Algorytm wymiany stron

Tomasz Szymko

17 grudnia 2001

Spis treści

1 Wstęp **3**

2 Wymiana stron w Linuxie **3**
2.1 Kto zaczyna wymianę stron? 3

3 Opis funkcji realizujących wymianę stron **3**
3.1 Opis funkcji swap_out 3
3.2 Opis funkcji swap_amount 4
3.3 Opis funkcji swap_out_mm 5
3.4 Opis funkcji swap_out_vma 5

4 Podsumowanie **6**

1 Wstęp

Niniejsza praca omawia algorytm wymiany stron zaimplementowany w jądrze Linuxa w wersji[1] 2.4.7. Ponieważ sam algorytm jest bardzo prosty, to skupiam się tu na opisie realizujących go funkcji.

2 Wymiana stron w Linuxie

Linux stara się zwalniać zajętą pamięć, zawsze gdy zaczyna mu jej brakować. Najpierw próbuje wykasować strony należące do buforów (bo ich zawartość i tak najpewniej jest na dysku i gdy zajdzie potrzeba to będzie można ją wczytać ponownie), a gdy to nie spowoduje zwolnienia wystarczającej ilości pamięci, wtedy rozpoczyna wymianę stron pamięci. Gdy i to nie wystarczy wtedy wybiera i morduje jeden z procesów.

Tu będzie nas interesować tylko przypadek, w którym pamięć jest wymieniana. Pozostałe aspekty związane z wymuszonym przez jądro zwalnianiem pamięci będą opisane w następnym rozdziale - w rozdziale poświęconym demonowi kswapd.

Jak każdy wie wymiana stron jest związana ze zwalnianiem pamięci RAM kosztem innych pamięci (np. dysków). Gdy systemowi zaczyna brakować pamięci i nie może już nic z niej skasować, wtedy próbuje wyrzucić pewną liczbę stron pamięci RAM do specjalnych obszarów wymiany (najczęściej plików dyskowych). Tam strona będzie przechowywana do czasu, aż będzie ponownie potrzebna, lub do czasu gdy stanie się bezużyteczna.

2.1 Kto zaczyna wymianę stron?

Wymiana stron jest realizowana w funkcji swap_out. Funkcja ta jest wołana tylko przez funkcję refill_inactive, która jest z kolei wołana tylko przez funkcję do_try_to_free_pages. Funkcja do_try_to_free_pages jest zaś wołana przez demona kswapd i w pewnych szczególnych przypadkach przez funkcję __alloc_pages przydzielającą ramki pamięci (szczegóły w pliku page_alloc.c).

Tak więc widzimy, że wymiana stron jest inicjalizowana przez funkcje związane ze zwalnianiem i przydzielaniem pamięci. Każde żądanie wymiany stron dotyczy całego systemu, i gdy tylko jest zgłaszane to system (po przez funkcję swap_out) stara się wymienić tak dużo stron jak tylko jest to możliwe.

3 Opis funkcji realizujących wymianę stron

3.1 Opis funkcji swap_out

Przejdźmy do opisu funkcji:

`static void swap_out(unsigned int priority, int gfp_mask)`

[1] W innych wersjach omawiane zagadnienia mogą być zrealizowane podobnie, albo zupełnie inaczej!

realizującej jedną sesję wymiany stron. Funkcja ta dostaje dwa parametry. Parametr priority mówi jak bardzo zależy wołającemu na zwolnieniu pamięci dzięki wymianie stron. Gdy wołającym jest kswapd, to priority=6, czyli chce on żeby zwolnić troszeczkę pamięci. Gdy jest to funkcja __alloc_pages, to priority=1, czyli trzeba zwolnić jak najwięcej się da! Drugim parametrem są flagi gfp_mask. Parametr ten nie jest używany w ciele funkcji.

Funkcja zaczyna wymianę stron od bieżącego procesu (procesu current), gdyż to on właśnie zażądał pamięci, i to przez niego dochodzi do wymiany, więc należy go ukarać! Wobec czego zostaje on potraktowany funkcją swap_out_mm (której opis jest parę punktów dalej).

Gdy swap_out zakończy wymianę stron procesowi bieżącemu, to ustala liczbę zarządców pamięci, którzy będą uczestniczyć w wymianie w następnym kroku. Liczba ta (nazwijmy ją: (counter)) ustalana jest według wzoru:

counter = (mmlist_nr « SWAP_MM_SHIFT) » priority

przy czym:

- mmlist_nr - ilość zarządców[2] pamięci w systemie;
- SWAP_MM_SHIFT - stała równa 4;
- priority - parametr funkcji swap_out.

Wymianę stron zaczynamy od zarządcy, który jest pierwszy na globalnej liście zarządców pamięci (struktur mm_struct). Dla tego zarządcy i kolejnych counter-1 zarządców z listy, robimy co następuje:
– traktujemy go funkcją swap_out_mm wymieniającą mu strony;
– przeczepiamy go na koniec wspomnianej listy zarządców.

Widać więc, że wymianę zaczynamy od zarządcy pamięci, którego ramki najdłużej nie podlegały wymianie.

3.2 Opis funkcji swap_amount

Zadaniem funkcji swap_amount jest ustalenie ile ramek należących do danego zarządcy pamięci ma zostać poddanych wymianie. Algorytm wyliczający tę liczbę jest bardzo prosty. Wystarczy się przyjrzeć ciału funkckji i wszystko będzie jasne:

```
#define SWAP_MM_SHIFT 4
#define SWAP_SHIFT 5
#define SWAP_MIN 8

static inline int swap_amount(struct mm_struct *mm) {
/*mm->rss - liczba ramek posiadanych przez zarządcę pamięci.*/
        int nr = mm->rss >> SWAP_SHIFT;
```

[2]Liczba ta nie jest równa liczbie procesów w systemie, gdyż może zaj ść sytuacja, że z jednego zarządcy pamięci korzysta więcej niż jeden proces.

```
        if (nr < SWAP_MIN) {
                nr = SWAP_MIN;
                if (nr > mm->rss)
                        nr = mm->rss;
        }
        return nr;
}
```

3.3 Opis funkcji swap_out_mm

Funkcja:

static int swap_out_mm(struct mm_struct * mm, int count)

dostaje jako parametry:

- mm - wskaźnik na zarządcę pamięci;
- count - liczbę ramek (ustaloną funkcją swap_amount), którą należy wymienić temu zarządcy.

Funkcja ta dla podanego menedżera pamięci:

— przechodzi listę jego obszarów VMA, zaczynając od obszaru VMA zawierającego adres mm->swap_address, dotąd aż dojdzie do jej końca (wtedy zeruje zmienną mm->swap_address) lub wymieni żądaną (count) liczbę ramek.

— dla każdego oglądanego obszaru VMA zapuszcza funkcję swap_out_vma.

3.4 Opis funkcji swap_out_vma

Funkcja:

static int swap_out_vma(struct mm_struct * mm, struct vm_area_struct * vma, unsigned long address, int count)

dostaje jako parametry:

- mm - wskaźnik na zarządcę pamięci, którego będzie obsługiwać
- vma - wskaźnik na obszar VMA, który ma poddać wymianie
- address - adres początku tego VMA
- count - liczbę ramek które mają zostać poddane wymianie

Funkcja swap_out_vma sprawdza czy dany obszar VMA ma ustawione flagi:
VM_LOCKED lub VM_RESERVED.
Jeśli tak to nic nie robi z tym obszarem, bo flagi te oznaczają, że dany obszar nie może być zabrany z pamięci. W przeciwnym przypadku zaczyna wymianę ramek wchodzących w skład tego VMA i w tym celu wywołuje funkcję:

swap_out_pgd, która z kolei woła funkcję: swap_out_pmd

Zadaniem tych dwóch funkcji jest wymiana wszystkich lub tylko count ramek należących do danego VMA. Toteż dla każdej ramki, o ile jest to poprawna ramka i nie ma ustawionej flagi zabraniającej wymiany, wołana jest funkcja:

try_to_swap_out

Funkcja try_to_swap_out dla zadanej ramki robi co następuje:
jeśli ramka ta przed chwilą była używana, procesor ustawił jej flagę dostępu, to zostawia ją w pamięci, ale kasuje tę flagę i zwiększa wiek o 3. W przeciwnym przypadku podejmuje się wymiany. Strona jest wymieniana jeśli jest na nią miejsce w obszarze wymiany. (to jest niewielkie uproszczenie, ale dobrze odzwierciedla całość).

4 Podsumowanie

Wymianie stron podlegają wszystkie procesy, których zarządcy pamięci są na globalnej liście struktur mm_struct. Wymianę zawsze zaczynamy od stron należących do bieżącego procesu. Następnie wymieniamy strony pewnej liczbie (zależnej od sytuacji) zarządców będących na początku wspomnianej listy. Każdy taki zarządca gdy wymienimy mu pewną część stron ląduje zawsze na końcu listy zarządców. Tak więc ogólna strategia sprowadza się do wymiany stron procesom najdłużej przebywającym w pamięci.

Pamiętać należy, że wymianie nie podlegają strony z ustawionymi flagami zabraniającymi ruszać je z pamięci. Nie wymienia się również stron ostatnio używanych przez procesor, gdyż jest duże prawdopodobieństwo, że wkrótce znowu zostaną użyte.

Bibliografia

[1] A. Silberschatz, P. B. Galvin: *Podstawy systemów operacyjnych*, WNT 2000

[2] Kody źródłowe jądra Linuxa w wersji 2.4.7.

[3] Projekty studentów MIMUW z lat poprzednich