Ćw 3 - MemoryManagement

sobota, 23 marca 2024 13:27

Badanie algorytmów zastępowania stron.
I. Wygenerować losowy ciąg n odwołań do stron
II. Dla wygenerowanego ciągu podać ***liczbę braków strony*** dla różnych algorytmów zastępowania stron:

1. FIFO (usuwamy stronę najdłużej przebywającą w pamięci fizycznej)
2. OPT (optymalny - usuwamy stronę, która nie będzie najdłużej używana)
3. LRU (usuwamy stronę, do której najdłużej nie nastąpiło odwołanie)
4. aproksymowany LRU (Last Recently Used)
5. RAND (usuwamy losowo wybraną stronę)

Wskazówki:

- Symulacje przeprowadzić (koniecznie na tym samym ciągu testowym) dla różnej liczby ramek, np. kilku (3, 5, 10?) wartości podanych przez użytkownika. Wymagana możliwość sterowania parametrami symulacji.
- Należy samodzielnie sformułować założenia symulacji:
  - rozmiar pamięci wirtualnej (ilość stron),
  - rozmiar pamięci fizycznej (ilość ramek),
  - długość (powinna być znaczna - min. 1000) i sposób generowania ciągu odwołań do stron (koniecznie uwzględnić zasadę lokalności odwołań).

1) **FIFO** – usuwana jest strona najdłużej przebywająca w pamięci
2) **OPT** – algorytm optymalny – usuwamy stronę, która najdłużej nie będzie używana; błąd strony będzie występował najrzadziej jak to możliwe; wada – algorytm wymaga wiedzy o przyszłym ciągu odniesień
3) **LRU** – last recently used – usuwamy stronę, która najdłużej nie była wykorzystywana, trzeba znać przeszłość – dane o przeszłości muszą być zapisane
4) **ALRU** – aproksymowany LRU (algorytm drugiej szansy) – zminimalizowana ilość danych, 1 bit – bit odniesienia; w momencie, gdy wykonywany jest kod strony to bit ustawia się na 1, gdy występuje błąd strony, szukamy strony do usunięcia – gdy trafimy na stronę z bitem 1, zmieniamy go na 0, natomiast gdy trafimy na stronę z 0 – usuwamy ją.

FIFO

W tym algorytmie w przypadku braku strony zastępowanie stron odbywa się na zasadzie kolejki (kolejkę tworzą strony, które znajdują się w pamięci fizycznej), tzn. zastępowana jest strona, która najdłużej przebywa w pamięci (jest na początku kolejki). Strona, która została ostatnio dodana znajduje się na końcu kolejki. Algorytm ten nie bierze pod uwagę właściwości obsługiwanych procesów, jest natomiast prosty w implementacji.

Chwila	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Odwołanie	6	3	1	4	2	2	2	3	5	3	3	6	1	6	1
I	**6_1**	6_1	6_1	6_1	**2_4**	2_4	2_4	2_4	2_3	2_2	2_2	2_1	**1_4**	1_4	1_4
II		**3_2**	3_2	3_2	3_1	3_1	3_1	3_1	**5_4**	5_3	5_3	5_2	5_1	5_1	5_1
III			**1_3**	1_3	1_2	1_2	1_2	1_2	1_1	**3_4**	3_4	3_3	3_2	3_2	3_2
IV				**4_4**	4_3	4_3	4_3	4_3	4_2	4_1	4_1	**6_4**	6_3	6_3	6_3
	X	X	X	X	X				X	X		X	X		

OPT

Algorytm optymalny - najefektywniejszy algorytm zastępowania stron, lecz niemożliwy do zaimplementowania w systemie operacyjnym. Polega na usunięciu strony, która najdłużej nie będzie używana. Czas ten oczywiście nie jest systemowi znany, co uniemożliwia napisanie algorytmu. OPT jest algorytmem teoretycznym, który służy do oceniania innych algorytmów.

Chwila	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Odwołanie	6	3	1	4	2	2	2	3	5	3	3	6	1	6	1
I	**6_{12}**	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{12}	6_{14}	6	6
II		**3_8**	3_8	3_8	3_8	3_8	3_8	3_{10}	3_{10}	3_{11}	3	3	3	3	3
III			**1_{13}**	1_{13}	1_{13}	1_{13}	1_{13}	1_{13}	1_{13}	1_{13}	1_{13}	1_{13}	1_{15}	1_{15}	1
IV				**4**	**2_6**	2_7	2	2	**5**	5	5	5	5	5	5
	X	X	X	X	X				X						

LRU

Algorytm LRU (least recently used) działa podobnie do algorytmu optymalnego, lecz usuwamy stronę która nie była używana przez najdłuższy okres. Zakładamy, że odwołania do stron przez proces będą symetryczne. Patrzymy na to, do jakich stron procesu odwoływały się w przeszłości i zakładamy, że w przyszłości będzie działo się podobnie. Algorytm jest szybszy od FIFO, lecz jest trudniejszy do zaimplementowania. Każda strona potrzebuje zawierać informację o tym, kiedy nastąpiło ostatnie odwołanie do niej. Trudna implementacja LRU prowadzi do tego, że często przybliża się tem algorytm za pomocą innego algorytmu

Chwila	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Odwołanie	6	3	1	4	2	2	2	3	5	3	3	6	1	6	1
I	**6_1**	6_1	6_1	6_1	**2_5**	2_6	2_7	2_7	2_7	2_7	2_7	2_7	**1_{13}**	1_{13}	1_{15}
II		**3_2**	3_2	3_2	3_2	3_2	3_2	3_8	3_8	3_{10}	3_{11}	3_{11}	3_{11}	3_{11}	3_{11}
III			**1_3**	1_3	1_3	1_3	1_3	1_3	**5_9**	5_9	5_9	5_9	5_9	5_9	5_9
IV				**4_4**	4_4	4_4	4_4	4_4	4_4	4_4	4_4	**6_{12}**	6_{12}	6_{12}	6_{12}
	X	X	X	X	X				X			X	X		

Algorytm drugiej szansy

Jednym z przybliżeń LRU jest tzw. algorytm drugiej szansy (Second Chance Algorithm). Bardzo przypomina FIFO, jednak drobna zmiana prowadzi do przybliżania LRU. To przybliżenie polega na przypisaniu każdej ramce bitu odwołania. Gdy sprowadzamy nową stronę lub odwołujemy się do strony, która jest już w pamięci to bit ustawiany jest na 1. Gdy potrzebujemy usunąć stronę z pamięci, a ma ona bit odwołania 1, to dostaje ona “drugą szansę”. Bit zmieniany jest na 0 i brana jest kolejna ramka, zgodnie z FIFO.

Chwila	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Odwołanie	6	3	1	4	2	2	2	3	5	3	3	6	1	6	1
I	**6_1**	6_1	6_1	6_1	**2_1**	2_1	2_1	2_1	2_1	2_1	2_1	2_1	**1_1**	1_1	1_1
II		**3_1**	3_1	3_1	3_0	3_0	3_0	3_1	3_0	3_1	3_1	3_1	3_0	3_0	3_0
III			**1_1**	1_1	1_0	1_0	1_0	1_0	**5_1**	5_1	5_1	5_1	5_0	5_0	5_0
IV				**4_1**	4_0	4_0	4_0	4_0	4_0	4_0	4_0	**6_1**	6_0	6_1	6_1
Kolejka	6	6	6	6	3	3	3	3	4	4	4	2	5	5	5
		3	3	3	1	1	1	1	2	2	2	5	3	3	3
			1	1	4	4	4	4	3	3	3	3	6	6	6
				4	2	2	2	2	5	5	5	6	1	1	1
	X	X	X	X	X				X			X	X		

Rand

Zasada działania algorytmu RAND jest bardzo prosta - usuwamy losową stronę z pamięci.

Ze względu na swoją prostotę oraz brak potrzeby przechowywania dodatkowych informacji, algorytm został użyty w procesorach ARM.

Chwila	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Odwołanie	6	3	1	4	2	2	2	3	5	3	3	6	1	6	1
I	**6**	6	6	6	6	6	6	6	6	6	6	6	**1**	1	1
II		**3**	3	3	3	3	3	3	3	3	3	3	3	3	3
III			**1**	1	1	1	1	1	**5**	5	5	5	5	**6**	6
IV				**4**	**2**	2	2	2	2	2	2	2	2	2	2
	X	X	X	X	X				X				X	X	