Mateusz Pater

Pamięć wirtualna - algorytmy zastępowania stron

Spis treści

1. Stronicowanie na żądanie

Stronicowanie na żądanie jest najczęstszym sposobem implementacji pamięci wirtualnej. W skrócie można je określić jako sprowadzanie strony do pamięci operacyjnej tylko wtedy, gdy jest ona potrzebna. Nazywa się to także procedurą leniwej wymiany. Takie rozwiązanie zmniejsza liczbę wykonywanych operacji wejścia wyjścia i zapotrzebowanie na pamięć operacyjną, ponieważ nie sprowadza się niepotrzebnych stron (pojedyncze wywołanie dużego wielofunkcyjnego programu może wymagać sprowadzenia jedynie niewielkiej części jego kodu). Dzięki temu zmniejsza się czas reakcji systemu. Można też obsłużyć większą liczbę użytkowników.
Gdy proces odwołuje się do pamięci, mogą wystąpić trzy sytuacje:

- odwołanie jest niepoprawne,
- odwołanie jest poprawne i strona jest w pamięci, oraz
- odwołanie jest poprawne, ale strony nie ma w pamięci.

W pierwszym przypadku zlecenie jest odrzucane, w drugim jest po prostu obsługiwane, a w trzecim musi dodatkowo obejmować sprowadzanie żądanej strony z dysku do pamięci.

Oto czynności wykonywane w trakcie obsługi braku strony:

1. Sprawdzenie wewnętrznej tablicy (zwykle znajduje się ona w bloku kontrolnym procesu), aby stwierdzić, czy odwołanie do pamięci było poprawne.

2. Znalezienie wolnej ramki w pamięci fizycznej.

3. Sprowadzanie brakującej strony z dysku.

4. Aktualizacja tablicy stron (wstawienie numeru ramki i ustawienie bitu poprawności na 1).

5. Wznowienie instrukcji przerwanej przez wystąpienie braku strony (teraz żądana strona jest już dostępna w pamięci).

Może się jednak tak zdarzyć, że nie ma wolnej ramki w pamięci. Należy wówczas znaleźć jakąś nieużywaną ramkę i przenieść ją na dysk (jeżeli oczywiście na dysku nie ma już jej kopii). Tym zajmuje się **algorytm zastępowania stron**. Należy go oczywiście tak opracować, żeby liczba zastępowanych stron była jak najmniejsza [1].

2. Algorytmy zastępowania stron

Gdy wystąpi brak strony, a nie będzie już wolnych ramek, trzeba wybrać jedną z ramek i zastąpić obecną w niej stronę żądaną stroną. Dzięki zastępowaniu stron pamięć logiczna może być większa niż pamięć fizyczna.

Zastępowanie strony obejmuje następujące czynności:

1. Znalezienie położenia żądanej strony na dysku.

2. Znalezienie strony-ofiary, która ma być usunięta z pamięci.

3. Zapisanie ofiary na dysk – o ile na dysku nie ma jej wiernej kopii.

4. Oznaczenie w tablicy stron odwołania do ofiary jako niepoprawnego.

5. Wczytanie żądanej strony do zwolnionej ramki.

6. Oznaczenie w tablicy stron odwołania do żądanej strony jako poprawnego.

Punkt drugi mówi o znalezieniu strony-ofiary, tą czynnością zajmują się algorytmy zastępowania stron. Jest to jedyne miejsce, gdzie wybór odpowiedniego algorytmu w systemie operacyjnym może wpłynąć na efektywność pamięci wirtualnej. Istnieje wiele takich algorytmów. Najważniejszym kryterium oceny będzie liczba braków stron. Im jest ona mniejsza, tym algorytm lepszy [1].

W dalszych rozdziałach sprawdzimy, jak algorytmy radzą sobie z następującym ciągiem odwołań do stron: 1,2,3,4,1,2,5,1,2,3,4,5

3. Algorytm optymalny

Ofiarą staje się strona, która będzie używana najpóźniej. Jest to algorytm najlepszy, ale niestety tylko teoretyczny. Nie ma możliwości zaimplementowania go w praktyce. Nie da się przecież przewidzieć przyszłych odwołań do pamięci. Służy on do oceny jakości innych algorytmów.

Algorytm dla naszego ciągu odwołań do stron: 1,2,3,4,1,2,5,1,2,3,4,5 daje wynik łącznie 6 zmian stron. Z tym wynikiem będziemy porównywać pozostałe algorytmy.

WYBRANY ALGORYTM: Algorytm optymalny LACZNA LICZBA ZMIAN STRON: 6

Chwila:	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie:	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1:	1	1	1	1	1	1	1	1	1	1	4	4
Ramka 2:	0	2	2	2	2	2	2	2	2	2	2	2
Ramka 3:	0	0	3	3	3	3	3	3	3	3	3	3
Ramka 4:	0	0	0	4	4	4	5	5	5	5	5	5

Rysunek 1. Wynik działania algorytmu optymalnego

4. First-In-First-Out (FIFO)

Strony znajdujące się w pamięci fizycznej tworzą kolejkę FIFO. Każda nowa strona jest przenoszona na koniec kolejki, a ofiarą staje się pierwsza strona na początku kolejki. Zaletą tego algorytmu jest prosta implementacja.

WYBRANY ALGORYTM: First-In-First-Out (FIFO) LACZNA LICZBA ZMIAN STRON: 9

Chwila:	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie:	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1:	1	1	1	4	4	4	5	5	5	5	5	5
Ramka 2:	0	2	2	2	1	1	1	1	1	3	3	3
Ramka 3:	0	0	3	3	3	2	2	2	2	2	4	4

Rysunek 2. Wynik działania algorytmu FIFO

Jak widać, występuje tu 9 braków stron. Jest to wynik zdecydowanie gorszy niż ten z algorytmu optymalnego.

A co, jeśli ramek będzie więcej? Ta liczba wzrasta.

WYBRANY ALGORYTM: First-In-First-Out (FIFO) LACZNA LICZBA ZMIAN STRON: 10

Chwila:	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie:	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1:	1	1	1	1	1	1	5	5	5	5	4	4
Ramka 2:	0	2	2	2	2	2	2	1	1	1	1	5
Ramka 3:	0	0	3	3	3	3	3	3	2	2	2	2
Ramka 4:	0	0	0	4	4	4	4	4	4	3	3	3

Rysunek 3. Wynik działania algorytmu FIFO dla 4 ramek

5. Least Recently Used (LRU)

W tym algorytmie ofiarą staje się strona, która była używana najdawniej. Jest to algorytm oparty na pomyśle z algorytmu optymalnego i, tyle że zamiast przyszłość bierze pod uwagę przeszłość.

WYBRANY ALGORYTM: Least Recently Used (LRU) LACZNA LICZBA ZMIAN STRON: 8

Chwila:	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie:	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1:	1	1	1	1	1	1	1	1	1	1	1	5
Ramka 2:	0	2	2	2	2	2	2	2	2	2	2	2
Ramka 3:	0	0	3	3	3	3	5	5	5	5	4	4
Ramka 4:	0	0	0	4	4	4	4	4	4	3	3	3

Rysunek 4. Wynik działania algorytmu LRU

Wynik w postaci ośmiu braków stron jest już istotnie lepszy od otrzymanego z użyciem algorytmu FIFO.

6. Algorytmy zliczające

Oba algorytmy polegają na zliczaniu liczby odwołań do każdej strony. Nie są one zbyt popularne z powodu kłopotliwej implementacji.

6.1. Least Frequently Used (LFU)

W przypadku **LFU** ofiarą staje się strona z najmniejszą liczbą odwołań. Zakładamy, że nie są to zbyt aktywnie użytkowane strony zatem należy usunąć je z pamięci.

WYBRANY ALGORYTM: Least Frequently Used (LFU) LACZNA LICZBA ZMIAN STRON: 7

Chwila:	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie:	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1:	1	1	1	1	1	1	1	1	1	1	1	1
Ramka 2:	0	2	2	2	2	2	2	2	2	2	2	2
Ramka 3:	0	0	3	3	3	3	5	5	5	3	3	5
Ramka 4:	0	0	0	4	4	4	4	4	4	4	4	4

Rysunek 5. Wynik działania algorytmu LFU

6.2. Most Frequently Used (MFU)

W drugim przypadku sprawa jest odwrotna. Zakładamy, że strony o małych licznikach odwołań zostały właśnie załadowane i wkrótce będą jeszcze używane, zatem ofiarą staje się strona z największą liczbą odwołań.

WYBRANY ALGORYTM: Most Frequently Used (MFU) LACZNA LICZBA ZMIAN STRON: 7

Chwila:	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie:	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1:	1	1	1	1	1	1	5	5	5	5	5	5
Ramka 2:	0	2	2	2	2	2	2	1	2	2	2	2
Ramka 3:	0	0	3	3	3	3	3	3	3	3	3	3
Ramka 4:	0	0	0	4	4	4	4	4	4	4	4	4

Rysunek 6. Wynik działania algorytmu MFU

6.3. Praktyczniej

W praktyce komputer musi w każdej sekundzie wykonać kilkadziesiąt (albo i więcej) tysięcy obliczeń. Sprawdźmy, jak nasze algorytmy różnią się w przypadku większej liczby odniesień. W każdym teście algorytmy korzystają z 4 ramek.

```
WYBRANY ALGORYTM:  First-In-First-Out (FIFO)          LACZNA LICZBA ZMIAN STRON: 1586

WYBRANY ALGORYTM:  Algorytm optymalny          LACZNA LICZBA ZMIAN STRON: 1115

WYBRANY ALGORYTM:  Least Frequently Used (LFU)          LACZNA LICZBA ZMIAN STRON: 1613

WYBRANY ALGORYTM:  Most Frequently Used (MFU)          LACZNA LICZBA ZMIAN STRON: 1600

WYBRANY ALGORYTM:  Least Recently Used (LRU)          LACZNA LICZBA ZMIAN STRON: 1585
```

Rysunek 7. Wynik działania algorytmów dla 2 000 losowych odniesień z przedziału od 0 do 20

```
WYBRANY ALGORYTM:  First-In-First-Out (FIFO)          LACZNA LICZBA ZMIAN STRON: 20104

WYBRANY ALGORYTM:  Algorytm optymalny          LACZNA LICZBA ZMIAN STRON: 14071

WYBRANY ALGORYTM:  Least Frequently Used (LFU)          LACZNA LICZBA ZMIAN STRON: 19999

WYBRANY ALGORYTM:  Most Frequently Used (MFU)          LACZNA LICZBA ZMIAN STRON: 20159

WYBRANY ALGORYTM:  Least Recently Used (LRU)          LACZNA LICZBA ZMIAN STRON: 20115
```

Rysunek 8. Wynik działania algorytmów dla 25 000 odniesień losowych odniesień z przedziału od 0 do 20

Tak na prawdę możemy jedynie potwierdzić, że algorytm optymalny jest znacząco lepszy od pozostałych. Co do reszty algorytmów nie można na podstawie takich testów określić, który z nich jest lepszy. Są to losowe odniesienia, które tak na prawdę nijak mogą się mieć do prawdziwych sytuacji.

```
WYBRANY ALGORYTM:  First-In-First-Out (FIFO)          LACZNA LICZBA ZMIAN STRON: 1938

WYBRANY ALGORYTM:  Algorytm optymalny          LACZNA LICZBA ZMIAN STRON: 1627

WYBRANY ALGORYTM:  Least Frequently Used (LFU)          LACZNA LICZBA ZMIAN STRON: 1920

WYBRANY ALGORYTM:  Most Frequently Used (MFU)          LACZNA LICZBA ZMIAN STRON: 1922

WYBRANY ALGORYTM:  Least Recently Used (LRU)          LACZNA LICZBA ZMIAN STRON: 1935
```

Rysunek 9. Wynik działania algorytmów dla 2 000 losowych odniesień z przedziału od 0 do 100

```
WYBRANY ALGORYTM:  First-In-First-Out (FIFO)          LACZNA LICZBA ZMIAN STRON: 23996

WYBRANY ALGORYTM:  Algorytm optymalny          LACZNA LICZBA ZMIAN STRON: 20314

WYBRANY ALGORYTM:  Least Frequently Used (LFU)          LACZNA LICZBA ZMIAN STRON: 23993

WYBRANY ALGORYTM:  Most Frequently Used (MFU)          LACZNA LICZBA ZMIAN STRON: 23970
```

Rysunek 10. Wynik działania algorytmów dla 25 000 odniesień losowych odniesień z przedziału od 0 do 100

7. Program do symulacji

7.1. Ustawienia

W programie powinniśmy wybrać liczbę ramek oraz uzupełnić tablicę odwołań. Następnie możemy wybrać jeden z pięciu zdefiniowanych algorytmów.

```c
#define CHWIL 12
#define RAMEK 4
int odwolania[] = { 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 };
```

Rysunek 11. Przykładowe uzupełnienie wykorzystanego przy wcześniejszym omawianiu algorytmów

7.2. Kompilowanie

Przed uruchomieniem programu należy go skompilować poleceniem:

```
$ gcc main.c -std=c99
```

Uruchomienie:

```
$ .\a.out
```

```c
#define FIFO 0
#define OPTYMALNY 1
#define LRU 2
#define MFU 3
#define LFU 4

int main()
{
    zastepowanieStron(FIFO);
    zastepowanieStron(OPTYMALNY);
    zastepowanieStron(LFU);
    zastepowanieStron(MFU);
    zastepowanieStron(LRU);

    return 0;
}
```

Rysunek 12. Zdefiniowane algorytmy oraz przykład ich użycia

```
MacBook-Air-Mateusz:Algorytmy pater$ ./a.out


WYBRANY ALGORYTM:  First-In-First-Out (FIFO)          LACZNA LICZBA ZMIAN STRON: 10
Chwila:         1       2       3       4       5       6       7       8       9       10      11      12
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Odwołanie:      1       2       3       4       1       2       5       1       2       3       4       5
Ramka 1:        1       1       1       1       1       1       5       5       5       5       4       4
Ramka 2:        0       2       2       2       2       2       2       1       1       1       1       5
Ramka 3:        0       0       3       3       3       3       3       3       2       2       2       2
Ramka 4:        0       0       0       4       4       4       4       4       4       3       3       3


WYBRANY ALGORYTM:  Algorytm optymalny          LACZNA LICZBA ZMIAN STRON: 6
Chwila:         1       2       3       4       5       6       7       8       9       10      11      12
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Odwołanie:      1       2       3       4       1       2       5       1       2       3       4       5
Ramka 1:        1       1       1       1       1       1       1       1       1       1       4       4
Ramka 2:        0       2       2       2       2       2       2       2       2       2       2       2
Ramka 3:        0       0       3       3       3       3       3       3       3       3       3       3
Ramka 4:        0       0       0       4       4       4       5       5       5       5       5       5


WYBRANY ALGORYTM:  Least Frequently Used (LFU)          LACZNA LICZBA ZMIAN STRON: 7
Chwila:         1       2       3       4       5       6       7       8       9       10      11      12
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Odwołanie:      1       2       3       4       1       2       5       1       2       3       4       5
Ramka 1:        1       1       1       1       1       1       1       1       1       1       1       1
Ramka 2:        0       2       2       2       2       2       2       2       2       2       2       2
Ramka 3:        0       0       3       3       3       3       5       5       5       3       3       5
Ramka 4:        0       0       0       4       4       4       4       4       4       4       4       4
```

Rysunek 13. Uruchamianie programu oraz przykładowy wynik

Literatura

1. Z zasadami działania pamięci wirtualnej można zapoznać się pod adresem *http://edu.pjwstk.edu.pl/wyklady/sop/scb/wyklad8/wyklad.html*