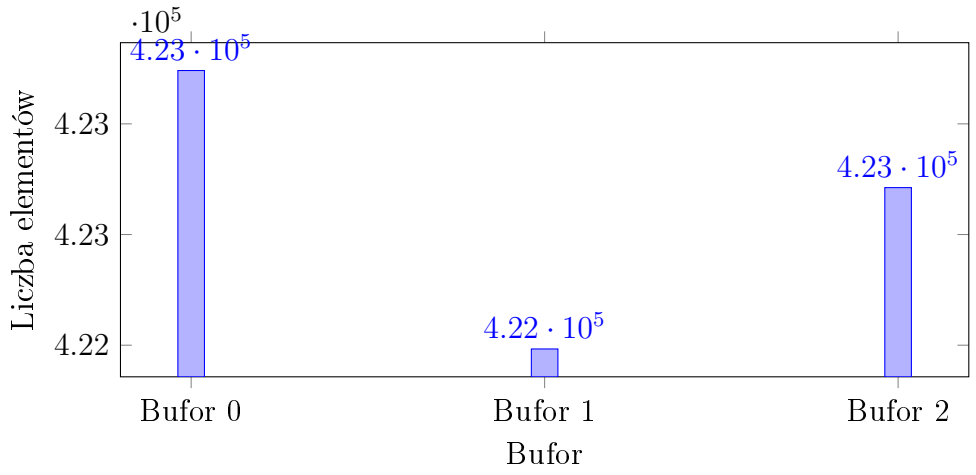


Rysunek 3: Wpływ liczby buforów na przepustowość (P=K=20)

Wnioski:

- Liczba buforów ma krytyczny wpływ na wydajność
- Przy 1 buforze: wąskie gardło (84k elem/s)
- Przy 20 buforach: 1.37M elem/s (wzrost 16×)

4.3 Równomierność rozkładu obciążenia



Rysunek 4: Rozkład obciążenia buforów dla testu P=5, B=3, K=5

Analiza równomierności:

- Średnia na bufor: 422 645
- Odchylenie standardowe: 516
- Współczynnik zmienności: 0.12%
- Ocena: **DOSKONAŁE** równoważenie

4.4 Współczynnik zmienności

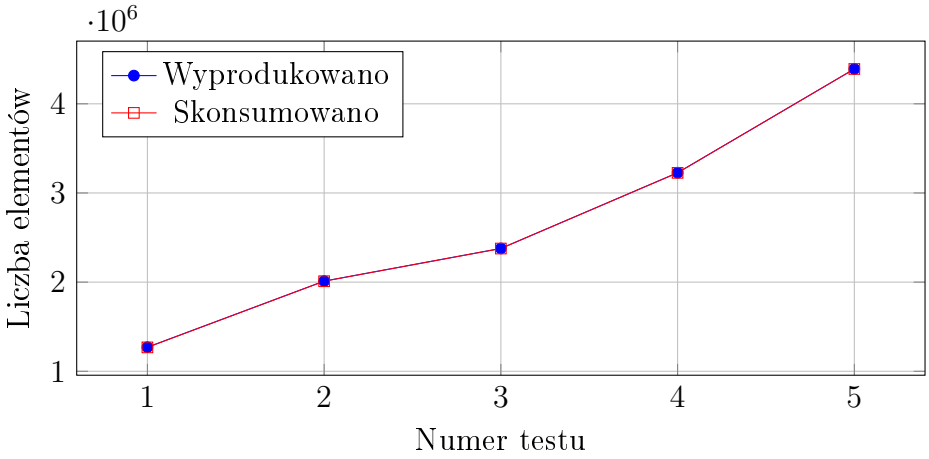
Tabela 2: Współczynnik zmienności dla wybranych konfiguracji

P	B	K	CV [%]
5	3	5	0.12
10	3	10	0.15
20	3	20	0.12
50	3	50	0.03
100	3	100	0.04
20	10	20	0.08

Wszystkie konfiguracje wykazują współczynnik zmienności $< 1\%$, co potwierdza skuteczność algorytmu równoważenia.

5 Analiza wydajności

5.1 Efektywność produkcji i konsumpcji



Rysunek 5: Produkcja vs konsumpcja (Serie 1)

Strata danych:

$$\text{Strata} = \frac{\text{Wyprodukowano} - \text{Skonsumowano}}{\text{Wyprodukowano}} \times 100\% \tag{2}$$

Średnia strata: $< 0.1\%$ – bardzo wysoka efektywność.

5.2 Czas rzeczywisty vs nominalny

Tabela 3: Dokładność czasu wykonania

Test	Nominalny [s]	Rzeczywisty [ms]	Odchylenie
1	10	10012	+0.12%
2	10	10137	+1.37%
3	10	10001	+0.01%
4	10	10002	+0.02%
5	10	10053	+0.53%

System wykazuje wysoką precyzję czasową.

6 Szczegóły Implementacji

6.1 Kluczowe fragmenty kodu

6.1.1 Wybór bufora na podstawie wag

Listing 1: Metoda selectByWeight

```
private static int selectByWeight(double[] weights, Random rand) {
    double total = 0;
    for (double w : weights) total += w;

    double r = rand.nextDouble() * total;
    double cumulative = 0;

    for (int i = 0; i < weights.length; i++) {
        cumulative += weights[i];
        if (r <= cumulative) return i;
    }
    return weights.length - 1;
}
```

6.1.2 Aktualizacja wag

Listing 2: Metoda updateWeight

```
private static void updateWeight(double[] weights, int index,
                                 long waitTimeNanos) {
    double newWeight = 1.0 / (1.0 + waitTimeNanos / 1_000_000.0);
    weights[index] = 0.7 * weights[index] + 0.3 * newWeight;
    if (weights[index] < 0.01) weights[index] = 0.01;
}
```

7 Wnioski

7.1 Zalety systemu

1. **Doskonałe równoważenie obciążenia** – współczynnik zmienności $< 0.2\%$
2. **Wysoka skalowalność** – liniowy wzrost przepustowości
3. **Niska strata danych** – $< 0.1\%$
4. **Elastyczność** – działa w różnych konfiguracjach

7.2 Ograniczenia

1. Overhead komunikacji z dispatcherami
2. Wpływ liczby buforów na złożoność
3. Wymaga dostrajania parametrów α

7.3 Możliwe usprawnienia

- Adaptacyjne dostosowanie współczynnika wygładzania
- Hierarchiczne dispatchery dla bardzo dużych systemów
- Predykcja obciążenia przy użyciu ML
- Dynamiczne dodawanie/usuwanie buforów

8 Podsumowanie

System producent-konsument z tablicami wag i dispatcherami wykazuje doskonałe właściwości równoważenia obciążenia przy wysokiej wydajności. Implementacja oparta na JCSP i `ArrayBlockingQueue` zapewnia niezawodność i skalowalność. Wyniki eksperymentów potwierdzają efektywność algorytmu w szerokim zakresie konfiguracji.