Systemy Operacyjne

Programy Symulacyjne

Kacper Leszczyński – 273125 – TIN

Spis treści:

1 Wstęp

Celem Projektu jest implementacja czterech wybranych algorytmów i programów symulacyjnych, przeprowadzenie prostych eksperymentów porównujących ich działanie oraz prezentacja uzyskanych rezultatów. Z puli zadań planowania czasu procesora wybrane zostały:

1. FCFS – First Come First Serve
2. LCFS – Last Come First Serve

Natomiast z puli algorytmów zastępowania stron wybrane zostały:

1. FIFO – First In First Out
2. LRU – Least Recently Used

2 Opis Algorytmów

2.1 Algorytmy planowania czasu procesora

2.1.1 FCFS (First Come First Serve) – Najprostszy algorytm planowania procesów. Procesy obsługiwane są w kolejności w jakiej przybyły do kolejki. Proces, który pierwszy trafił do kolejki, zawsze zostanie obsłużony jako pierwszy. Nie uwzględnia takich parametrów jak czas czy priorytet potrzebny do wykonania danego zadania. Zaletą tego algorytmu jest prostota w zrozumieniu oraz implementacji. Wadą natomiast jest fakt, że może prowadzić do efektu konwoju (ang. Convoy effect), gdzie krótkie zadania muszą czekać, aż wykonają się zadania długie. Wpływa to również na niską wydajność kodu zwłaszcza w wypadku dużej różnicy czasu wykonywania procesów.

2.1.2 LCFS (Last Come First Serve) – W przeciwieństwie do algorytmu FCFS, procesy są wykonywanie od końca, tzn. że ostatni proces który przybył do kolejki, jest wykonywany w pierwszej kolejności. Istnieją dwie możliwości implementacji algorytmu LCFS: z wywłaszczeniami lub bez. W tej analizie skupimy się algorytmie który porzuca aktualnie wykonywany proces na rzecz nowego. Podobnie jak algorytm FCFS nie uwzględnia parametrów takich jak czas czy priorytet. Zaletą jest prostota implementacji. Istnieje niestety kilka wad LCFS takich jak: efekt konwoju, brak uwzględnienia czasów wykonania, oraz niektóre procesy mogą czekać na swoją kolei bardzo długo

2.2 Algorytmy zastępowania stron

2.2.1 FIFO (First In First Out) – Działa na zasadzie kolejki. Pierwsza strona, która została wczytana do pamięci, jest pierwsza do opuszczenia, gdy nowa strona musi zostać dodana, a pamięć jest już pełna. Zaletami takiego algorytmu jest oczywista prostota implementacji, a także w porównaniu z LRU brak potrzeby na zapamiętywanie dodatkowych informacji takich jak

czas dosępu. Wadami natomiast jest brak priorytetów w nadpisywaniu starych stron.

2.2.2 LRU (Least Recently Used) – Opiera się na założeniu, że strona, która nie była używana przez najdłuższy czas, jest najmniej ważna i może być zastąpiona. Każda strona jest oznakowana znacznikiem czasowym. Gdy dana strona jest używana jej znacznik zostaje zaktualizowany. W zaletach możemy wymienić, iż w większości scenariuszy jest skuteczniejszy od FIFO, oraz względną prostotę implementacji. Wadą natomiast na pewno jest konieczność monitorowania czasu dostępu każdej strony, co może być kosztowne.

3. Algorytmy planowania czasu procesora

3.1 Parametry Eksperymentu

3.1.1 Maksymalny czas nadejścia procesu – 80 [s]

3.1.2 Początkowa liczba procesów - 40

3.1.3 Zakres procesów - <1,9>

3.1.4 Zakres czasu procesów - <1,15>

3.2 Parametry Symulacji

3.2.1 Początkowa liczba procesów – początkowa liczba procesów wylosowanych do symulacji

3.2.2 Zakres procesów – zakres który może zostać wylosowany od 1 do podanej liczby. Oznacza indeks procesu.

3.2.3 Zakres czasu procesów – zakres który może zostać wylosowany od 1 do podanej liczby i odnoszący się do czasu który dany proces musi być wykonywany

3.2.4 Maksymalny czas nadejścia procesu –oznacza ostatni moment od zaczęcia programu w którym nowe procesy mogą zostać wylosowane

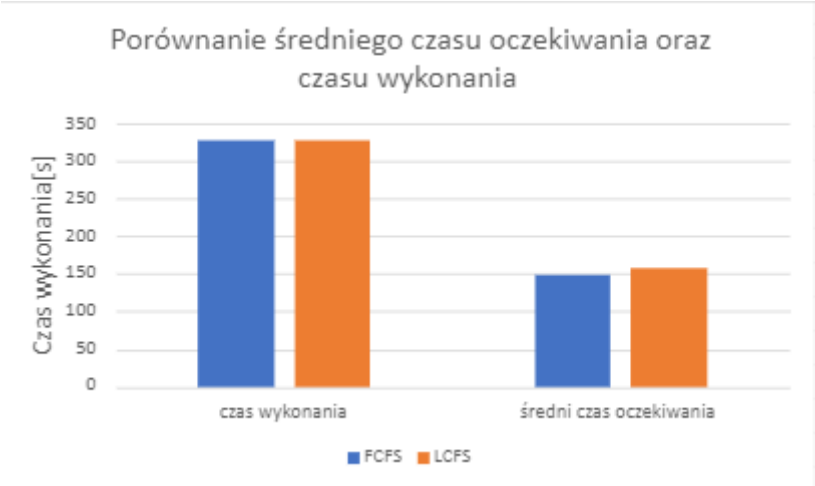
3.2.5 Średni czas oczekiwania –policzony średni czas oczekiwania każdego zadania na wykonanie wszystkich zadań

3.2.6 **Czas wykonania –** czas wykonania jak program potrzebował na wykonanie wszystkich zadań

3.3 Test

Tabela 3.2 Porównanie algorytmów FCFS i LCFS

Algorytm	czas wykonania	średni czas oczekiwania
FCFS	328.7	149.375
LCFS	328.5	156.8



Wykres 3.2 Porównanie algorytmów FCFS i LCFS

3.4 Wnioski

Między algorytmami nie widać zbyt dużej różnicy i wynika to z kilku powodów. Pierwszym z nich jest fakt, że oba algorytmy są bardzo proste i różnią się jedynie tym który proces obsługują. Algorytm FCFS zawsze obsługuje najstarszy proces, natomiast algorytm LCFS zawsze obsługuje najnowszy proces. Algorytm LCFS został dodatkowo zaimplementowany w taki sposób by przerywać wykonywanie pracy nad danym procesem na rzecz innego, nowszego który przyszedł w czasie wykonywania.

Algorytm FCFS miał minimalnie niższy czas oczekiwania od algorytmu LCFS, co sugeruje że dla danych parametrów symulacji oraz wygenerowanych danych był on minimalnie lepszy.

Podsumowując algorytmy LCFS oraz FCFS nie wykazują znaczącej różnicy między uzyskanymi wynikami. Warto pamiętać, że dobranie prawidłowego algorytmu zarządzające taką czynnością jak zarządzanie czasem procesora jest bardzo ważne. Zależy on od wielu czynników takich jak chociażby liczba procesów czy zakres czasu ich wykonania.

4. Algorytmy zastępowania stron

4.1 Parametry Testów

Zostały przeprowadzone 3 testy dla wygenerowanego losowo ciągu:

1. 500 stron o zakresie <1,20> dla 4 ramek o rozmiarach: 3, 5, 10, 30
2. 500 stron o zakresie <1,30> dla 4 ramek o rozmiarach: 3, 5, 10, 30
3. 500 stron o zakresie <1,50> dla 4 ramek o rozmiarach: 3, 5, 10, 30

4.2 Parametry Symulacji

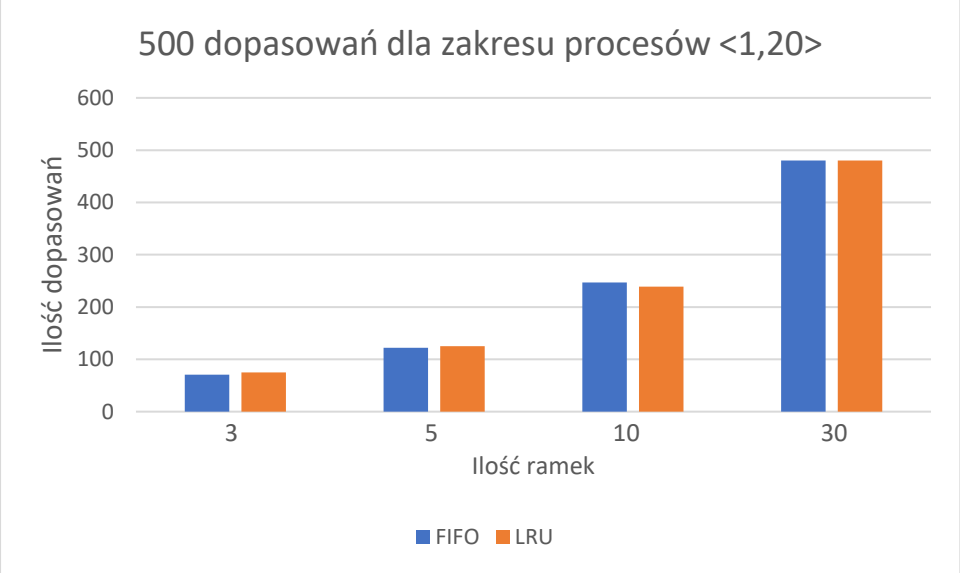
- **Metoda Generowania:** (Pseudo)Losowe liczby z wyszczególnionych w testach przedziałów, takie same pomiędzy różnymi algorytmami dla tych samych danych
- **Liczba stron:** Liczba stron ustawiona na 50
- **Ilość ramek:** 4 rozmiary ramek: 3,5,10,30

4.3 Test 1

500 stron o zakresie <1,20> dla 4 ramek o rozmiarach: 3, 5, 10, 30

Tabela 4.1 Ilość Dopasowań

500 stron – zakres <1,20>	Dopasowań	
ramek	FIFO	LRU
3	71	75
5	122	125
10	247	239
30	480	480



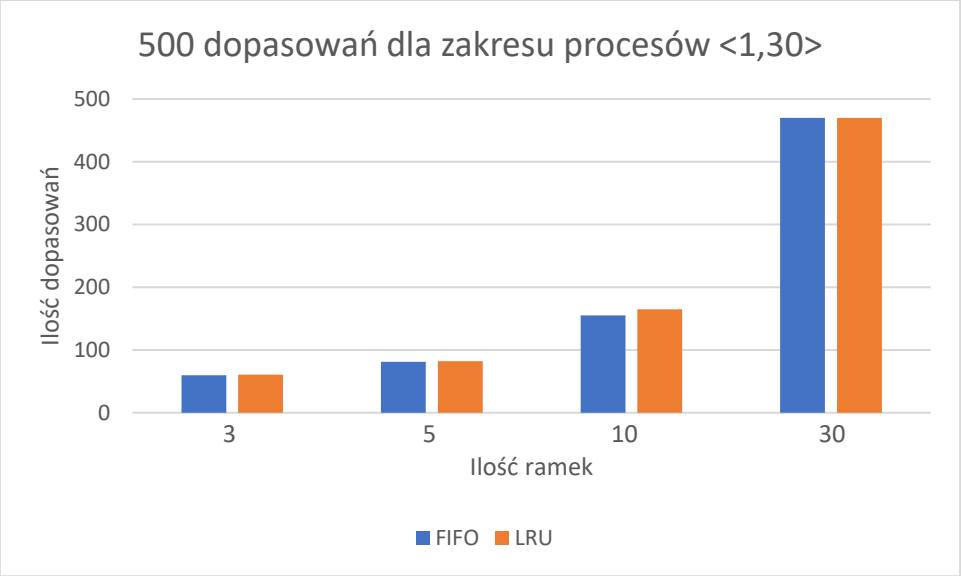
Wykres 4.1 Ilość dopasowań dla różnych wielkości ramki

4.4 Test 2

500 stron o zakresie <1,30> dla 4 ramek o rozmiarach: 3, 5, 10, 30

5. Tabela 4.2 Ilość Dopasowań

500 stron - zakres <1,30>	Dopasowań	
ramek	FIFO	LRU
3	60	61
5	81	82
10	155	165
30	470	470



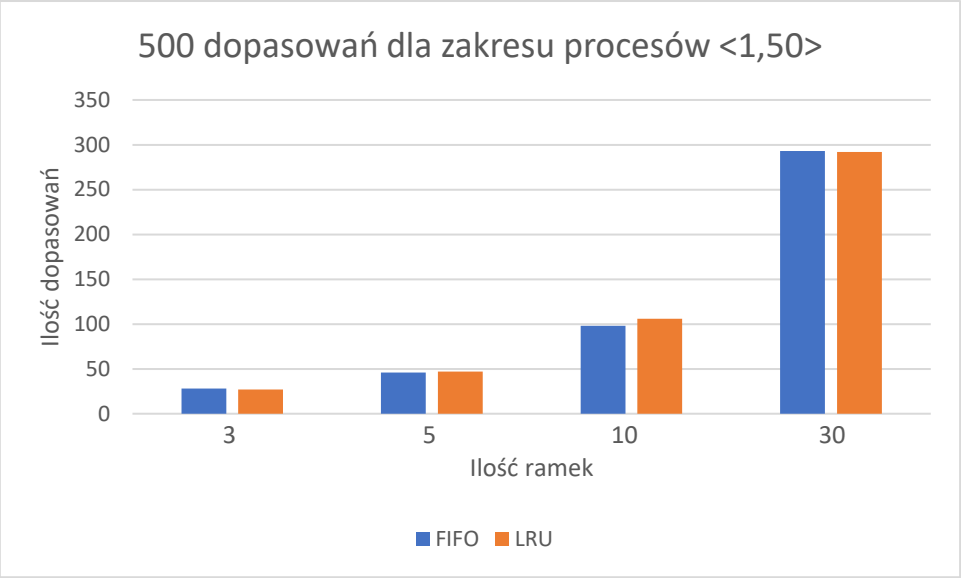
Wykres 4.2 Ilość dopasowań dla różnych wielkości ramki

4.5 Test 2

500 stron o zakresie <1,50> dla 4 ramek o rozmiarach: 3, 5, 10, 30

Tabela 4.3 Ilość Dopasowań

500 stron - zakres <1,50>	Dopasowań	
ramek	FIFO	LRU
3	28	27
5	46	47
10	98	106
30	293	292



Wykres 4.3 Ilość dopasowań dla różnych wielkości ramki

4.6 Wnioski:

W obu algorytmach widać wzrost liczby dopasowani (oraz co za tym idzie spadek liczby błędów) wraz ze wzrostem liczby ramek, a nawet w skrajnych przypadkach gdy zakres losowanych stron jest mniejszy od długości ramki widzimy maksymalną możliwą ilość dopasowań. Są to oczekiwane wyniki potwierdzające, że wraz ze wzrostem ramek ilość dopasowań rośnie dla takich samych zakresów. Wynika to z faktu, że możemy przechować więcej stron w pamięci.

Między algorytmami nie widać zbyt dużej różnicy i wynika to z kilku powodów. Pierwszym z nich jest fakt, że dane są losowe co znaczy że algorytm LRU nie miał dużej przewagi nad algorytmem FIFO. Dzieję się tak ponieważ, zakresy stron nie są losowane z zakresu normalnego gdzie pewne strony pojawiają się częściej od innych podobnie jak działoby się w rzeczywistych urządzeniach. W sytuacji gdy pewne strony pojawiają się częściej od innych LRU miałby wyraźniejszą przewagę. Jednak nie zmienia to faktu, że LRU w powyższych testach wypadł minimalnie lepiej od FIFO.

Warto jednak rozważyć, że w środowisku gdzie zakres stron jest losowany (a nie wynika z rozkładu normalnego) algorytm LRU wypada gorzej pod kątem zużytych zasobów, co wpływa

niekorzystnie na jego ostateczną użyteczność w założonych parametrach symulacji.

Podsumowując dla powyższych symulacji algorytm FIFO jest prawdopodobnie lepszym wyborem z powodu mniejszego zużycia pamięci.