

Algorytmy wymiany stron

Podstawowe pojęcia (1)

- ☼ $\mathcal{N} = \{1, 2, \dots, n\}$ — zbiór numerów stron wirtualnych danego procesu
- ☼ $\mathcal{M} = \{1, 2, \dots, m\}$ — zbiór numerów ramek danego procesu w pamięci fizycznej
- ☼ \mathcal{VA} — wirtualna przestrzeń adresowa,
- ☼ $va \in \mathcal{VA}$ — adres wirtualny
 $va = \langle x, w \rangle = x \cdot c + w$, gdzie c jest rozmiarem strony (ramki)
- ☼ \mathcal{RA} — fizyczna przestrzeń adresowa
- ☼ $ra \in \mathcal{RA}$ — adres fizyczny

Podstawowe pojęcia (2)

$$ra = \begin{cases} f_i(x) \cdot c + w, & \text{gdy } f_i(x) \neq 0 \\ \text{nieokreślony,} & \text{gdy } f_i(x) = 0 \rightarrow \text{błąd strony} \end{cases}$$

$$f_i(x) = \begin{cases} y, & \text{gdy strona jest w ramce } y \text{ w chwili } t \\ 0, & \text{gdy brak strony w pamięci} \end{cases}$$

Podstawowe pojęcia (3)

- ☼ Dynamiczne własności procesu określone są przez ciąg zgłoszeń (odniesień do stron) $\omega = r_1, r_2, \dots, r_p, \dots$, gdzie t jest dyskretną chwilą czasową, a $r_t = x$ ($x \in \mathcal{N}$) oznacza, że odniesienie do strony numer x wystąpiło jako t -te.
- ☼ \mathcal{S}_t — stan pamięci operacyjnej zdefiniowany jako zbiór numerów stron wirtualnych, znajdujących się w ramach pamięci fizycznej w chwili t ($\mathcal{S}_t \subseteq \mathcal{N}$, $|\mathcal{S}_t| \leq m$)
- ☼ Algorytmem wymiany jest przetwarzanie ciągu zgłoszeń $\omega = r_1, r_2, \dots, r_p, \dots$ i generowanie kolejnych stanów pamięci $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_p, \dots$,

Podstawowe pojęcia (4)

- ☼ \mathcal{X}_t — zbiór numerów stron sprowadzonych do pamięci w chwili t ($\mathcal{X}_t \subseteq \mathcal{N} \setminus \mathcal{S}_{t-1}$)
- ☼ \mathcal{Y}_t — zbiór numerów stron usuwanych z pamięci w chwili t ($\mathcal{Y}_t \subseteq \mathcal{S}_{t-1}$)
- ☼ Zmiana stanu pamięci w chwili t w stosunku do chwili $t-1$ wynika z ze sprowadzenia stron o numerach ze zbioru \mathcal{X}_t i usunięciu stron o numerach ze zbioru \mathcal{Y}_t
 $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathcal{X}_t \setminus \mathcal{Y}_t$

Formalna definicja algorytmu wymiany stron

- ☼ Algorytm wymiany A jest zdefiniowany poprzez funkcję przejścia g_A , taką że
 $g_A(\mathcal{S}_{t-1}, \mathcal{Q}_{t-1}, r_t) = (\mathcal{S}_t, \mathcal{Q}_t)$, gdzie
 - \mathcal{Q}_t — stan wymiany w chwili t , reprezentujący historię wymiany
 - \mathcal{S}_t — stan pamięci operacyjnej w chwili t
 - r_t — numer strony, do której nastąpiło odniesienie w chwili t
- ☼ $(\mathcal{S}_t, \mathcal{Q}_t)$ nazywana jest konfiguracją algorytmu wymiany
- ☼ Algorytm sprowadzania A_F w chwili t określa zbiór \mathcal{X}_t
- ☼ Algorytm usuwania A_R w chwili t określa zbiór \mathcal{Y}_t

Sprowadzanie i usuwanie na żądanie (1)

- Algorytm sprowadzania jest *algorytmem sprowadzania na żądanie* A_{DF} , gdy sprowadzenie strony następuje tylko przy wystąpieniu odniesienia do niej i tylko wówczas, gdy nie ma jej w pamięci.
- Algorytm usuwania jest *algorytmem usuwania na żądanie* A_{DR} , gdy usunięcie strony z pamięci następuje w przypadku braku wolnych ramek i wynika z konieczności zwolnienia ramki na potrzeby innej sprowadzanej strony.

Sprowadzanie i usuwanie na żądanie (2)

- Algorytm sprowadzania na żądanie A_{DF} :

$$\forall_{i \geq 1} \forall_{m \geq 1} (|\mathcal{X}_i| \in \{0, 1\} \wedge (r_i \in \mathcal{S}_{i-1} \Rightarrow \mathcal{X}_i = \emptyset))$$

- Algorytm usuwania na żądanie A_{DR} :

$$\forall_{i \geq 1} \forall_{m \geq 1} (|\mathcal{Y}_i| \in \{0, 1\} \wedge ((r_i \in \mathcal{S}_{i-1} \vee |\mathcal{S}_{i-1}| < m) \Rightarrow \mathcal{Y}_i = \emptyset))$$

Klasyfikacja algorytmów wymiany

- Algorytmy wymiany na żądanie A_{DFDR} — zarówno sprowadzanie stron, jak i ich usuwanie odbywa się na żądanie.
- Algorytmy wymiany ze sprowadzaniem na żądanie A_{DFR} — tylko sprowadzanie stron odbywa się na żądanie, ich usuwanie wykonywane jest przez dowolny algorytm.
- Algorytmy wymiany ze wstępnym stronicowaniem na żądanie A_{DP} — w wyniku wystąpienia błędu strony sprowadzany do pamięci jest zbiór stron (w tym strona żądana).

Algorytm wymiany na żądanie A_{DFDR}

- Funkcja przejścia $g_{A_{DFDR}}$ spełnia w każdej chwili t następujące warunki:

$$\mathcal{S}_t = \begin{cases} \mathcal{S}_{t-1} & \text{jeśli } r_t \in \mathcal{S}_{t-1} \\ \mathcal{S}_{t-1} \cup \{r_t\} & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \wedge |\mathcal{S}_{t-1}| < m \\ \mathcal{S}_{t-1} \cup \{r_t\} \setminus \{y\} & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \wedge |\mathcal{S}_{t-1}| = m \end{cases}$$

y — numer strony wirtualnej usuwanej z pamięci

Algorytm wymiany ze sprowadzaniem na żądanie A_{DFR}

- Funkcja przejścia $g_{A_{DFR}}$ spełnia w każdej chwili t następujące warunki:

$$\mathcal{S}_t = \begin{cases} \mathcal{S}_{t-1} \setminus \mathcal{Y}_t & \text{jeśli } r_t \in \mathcal{S}_{t-1} \wedge r_t \notin \mathcal{Y}_t \\ \mathcal{S}_{t-1} \cup \{r_t\} \setminus \mathcal{Y}_t & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \wedge |\mathcal{S}_{t-1}| < m \\ \mathcal{S}_{t-1} \cup \{r_t\} \setminus \mathcal{Y}_t \wedge \mathcal{Y}_t \neq \emptyset & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \wedge |\mathcal{S}_{t-1}| = m \end{cases}$$

Algorytm wymiany ze wstępnym stronicowaniem na żądanie A_{DP}

- Funkcja przejścia $g_{A_{DP}}$ spełnia w każdej chwili t następujące warunki:

$$\mathcal{S}_t = \begin{cases} \mathcal{S}_{t-1} & \text{jeśli } r_t \in \mathcal{S}_{t-1} \\ \mathcal{S}_{t-1} \cup \mathcal{X}_t & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \wedge r_t \in \mathcal{X}_t \wedge |\mathcal{S}_{t-1} \cup \mathcal{X}_t| \leq m \\ \mathcal{S}_{t-1} \cup \mathcal{X}_t \setminus \mathcal{Y}_t & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \wedge r_t \in \mathcal{X}_t \wedge |\mathcal{S}_{t-1} \cup \mathcal{X}_t \setminus \mathcal{Y}_t| = m \end{cases}$$

Koszt wymiany stron (1)

- ☼ Z punktu widzenia efektywności przetwarzania koszt wymiany możemy utożsamiać z czasem realizacji wymiany.
- ☼ W celu uproszczenia analizy możemy przyjąć, że koszt usunięcia strony stanowi stałą część kosztu jej sprowadzenia.
- ☼ Koszt wynika zatem z czasu sprowadzania, który w ogólności zależy od ciągu zgłoszeń, liczby dostępnych ramek i algorytmu wymiany — $C(A, m, \omega)$

Koszt wymiany stron (2)

- ☼ Niech $h(k)$ oznacza koszt jednorazowego sprowadzenia grupy k stron, przy czym: $h(0) = 0, h(1) = 1$.
- ☼ Całkowity koszt sprowadzenia stron i tym samym postać funkcji $C(A, m, \omega)$ w przypadku ogólnym dana jest wzorem:

$$C(A, m, \omega) = \sum_t h(|\mathcal{X}_t|)$$

Koszt wymiany stron (3)

- ☼ W przypadku zastosowania dysku jako urządzenia wymiany, na czas sprowadzania wpływ mają:
 - T_w — czas oczekiwania (suma czasu oczekiwania w kolejce do urządzenia oraz czasu przygotowania urządzenia do transmisji)
 - T_{tr} — czas transmisji danych
- ☼ Postać funkcji $h(k)$ jest zatem następująca:
 $h(k) = T_w + k \cdot T_{tr}$

Koszt wymiany stron (4)

- ☼ Własność funkcji $h(k)$ przy założeniu, że $T_w > 0$:
 $h(k) < k \cdot (T_w + T_{tr})$
- ☼ Wniosek: sprowadzenie kilku stron jako wyniku realizacji jednego żądania jest mniej kosztowne niż sprowadzanie poszczególnych stron osobno.

Koszt wymiany stron (5)

- ☼ Koszt wymiany można aproksymować za pomocą parametrów FN (liczba wygenerowanych błędów strony) i TN (liczba transmisji stron).
- ☼ Całkowity czas oczekiwania na urządzenie (istotna składowa czasu wymiany) również jest zależny od liczby błędów strony.
- ☼ Wniosek: należy minimalizować liczbę błędów strony i tym samym redukować czas realizacji wymiany, czyli koszt.

Klasyfik. alg. wymiany ze wzgl. na sposób zastępowania stron

- ☼ Zastępowanie lokalne (ang. local replacement) — algorytm wymiany zastępuje tylko strony w ramach przydzielonych procesowi, który spowodował błąd strony → stronicowanie statyczne.
- ☼ Zastępowanie globalne (ang. global replacement) — algorytm wymiany zastępuje strony znajdujące się w dostępnej puli ramek w całym systemie (w szczególności zatem usuwa strony innych procesów) → stronicowanie dynamiczne.

Algorytmy wymiany na żądanie — definicje pomocnicze

- ☼ Dystans w przód:
$$d_t(x) = \begin{cases} k & \text{gdy } r_{t+k} = x \wedge \bigvee_{0 < i < k} r_{t+i} \neq x \\ \infty & \text{gdy } \bigvee_{0 < i} r_{t+i} \neq x \end{cases}$$
- ☼ Dystans w tył:
$$b_t(x) = \begin{cases} k & \text{gdy } r_{t-k} = x \wedge \bigvee_{0 < i < k} r_{t-i} \neq x \\ \infty & \text{gdy } \bigvee_{i < t} r_i \neq x \end{cases}$$
- ☼ Chwila ostatniego sprowadzenia do pamięci:
$$g_t(x) = \max \{i : i \leq t \wedge x \in \mathcal{X}_i\}$$

Algorytmy wymiany na żądanie — własności

- ☼ Strona x sprowadzana jest w chwili t tylko wówczas, gdy $r_t = x$ i $x \notin \mathcal{S}_{t-1}$
- ☼ Strona y usuwana jest w chwili t tylko wówczas, gdy $|\mathcal{S}_{t-1}| = m$ i $r_t \notin \mathcal{S}_{t-1}$
- ☼ Zachodzi warunek $\bigvee_t 0 \leq |\mathcal{X}_t| \leq 1$
- ☼ O specyfice algorytmu wymiany na żądanie decyduje algorytm usuwania stron.

Algorytmy wymiany na żądanie — funkcja kosztu

- ☼ Funkcja kosztu sprowadza się do następującej postaci

$$C(A, m, \omega) = \sum_t |\mathcal{X}_t|$$

Algorytmy wymiany na żądanie — MIN

- ☼ Wybór strony usuwanej w chwili t :
$$y = \min \{z \in \mathcal{S}_{t-1} : d_t(z) = \max_{u \in \mathcal{S}_{t-1}} \{d_t(u)\}\}$$

Twierdzenie: w klasie algorytmów wymiany na żądanie algorytm MIN minimalizuje koszt $C(A, m, \omega)$ dla każdego m i ω .

Algorytmy wymiany na żądanie — LRU (Least Recently Used)

- ☼ Wybór strony usuwanej w chwili t : usuwana jest strona y , dla której
$$b_t(y) = \max_{u \in \mathcal{S}_{t-1}} \{b_t(u)\}$$
- ☼ Algorytm daje dobre wyniki w przypadku procesów wykazujących lokalność czasową odwołań do pamięci, gdyż dystans w tył dobrze przybliża dystans w przód.

Alg. wymiany na żąd.— LFU/MFU (Least/Most Frequently Used)

- ☼ Niech $\text{FREQ}_t(x)$ oznacza liczbę odniesień do strony x w ciągu:
 - r_1, r_2, \dots, r_{t-1} lub
 - $r_k, r_{k+1}, \dots, r_{t-1}$, gdzie $k = g_t(x)$
- ☼ Wybór strony usuwanej w chwili t : usuwana jest strona y , dla której
 - LFU: $\text{FREQ}_t(y) = \min_{u \in \mathcal{S}_{t-1}} \{\text{FREQ}_t(u)\}$
 - MFU: $\text{FREQ}_t(y) = \max_{u \in \mathcal{S}_{t-1}} \{\text{FREQ}_t(u)\}$

Algorytmy wymiany na żądanie — FIFO (ang. First In First Out)

- Wybór strony usuwanej w chwili t : usuwana jest strona y , dla której

$$g_t(y) = \min_{u \in S_{t-1}} \{g_t(u)\}$$

- Algorytm daje dobre wyniki w przypadku programów o charakterze sekwencyjnym, z rzadką zmianą przepływu sterowania (mało pętli, wywołań procedur)

Algorytmy wymiany na żądanie — LIFO (ang. Last In First Out)

- Wybór strony usuwanej w chwili t : usuwana jest strona y , dla której

$$g_t(y) = \max_{u \in S_{t-1}} \{g_t(u)\}$$

- Algorytm daje dobre wyniki w przypadku realizacji fragmentu programu zawierającego pętlę, gdyż ponowne wykonanie tej samej instrukcji nastąpi dopiero po wykonaniu całej kolejnej iteracji.

Algorytmy wymiany na żądanie — LD (ang. Loop Detection)

- Definicja czasu biernego l_t : niech $k = t - b_t(z)$

$$l_t(z) = b_k(z)$$

- Wybór strony usuwanej w chwili t : usuwana jest strona y , dla której

moment ostatniego odniesienia do strony w przeszłości

$$\begin{cases} b_{t-1}(y) = \max_{z \in S_{t-1}} \{b_{t-1}(z)\} & \text{jeśli } \forall_{z \in S_{t-1}} b_{t-1}(z) > l_{t-1}(z) \\ l_{t-1}(y) - b_{t-1}(y) = \max_{z \in S_{t-1}} \{l_{t-1}(z) - b_{t-1}(z)\} & \text{jeśli} \\ & \exists_{z \in S_{t-1}} b_{t-1}(z) \leq l_{t-1}(z) \end{cases}$$

Zagadnienia implementacyjne

- Implementacja algorytmu FIFO
- Implementacja algorytmu LRU
- Algorytmy przybliżające metodę LRU
 - algorytm drugiej szansy (FINUFO)
 - algorytm dodatkowych bitów odwołań
 - ulepszony algorytm drugiej szansy
- Implementacja algorytmów licznikowych

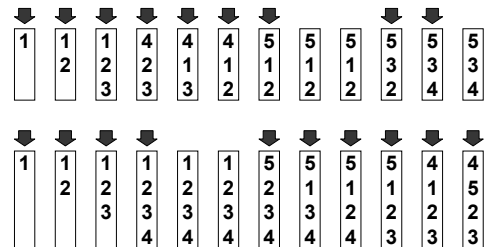
Implementacja algorytmu FIFO

- Utrzymywanie listy numerów stron w kolejności ich sprowadzania do pamięci
- Umieszczanie numeru sprowadzanej strony na końcu listy
- Usuwanie z pamięci (i z listy) strony, której numer znajduje się na początku listy
- Możliwość wystąpienia anomalii Belady'ego

dla 3 ramek mamy 9 błędów strony, a dla 4 ramek mamy 10 błędów strony

Przykład anomalii Belady'ego

- Ciąg odniesień: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



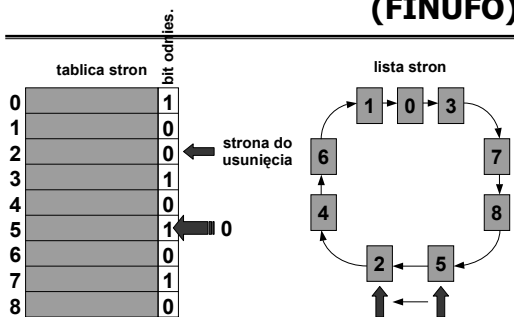
Implementacja algorytmu LRU

- ☼ Licznik — przed każdym odwołaniem do pamięci zwiększana jest wartość pewnego licznika i wpisywana do odpowiedniej pozycji opisującej stronę w tablicy stron (lub w innej specjalnej strukturze systemu operacyjnego). Z pamięci usuwana jest wówczas strona z najmniejszą wartością tego licznika, co wymaga przejrzenia całej tablicy stron.
- ☼ Stos — numery stron, do których następuje odwołanie, odkładane są na szczycie stosu. Przed odłożeniem na szczycie numer strony musi być wydobyty ze środka stosu, czyli z miejsca, gdzie był ostatnio odłożony. W tej implementacji pamięci usuwana jest strona z dna stosu.

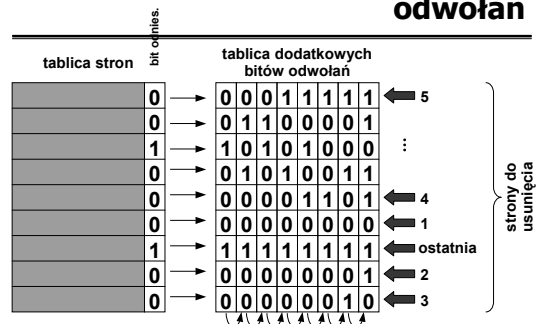
Algorytmy przybliżające metodę LRU

- ☼ Niezbędne wspomaganie sprzętowe:
 - bit odniesienia (ang. reference bit) — ustawiany dla danej strony przez sprzęt zawsze, gdy następuje **zapis lub odczyt** na tej stronie,
 - bit modyfikacji (ang. modify bit) — ustawiany dla danej strony przez sprzęt zawsze, gdy następuje **zapis** na tej stronie.
- ☼ Algorytmy korzystające ze wspomagania sprzętowego:
 - algorytm drugiej szansy — wykorzystuje bit odniesienia,
 - algorytm dodatkowych bitów odwołań — wykorzystuje bit odniesienia,
 - ulepszony algorytm drugiej szansy — wykorzystuje bit odniesienia i bit modyfikacji.

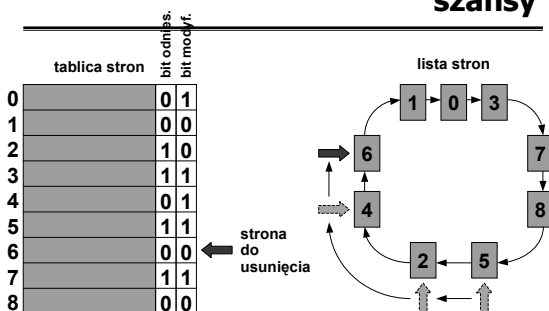
Algorytm drugiej szansy (FINUFO)



Algorytm dodatkowych bitów odwołań



Ulepszony algorytm drugiej szansy



Implementacja algorytmów LFU i MFU

- ☼ Utrzymywanie liczników odniesień do każdej ze stron i ich inkrementacja przy każdym odwołaniu do odpowiedniej strony
- ☼ Usuwanie strony z najmniejszą (LFU) lub największą (MFU) wartością licznika
- ☼ Zerowanie licznika po usunięciu strony z pamięci

Dobór liczby ramek

- Minimalna liczba ramek — zdefiniowana przez architekturę komputera (zależna od maksymalnej liczby komórek adresowanych przez jeden rozkaz).
- Liczba ramek przydzielona dla procesu
 - podział równomierny (ang. equal allocation)
 - podział proporcjonalny (ang. proportional allocation)
 - przydział zależny od priorytetu procesu

Przydział ramek

- Stronicowanie statyczne
 - liczba ramek jest ustalona na cały czas cyklu przetwarzania procesu,
- Stronicowanie dynamiczne — liczba przydzielonych ramek jest zmienna.
 - liczba ramek przydzielonych procesowi zmienia się w czasie w zależności od potrzeb samego procesu oraz działań i potrzeb innych procesów w systemie.

Problem stronicowania statycznego

- Niech f_i, f_j i \bar{f} oznaczają średnią częstość błędów strony odpowiednio procesu P_i, P_j oraz całego systemu.
- Niech m_i i m_j oznacza liczbę ramek odpowiednio procesu P_i, P_j .
- Niech f'_i oznacza średnią częstość błędów strony procesu P_i przy liczbie ramek m'_i
- $f_j > \bar{f} \wedge (m'_i < m_i \not\Rightarrow f'_i > f_i)$

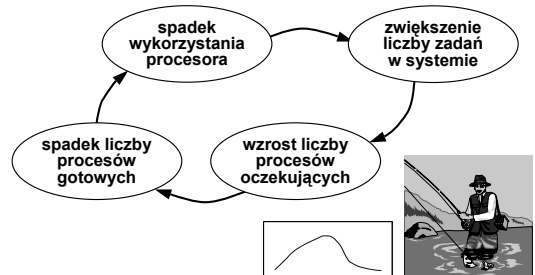
Zjawisko szamotania (migotania) stron (ang. trashing)

- Szamotanie oznacza istotny wzrost częstości generowania błędów strony, wynikający z niemożności utrzymania w pamięci stron aktywnych.
- Przyczyna szamotania: zbyt mała liczba dostępnych ramek.

Szamotanie w przypadku stronicowania statycznego

- Założenie: program wykazuje lokalność przestrzenną odwołań do pamięci.
- Model strefowy (ang. locality model)
 - strefa — zbiór stron aktywnych w czasie wykonywania określonego fragmentu programu,
 - wykonanie programu sprowadza się do przechodzenia z jednej strefy do drugiej,
 - program składa się z wielu stref, najczęściej zachodzących na siebie.
- Przyczyna szamotania: liczba dostępnych ramek jest mniejsza od rozmiaru strefy.

Szamotanie w przypadku stronicowania dynamicznego



Zapobieganie szamotaniu

- W systemie stronicowania dynamicznego szamotanie można **ograniczyć** przez zastosowanie algorytmu zastępowania lokalnego lub priorytetowego (nie likwiduje to jednak problemu!!!).
- Szamotanie można zlikwidować, dostarczając procesowi tylu wolnych ramek ilu potrzebuje do wykonania określonego fragmentu programu (np. procedury, pętli).

Jak się dowiedzieć ile wolnych ramek jest potrzebnych procesowi?

Alg. wymiany ze sprowadzaniem na żąd. — dodatkowe oznaczenia

- \bar{f} — średnia częstość błędów strony
- c_{pf} — koszt obsługi błędu strony
- c_u — koszt przechowywania strony w pamięci przez jednostkę czasu wirtualnego

Alg. wymiany ze sprowadzaniem na żądanie — własności

- Strona x sprowadzana jest w chwili t tylko wówczas, gdy $r_t = x$ i $x \notin S_{t-1}$
- Możliwe jest usunięcie strony nawet w przypadku odniesienia, które nie powoduje błędu strony
- Możliwy jest warunek: $|\mathcal{Y}_t| > |\mathcal{X}_t|$ ($|\mathcal{X}_t| \leq 1$)

Alg. wymiany ze sprow. na żąd. — VMIN (ang. Variable-space MIN)

- Wybór strony usuwanej w chwili t : strona usuwana jest natychmiast po jej przetworzeniu i tylko wówczas, gdy nie będzie ona wykorzystywana przez czas dłuższy niż c_{pf}/c_u jednostek czasu wirtualnego

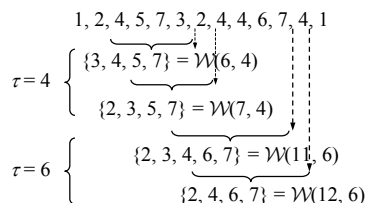
$$\mathcal{Y}_t = \begin{cases} \{r_{t-1}\} & \text{jeśli } d_t(y) > \frac{c_{pf}}{c_u} \\ \emptyset & \text{inaczej} \end{cases}$$

Twierdzenie: w klasie algorytmów wymiany ze sprowadzaniem na żądanie, przy określonym średnim rozmiarze przydzielonej procesowi pamięci operacyjnej m_w , algorytm VMIN minimalizuje średnią częstość generowania błędów strony.

Zbiór roboczy — definicja

- Zbiór roboczy procesu (ang. working-set) — zbiór stron, do których nastąpiło odwołanie w ciągu ostatnich τ odwołań (tzw. oknie zbioru roboczego) w czasie działania procesu
- Okno zbioru roboczego (ang. working-set window) — liczba odwołań do pamięci, przez które dana strona pozostaje w zbiorze roboczym od ostatniego odwołania do niej

Zbiór roboczy — przykład



Ustalanie liczby potrzebnych ramek

- ☼ Zapotrzebowanie na ramki w całym systemie jest sumą rozmiarów zbiorów roboczych, czyli $Z = \sum |\mathcal{W}_i|$, gdzie $|\mathcal{W}_i|$ jest mocą (liczbą elementów) zbioru \mathcal{W}_i , czyli zapotrzebowaniem i -tego procesu na ramki.
- ☼ Jeśli $Z > m$, gdzie m jest całkowitą liczbą ramek w systemie dostępną dla procesów użytkownika, to zapotrzebowanie jest zbyt duże i może powstać szamotanie. System operacyjny musi wówczas wstrzymać wykonywanie jakiegos procesu i zwolnić zajmowane przez niego ramki
- ☼ Jeśli w systemie są wolne ramki ($Z < m$), to można rozpocząć nowy proces.

Alg. wymiany ze sprow. na żąd. — WS (ang. Working Set)

- ☼ Formalna definicja zbioru roboczego $\mathcal{W}(t, \tau)$
$$\mathcal{W}(t, \tau) = \left\{ z : \exists_{0 \leq i < \tau} r_{t-i} = z \right\} = \{ z : b_i(z) < \tau \}$$
- ☼ Wybór strony usuwanej w chwili t : usuwana jest strona, która nie była używana w ostatnich τ jednostkach czasu wirtualnego
$$\mathcal{Y}_t = \{ y : y \in \mathcal{S}_{t-1} \wedge y \notin \mathcal{W}(t, \tau) \} = \mathcal{S}_{t-1} \setminus \mathcal{W}(t, \tau)$$
- ☼ Problem: jak dobrać rozmiar okna zbioru roboczego τ ?

Alg. wymiany ze sprow. na żąd. — WS (ang. Working Set)

- ☼ Jak należałoby zinterpretować fakt wystąpienia problemu zastępowania (brak wolnej ramki) w przypadku obsługi błędu strony?



Alg. wymiany ze sprow. na żąd. — DWS (ang. Damped Working Set)

- ☼ Wybór strony usuwanej w chwili t : usuwana jest strona, która nie była używana w ostatnich τ jednostkach czasu wirtualnego lub w przypadku wystąpienia błędu strony strona, które nie była wykorzystywana przez więcej niż $\alpha\tau$ jednostek czasu wirtualnego ($\alpha < 1$).

$$\mathcal{Y}_t = \begin{cases} \mathcal{S}_{t-1} \setminus \mathcal{W}(t, \tau) & \text{jeśli } r_t \in \mathcal{S}_{t-1} \\ \left\{ y : b_i(y) = \max_{z \in \mathcal{S}_{t-1}} \{ b_i(z) \} \wedge b_i(y) > \alpha\tau \right\} & \text{jeśli } r_t \notin \mathcal{S}_{t-1} \end{cases}$$

Koncepcja identyfikacji zbioru roboczego

- ☼ Dla każdej ramki utrzymywany jest wirtualny czas ostatniego odniesienia do niej.
- ☼ Po każdym odniesieniu do strony zwiększana jest wartość licznika odniesień i wpisywana do odpowiedniej tablicy na pozycji odpowiadającej ramce, w której znajdują się adresowana strona.
- ☼ Do zbioru roboczego należą te strony, dla których różnica pomiędzy bieżącą wartością licznika odniesień, a wartością wpisaną w tablicy jest mniejsza lub równa rozmiarowi okna zbioru rob.

Problemy zastosowania koncepcji zbioru roboczego

- ☼ Kwestia zasadności przesłanki o lokalności czasowej odniesień do pamięci
- ☼ Problem precyzyjnej identyfikacji zbioru roboczego:
 - monitorowanie odniesień do pamięci
 - utrzymywanie odpowiednich struktur danych
- ☼ Problem doboru rozmiaru okna zbioru roboczego

Problemy identyfikacji zbioru roboczego

- ☼ Problem monitorowania odniesień do stron
 - zbyt duży koszt (narzut czasowy) dostępu do pamięci w przypadku rozwiązań programowych
 - zbyt duży koszt realizacji rozwiązań sprzętowych
- ☼ Problem złożoności obliczeniowej
 - pamięciowej — duża złożoność struktur danych
 - czasowej — czas potrzebny na aktualizację struktur danych oraz na zidentyfikowanie stron (ramek), które znalazły się poza zbiorem roboczym

Przybliżona realizacja koncepcji zbioru roboczego

- ☼ Wspomaganie sprzętowe — bit odniesienia.
- ☼ Okresowe (wyznaczone przez czasomierz lub przez wystąpienie błędu strony) inkrementowanie licznika reprezentującego czas wirtualny oraz sprawdzanie bitu odniesienia **dla każdej z ramek**.
- ☼ Jeśli bit odniesienia jest ustawiony, to skasowanie bitu odniesienia i wpisanie bieżącej wartości licznika do tablicy na pozycji odpowiadającej ramce.
- ☼ Jeśli bit odniesienia jest skasowany to sprawdzenie różnicy pomiędzy bieżącą wartością licznika a wartością wpisaną na odpowiedniej pozycji w tablicy i usunięcie strony, gdy różnica jest większa niż rozmiar okna.

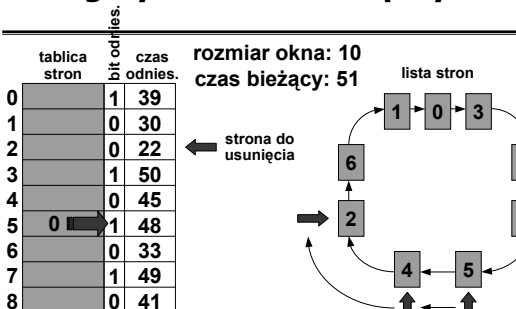
Algorytm WSClock (1)

1. Dla każdej ramki utrzymywany jest wirtualny czas (przybliżony) ostatniego odniesienia do niej.
2. Wszystkie strony w pamięci (niezależnie od przynależności do procesu) powiązane są w cykl zgodnie z kolejnością ich sprowadzenia (podobnie jak w przypadku alg. drugiej szansy)
3. W wyniku wystąpienia błędu strony sprawdzany jest bit odniesienia do strony wskazywanej jako kolejna do usunięcia.
4. Jeśli bit odniesienia jest ustawiony, zostaje on skasowany, następuje wskazanie strony następnej w cyklu i sprawdzenie bitu odniesienia.

Algorytm WSClock (2)

5. Jeśli bit odniesienia jest skasowany, sprawdzana jest różnica pomiędzy wirtualnym czasem bieżącym, a czasem ostatniego odniesienia do wskazywanej strony.
 - jeśli różnica jest większa od rozmiaru okna zbioru roboczego, strona jest usuwana, a ramka zostaje wykorzystana dla strony sprowadzanej,
 - w przeciwnym razie strona pozostaje i następuje wskazanie strony następnej w cyklu i ponawiany jest ciąg działań począwszy od punktu 4 algorytmu.
6. W przypadku braku wolnych ramek następuje wstrzymanie (zawieszenie) jakiegoś procesu (☹ planista średnioterminowy).

Algorytm WSClock — przykład



Algorytm WSClock a koncepcja zbioru roboczego

1. Na czym polega naruszenie idei zbioru roboczego przez algorytm WSClock?
2. Dlaczego w przypadku zbioru roboczego jest sens stosować algorytm usuwania inny niż na żądanie?



Implementacja przybliżająca algorytm DWS (1)

- ☼ Dla każdej ramki w systemie utrzymywany jest przybliżony czas ostatniego odniesienia.
- ☼ W przypadku wystąpienia błędu strony lub w wyniku przerwania od czasomierza sprawdzany jest bit odniesienia **dla każdej ramki**.
 - jeśli bit odniesienia jest ustawiony, jest on kasowany i następuje aktualizacja czasu odniesienia do ramki,
 - jeśli bit odniesienia jest skasowany sprawdzany jest zarejestrowany wcześniej czas ostatniego odniesienia i obliczany jest dystans w tył b_i .

Implementacja przybliżająca algorytm DWS (2)

- ☼ W zależności od wartości dystansu w tył podejmowane są następujące akcje:
 - $b_i > \tau \Rightarrow$ strona jest usuwana, a ramka dołączana do puli wolnych ramek,
 - $\alpha\tau < b_i \leq \tau \Rightarrow$ ramka jest dołączana do zbioru potencjalnych ramek-ofiar, ale strona nie jest usuwana,
 - $b_i \leq \alpha\tau \Rightarrow$ strona pozostaje w zbiorze roboczym.

Implementacja przybliżająca algorytm DWS (3)

- ☼ W przypadku wystąpienia błędu strony podejmowane są następujące akcje:
 - jeśli istnieją wolne ramki, jedna z nich zostanie wykorzystana do sprowadzenia strony,
 - jeśli nie ma wolnych ramek ale istnieją potencjalne ramki-ofiary, następuje zastąpienie strony w wybranej ramce przez stronę sprowadzaną,
 - jeśli wszystkie ramki są zajęte następuje zawieszenie jednego z procesów (☞ planista średnioterminowy) i wykorzystanie zwolnionych przez niego ramek.

Alg. wymiany ze spraw. na żąd. — PFF (ang. Page Fault Frequency)

- ☼ Jeśli częstotliwość błędów strony generowanych przez proces przekroczy ustalony poziom f_H , to dla stron tego procesu sprowadzanych do pamięci przydzielana są nowe ramki.
- ☼ Jeśli częstotliwość błędów strony spadnie poniżej ustalonego poziomu f_L , to usuwane są wszystkie strony, do których nie było odniesienia od chwili wystąpienia ostatniego błędu strony w danym procesie.
- ☼ W szczególności $f_H = f_L$

Implementacja PFF — kontrola częstości błędów strony

- ☼ Przy każdym błędzie strony generowanym przez proces zwiększany jest licznik błędów strony danego procesu oraz zerowane są bity odniesienia do jego stron.
- ☼ W określonych okresach czasu (generowanych przez czasomierz) sprawdzane są (a następnie zerowane) wartości liczników poszczególnych procesów.
- ☼ Jeśli wartość licznika jest większa od ustalonej górnej granicy, to procesowi przydzielana jest dodatkowa ramka.
- ☼ Jeśli wartość licznika jest mniejsza od ustalonej dolnej granicy, to procesowi zabierana jest jedna ramka (lub wszystkie ramki z wykasowanym bitem odniesienia).

Implementacja PFF — kontrola okresu pomiędzy błędami strony

- ☼ Przy każdym błędzie strony sprawdzany jest czas jaki upłynął od poprzedniego błędu strony.
- ☼ Jeśli czas jest mniejszy od ustalonej wielkości T_{min} , to na potrzeby sprowadzanej strony przydzielana jest dodatkowa ramka.
- ☼ Jeśli czas jest większy od ustalonej wielkości T_{max} , to wszystkie strony ze skasowanym bitem odniesienia są usuwane (ramki są zwalniane), a bit odniesienia dla stron pozostających w pamięci jest kasowany.

Alg. wymiany ze sprowadzaniem na żąd. — VSWS (ang. variable-interval sampled working set)

- Przy każdym błędzie strony zwiększany jest licznik błędów strony danego procesu — n .
- Co pewien okres czasu T_p , wynikający ze stanu wymiany, następuje kontrola odniesień do stron.
- W ramach kontroli te strony, do których nie było odniesienia są usuwane (ich ramki są zwalniane), a bity odniesienia stron pozostających na następny okres (do których było odniesienie) są kasowane. Wartość licznika n jest zerowana.

Wielkość interwału czasu dla algorytmu VSWS

- Alg. sterowany jest następującymi parametrami:
 - T_{max} — maksymalna wielkość interwału czasu
 - T_{min} — minimalna wielkość interwału czasu
 - n_{max} — maksymalna liczba błędów strony
- Kontrola następuje w chwili t , po czasie T_t od momentu poprzedniej kontroli, ustalonym następująco:
$$\begin{cases} n_t < n_{max} \Rightarrow T_t = T_{max} \\ n_t \geq n_{max} \Rightarrow T_{min} \leq T_t < T_{max} \end{cases}$$

Algorytmy wymiany z wstępnym stronicowaniem — dodatkowe oznaczenia

- $PRED(x)$ — funkcja określająca numer strony, którą należy ewentualnie wstępnie sprowadzić w przypadku wystąpienia błędu strony w momencie odniesienia do strony nr x .

Algorytmy wymiany z wstępnym stronicowaniem — DPMIN (ang. Demand Prepaging MIN)

- W chwili wystąpienia błędu strony do pamięci sprowadzane jest m pierwszych stron w przyszłym ciągu odniesień.

$$\mathcal{S}_i = \begin{cases} \mathcal{S}_{i-1} & \text{jeśli } r_i \in \mathcal{S}_{i-1} \\ \{x_1, x_2, \dots, x_l\} \ (l \leq m) & \text{jeśli } r_i \notin \mathcal{S}_{i-1} \end{cases}$$

Twierdzenie: w klasie algorytmów wymiany z wstępnym stronicowaniem na żądanie algorytm DPMIN jest optymalny ze względu na liczbę błędów strony.

Algorytmy wymiany z wstępnym stronicowaniem — OBL (ang. One Block Lookahead)

- Niech $PRED(i) = i + 1$
- W chwili t wystąpienia błędu strony następuje sprowadzenie strony $x_1 = r_t$ i wymiana zgodnie z algorytmem LRU. Jeśli strona nr $x_2 = PRED(x_1) \notin \mathcal{S}_{t-1}$, to następuje sprowadzenie strony x_2 i wymiana zgodnie z algorytmem LRU.

Algorytmy wymiany z wstępnym stronicowaniem — SL (ang. Spatial Lookahead) (1)

- Niech u oznacza nr strony ostatnio sprowadzanej do pamięci w wyniku odniesienia do niej.
- Niech początkowo $PRED(i) = i + 1$

Algorytmy wymiany z wstępnym stronicowaniem — SL (2)

- ☼ W chwili t wystąpienia błędu strony następuje sprowadzenie strony $x_1 = r_t$ i wymiana zgodnie z algorytmem LRU. Jeśli przy poprzedniej wymianie nie dokonano wstępnego sprowadzenia lub nie było odniesienia do strony wstępnie sprowadzonej do chwili obecnej $PRED(u) := r_t$.
- ☼ Jeśli strona nr $x_2 = PRED(x_1) \notin \mathcal{S}_{t-1}$, to następuje sprowadzenie strony x_2 i wymiana zgodnie z algorytmem LRU.
- ☼ $u := r_t$.

Algorytmy wymiany z wstępnym stronicowaniem — FDPA (Fixed-space Demand Prepaging Alg.) (1)

- ☼ Algorytm wykorzystuje dodatkową wiedzę wynikającą ze statycznej analizy kodu (przed wykonaniem programu) lub z informacji dostarczonych przez programistę.
- ☼ Uzyskana wiedza reprezentowana jest w ciągu zgłoszeń przez dodatkowe instrukcje dla systemu operacyjnego:
 - $FREE(x)$ — strona x nie będzie wykorzystywana w niedalekiej przyszłości,
 - $PRE(x)$ — strona x będzie wkrótce wykorzystywana.

Algorytmy wymiany z wstępnym stronicowaniem — FDPA (2)

- ☼ Z każdą stroną x skojarzone są następujące bity:
 - bit usunięcia $D(x)$
 - bit wstępnego sprowadzenia $P(x)$
- ☼ Z zbiorze stron w pamięci \mathcal{S}_t wyróżnia się następujące podzbiory:
 - $\mathcal{D}_t = \{x : x \in \mathcal{S}_t \wedge D(x) = 1\}$
 - $\mathcal{P}_t = \{x : x \in \mathcal{S}_t \wedge P(x) = 1\}$

Algorytmy wymiany z wstępnym stronicowaniem — FDPA (3)

- ☼ Zasady usuwania stron

$$\begin{cases} 0 & \text{jeśli } |\mathcal{S}_{t-1}| < m \\ y \in \mathcal{D}_{t-1} & \text{jeśli } \mathcal{D}_{t-1} \neq \emptyset \\ \text{zgodnie z LRU} & \text{w pozostałych przypadkach} \end{cases}$$

Algorytmy wymiany z wstępnym stronicowaniem — FDPA (4)

- ☼ Obsługa odniesienia do strony

$$\begin{cases} r_t = FREE(x) \wedge x \in \mathcal{S}_{t-1} \Rightarrow D(x) := 1 \\ r_t = PRE(x) \wedge x \notin \mathcal{S}_{t-1} \Rightarrow P(x) := 1 \\ r_t = x \wedge x \in \mathcal{S}_{t-1} \Rightarrow \mathcal{S}_t := \mathcal{S}_{t-1}, P(x) := 0 \\ r_t = x \wedge x \notin \mathcal{S}_{t-1} \Rightarrow \text{dokonaj wymiany} \end{cases}$$

Algorytmy wymiany z wstępnym stronicowaniem — FDPA (5)

- ☼ Realizacja wymiany: sprowadzenie żądanej strony — x , ustawienie $D(x) := 0$, i usunięcie, jeśli to konieczne, strony ze zbioru $\mathcal{S}_{t-1} \setminus \mathcal{P}_{t-1}$.
- ☼ Wstępne stronicowanie: jeśli $|\mathcal{S}_{t-1} \setminus \mathcal{D}_{t-1}| < m$ i $\exists x: P(x) = 1$, dokonaj sprowadzenia strony x z ewentualną wymianą.