Teoria współbieżności

Problem producent-konsument - porównanie wydajnościowe

Mateusz Łopaciński

1. Sposób przeprowadzenia eksperymentu

Zarówno implementacja problemu producenta-konsumenta z wykorzystaniem jednego locka i 4 Conditions, jak i implementacja przy użyciu 3 locków i 2 Conditions, zostały uruchomione dla stałego czasu rzeczywistego, wynoszącego 1 sekundę. Oba programy zostały uruchomione dla różnych liczb wątków (od 10 do 100, z krokiem wynoszącym 10). Testy zostały wykonane dla bufora o wielkości 100 elementów. Wątki konsumentów konsumują losową liczbę porcji w przedziale od 1 do 50, a wątki producentów produkują losową liczbę porcji w przedziale od 1 do 50.

W celu zapewnienia dokładnych wyników, dla każdego zestawu parametrów, testy zostały powtórzone 10-krotnie, a następnie, wyniki testów zostały uśrednione.

W każdym z testów liczba producentów była równa liczbie konsumentów i wynosiła połowę liczby wątków uruchamianych w danym teście.

Po upływie 1 sekundy czasu rzeczywistego, wszystkie wątki (producenci oraz konsumenci) zostały zatrzymane, po czym zmierzony został sumaryczny czas CPU, a także czas CPU przypadający na 1 wątek, liczba wykonanych operacji na buforze (ile razy wątki uzyskały dostęp do monitora) oraz liczba operacji przypadająca na jednostkę czasu CPU. Wyniki zostały zebrane w następnym punkcie sprawozdania.

1. Wyniki eksperymentu
   1. Liczba operacji w zależności od liczby wątków

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

*2.1.1. Wykres liczby operacji w zależności od liczby wątków dla rozwiązania na 4 Conditions*

*2.1.2. Wykres liczby operacji w zależności od liczby wątków dla rozwiązania na zagnieżdżonych lockach*

* 1. Łączny czas CPU w zależności od liczby wątków

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

*2.2.2. Wykres zależności łącznego czasu CPU od liczby wątków dla rozwiązania na zagnieżdżonych lockach*

*2.2.1. Wykres zależności łącznego czasu CPU od liczby wątków dla rozwiązania na 4 Conditions*

* 1. Czas CPU przypadający na 1 wątek

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

*2.3.2. Wykres zależności czasu CPU, przypadającego na pojedynczy wątek, w zależności od liczby wątków dla rozwiązania na zagnieżdżonych lockach*

*2.3.1. Wykres zależności czasu CPU, przypadającego na pojedynczy wątek, w zależności od liczby wątków dla rozwiązania na 4 Conditions*

* 1. Liczba operacji na sekundę łącznego czasu CPU

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

*2.4.2. Wykres zależności liczby operacji na sekundę czasu CPU od liczby wątków dla rozwiązania na zagnieżdżonych lockach*

*2.4.1. Wykres zależności liczby operacji na sekundę czasu CPU od liczby wątków dla rozwiązania na 4 Conditions*

1. Obserwacje

- Dla takiej samej liczby wątków, rozwiązanie z wykorzystaniem zagnieżdżonych locków pozwala na uzyskanie prawie 2-krotnie większej liczby wykonanych operacji na buforze,

- W rozwiązaniu na zagnieżdżonych lockach, łączny czas CPU oraz czas CPU, przypadający na pojedynczy wątek, jest około 1.5 razy większy niż w rozwiązaniu z wykorzystaniem 4 Conditions,

- Liczba operacji, przypadająca na jednostkę czasu CPU, jest zbliżona w obu rozwiązaniach.

1. Wnioski

- Rozwiązanie z wykorzystaniem zagnieżdżonych locków jest bardziej efektywne od rozwiązania w oparciu o 4 Conditions,

- Ponieważ czas CPU jest większy w przypadku rozwiązania na zagnieżdżonych lockach, podczas gdy czas rzeczywisty jest taki sam dla obu implementacji, można wywnioskować, iż wątki częściej znajdują się w stanie **Running** w rozwiązaniu z zagnieżdżonymi lockami. Oznacza to, że mniej zasobów jest marnowanych na czekające na dostęp do zasobów (do monitora) wątki i rzadziej występują sytuacje, w których wątki oczekują na wejście do monitora niż w przypadku rozwiązania na 4 Conditions.