Kradení stránek procesům - zloděj stránek

Úvod

Kradení stránek procesům provádí speciální proces jádra, který se nazývá zloděj stránek. Tento proces se zavádí při startu systému a jeho náplní je dohlížet a udržovat dostatečné množství volných paměťových stránek. Kontrola počtu volných stránek je prováděna periodicky po určitém časovém okamžiku (který je vždy spočten) nebo při akutním nedostatku volných stránek - přímým voláním jádra. I v případě, kdy je dostatečný počet volných stránek, se zloděj pokusí nejaké stránky ukrást. Algoritmus kradení stránek v operačním systému LINUX je v porovnání s jinými OS více agresivní.

Obecné seznámení

- Každá stránka má svůj věk.
- Čím více je přístupů na stránku (čtení, modifikace), tím je stránka mladší.
- Staré stránky jsou kandidáty na odložení (swapping).
- Stránky se stávají staršími při každém spuštění zloděje stránek.
- Kernel Swap Daemon používá tři způsoby redukce stránek:
    1. redukce velikosti cache bufferů (jádra, ovladačů zařízení,...) a stránkových cache (datové soubory, programy,...). Jelikož nejsou měněny, není třeba je ukládat. V případě potřeby je můžeme vždy natáhnout zpět do paměti.
    2. swapování sdílených stránek v daemonu je provedeno pouze symbolicky - z logické souvyslosti. Vlastní mechanismus je implementován v jiné části jádra.
    3. redukce ostatních stránek (stránky procesů) - modifikované swapujeme, nemodifikované "zapomínáme" (pokud byly již dříve swapovány a jsou tedy uloženy na disku). Odkládání celých procesů se nepoužívá, neboť mechanismus "uždibování" stránek procesům je výhodnější.

Popis funkcí

- `void kswapd_setup(void)` *"mm/vmscan.c"*
  Vypíše inicializační zprávu. Výpis není včleněn do funkce `kswapd()`, protože ta je spouštěna jako vlákno a vypsaný text by mohl být zobrazen do jiných výpisů (což by bylo matoucí).
- `int kswapd(void * unused)` *"mm/vmscan.c"*
  Vlastní zloděj stránek. Je spuštěn při inicializaci systému jako vlákno a je aktivní až do ukončení běhu systému. Z tohoto důvodu nevrací žádnou hodnotu.
- `int try_to_free_page(int priority, int dma, int wait)` *"mm/vmscan.c"*
  Volá další funkce, které provádějí vlastní uvolňování stránek. Při jednom vykonání `try_to_free_page()` jsou tyto funkce volány vícekrát. Každé jejich následující volání je prováděno s vyšším nátlakem na uvolnění stránky. Pokud se podaří uvolnit stránku, vrací hodnotu 1, jinak 0.
- `int shrink_mmap(int priority, int dma, int free_buf)` *"mm/filemap.c"*
  Prochází existující stránky. Vhodné stránky uvolní a u ostatních stránek sníží věk (stránky stárnou). Uvolňované stránky jsou: cache stránky (stránky urychlující přístup k souborům na disku) a stránky cache bufferů (stránky používané pro efektivnější alokaci a dealokaci bufferů). Druhé zmiňované mohou být uvolňovány pouze pokud je nastaven příznak `free_buf`, který je do funkce předáván jako parametr `can_do_io`. V případě uvolnění stránky vrací funkce hodnotu 1, jinak 0.
- `static inline void age_page(struct page *page)` *"linux/swapctl.h"*
  Výpočet nového věku stránky. Pokud je věk stránky větší než hodnota konstanty `DEC_LINE` (v

inicializaci nastaveno na 1) je věk stránky o tuto hodnotu snížen - stránka stárne. V opačném případě je věk stránky nastaven na hodnotu 0 - stránka je nejstarší.

- `int shm_swap(int prio, unsigned long limit)` *"ipc/util.c"*
  Funkce nevykonává žádnou činnost. Uvedena je pouze pro logickou souvislost s ostatními funkcemi, které se týkají uvolňování stránek. Vždy vrací hodnotu 0.
- `int swap_out(unsigned int priority, int dma, int wait, int can_do_io)` *"mm/vmscan.c"*
  Prochází jednotlivými procesy (tasky) a při nalezení swapovatelného se pokusí o uvolnění některých jeho stránek. Vlastní swapování provádějí další funkce jádra. Tato činnost se již přímo nevztahuje k problematice algoritmu kradení stránek, a proto zde nebude více rozebírána.

Implementace

Daemon je realizován funkcí `kswapd()`, která je spuštěna jako vlákno. Na počátku je změněna její priorita na prioritu procesu reálného času. Výkonná část funkce je opakována v nekonečném cyklu, který je popsán až do konce tohoto odstavce. Je spočten příští okamžik probuzení a daemon je uspán s možností přerušení, tzn. že je probuzen po vypršení spočteného času nebo voláním jádra (při akutním nedostatku volných paměťových stránek). Po probuzení daemon zjišťuje, zda je v systému k dispozici dostatečný počet volných stránek a na základě tohoto údaje určí minimální počet stránek, které by měly být ukradeny. Tuto hodnotu zvýší o předem určenou rezervu. Navýšení o tuto rezervu zabraňuje, aby se počet volných stránek pohyboval trvale u svého minima. Tím se snižuje počet probouzení daemona. Dále je volána funkce`try_to_free_page(GFP_KERNEL, 0, nr_free_pages <= min_free_pages)` tolikrát, kolik stránek potřebujeme ukrást. Poslední parametr (podmínka) určuje naléhavost provedení - viz dále.

Pokud je podmínka pravdivá (počet volných stránek <= minimální potřebný počet volných stránek), je naléhavost získání volné stránky větší a následující volání funkcí ve smyčce `do...while` je provedeno ve dvojnásobném množství (zde konkrétně 6krát), než při nesplnění podmínky (pouze 3krát). Celá smyčka je uzavřena do příkazu `switch` a s použitím lokální statické proměnné `state` je dosaženo pokračování programu ve větvi `case` naposledy prováděné. V jednotlivých větvích jsou volány funkce `shrink_mmap(i, dma, can_do_io)`, `shm_swap(i, dma)`, `swap_out(i, dma, wait, can_do_io)`.

Funkce `shrink_mmap()` probíhá v cyklu, jehož meze nejprve spočteme (z počtu stránek paměti a priority kradení, která je předána jako vstupní parametr `i`). Meze jsou v cyklu dekrementovány a v případě alespoň jedné hodnoty menší než 0 je cyklus ukončen. Funkce je též ukončena po uvolnění stránky. Postupně je procházeno pole odkazů na struktury stránek, a protože indexem tohoto pole je statická proměnná `clock`, pokračuje prohledávání vždy v místě posledního použitého odkazu. Stránky zamčené a právě odkládané jsou z dalšího zpracování vyloučeny (přeskočeny). U ostatních stránek prohlédneme všechny její buffery (cyklický zřetězený seznam začínající v `page->buffers`) a testujeme, zda byly dotknuty (touched), tzn. byly čteny, modifikovány apod. Pokud ano, je příznak dotknutí zrušen (připraven pro novou detekci) a nastaví se příznak stránky - "byla odkazována" (`PG_referenced`). Další činnost funkce závisí na počtu uživatelů stránky. Pokud není žádný, neprovádí se nic. V případě více než jednoho uživatele nelze stránku uvolnit a nastaví se příznak odkazování`PG_referenced`. Jestliže existuje pouze jeden uživatel stránky a je nastaven příznak odkazování, příznak zrušíme. V případě vyšší naléhavosti (`priority < 4`) vypočteme nový věk stránky pomocí funkce `age_page()` (stránka stárne). Pokud není nastaven příznak odkazování, zjišťujeme, zda se jedná o cache stránku (platná položka `page->inode`) - ta je uvolněna nebo zda se jedná o stránku cache bufferů - v tomto případě jsou otestovány všechny buffery na této stránce a nejsou-li používané, uvolní se.

Funkce `shm_swap()` je použita pro odkládání sdílených paměťových stránek. To je však obsluhováno

vlastním mechanismem, a proto tato funkce nevykonává žádnou činnost. Uvedena je pouze pro logickou souvislost s okolními funkcemi.

Funkce `swap_out()` nejprve naplní proměnnou `shfrv` v závislosti na hodnotě priority - čím nižší číslo priority, tím vyšší priorita a tím nižší hodnota `shfrv`. Podle hodnot priority je intenzita hledání vhodné stránky rozdělena do tří základních skupin:

- přátelská - priorita 6, 5, 4 (`shfrv=10`)
- více intenzivní - priorita 3, 2, 1 (`shfrv=9`)
- nekompromisní - priorita 0 (`shfrv=8`)

Nyní, když je známo s jakou důležitostí je třeba stránku uvolnit, stanoví se hodnota proměnné `counter`, která určuje počet průchodů cyklem `for` tvořícího hlavní část těla funkce. Cyklus`for` lze dle činnosti rozdělit do dvou částí. První část je prováděna v nekonečné smyčce a cyklicky prohledává pole odkazů na tasky. Smyčka je ukončena po nalezení swapovatelného tasku nebo po dvojnásobném překročení horní hranice počtu tasků, tzn. swapovatelný task nebyl nalezen - v tomto případě je funkce ukončena a vrací hodnotu 0. Ve druhé části (v případě nalezeného vhodného procesu v první části) nejprve zjistíme, zda má proces nějaké stránky ke swapování. Jestliže ne, je mu tento počet stanoven funkcí`AGE_CLUSTER_SIZE()` (nová hodnota není nulová). Teď již můžeme zavolat funci `swap_out_process(p, dma, wait, can_do_io)`, která za pomoci celé řady dalších funkcí provede vlastní pokus o vyswapování některých stránek procesu. Po úspěšném provedení vrací hodnotu 1, při neúspěchu vrací hodnotu 0.