Algorytm przydziału nowego i-węzła na dysku

Grzegorz Czaja, Artur Lasek
Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego

20 grudnia 2000

Spis treści

1 Wprowadzenie

Kilka słów o organizacji danych w systemie plików:

” każdy zamontowany system plików zawiera blok identyfikacyjny czyli super blok,

” każdy super blok ma pewną ilość grup,

” grupa składa się z tablicy i-węzłów i bloków dyskowych,

” każda grupa zna numer bitmapy zajętości i-węzłow oraz bitmapy zajętości bloków dyskowych,

” w pamięci znajduje się lista wolnych i-węzłów (w pamięci istnieje osobna lista dla i-węzłów używanych).

2 Wiadomości wstępne

Algorytm ialloc służy do przydzialu i-węzła dyskowego nowo tworzonemu plikowi. Algorytm opiszemy na przykładzie implementacji dla systemu ext2. W Linuxie 2.2.17 jest on zaimplementowany w funkcji ext2_new_inode i znajduje sie w pliku fs/ext2/ialloc.c. Algorytm różni się szczegółami w zależności od rodzaju obsługiwanego systemu plików.

Algorytm przydziału i-węzla zależy od tego czy przydzielamy i-węzeł katalogowi czy plikowi innego typu. Jeśli plik jest katalogiem to przydzielany jest mu i-węzeł z grupy bloków, w której znajduje się optymalna (dla algorytmu) ilość wolnych i-węzłów i bloków dyskowych. Wymaga to przeszukania wszystkich deskryptorów grup, w których znajdują się liczniki wolnych i-węzłów. Nie jest to kosztowne, bowiem tworzenie nowego pliku nie jest czynnością powtarzaną z dużą częstotliwością w systemie, a ponadto wszystkie deskryptory grup wczytywane są do pamięci w momencie ładowania systemu i pozostają tam stale do końca działania systemu (jest ich od kilkudziesięciu do kilkuset).

Jeżeli plik nie jest katalogiem, to szukanie wolnego i-węzła zaczynamy od grupy, w której znajduje się katalog tworzonego pliku. Powoduje to, że z większym prawdopodobieństwem pliki leżące w tym samym katalogu będą umieszczone w tej samej grupie bloków. Jeżeli w grupie katalogu nie istnieją wolne i-węzły, to przeszukiwanie grup odbywa się przy pomocy algorytmu quadratic hash. Gdy i to nie da rezultatu następuje liniowe przeszukiwanie wszystkich grup w celu znalezienia grupy z wolnymi i-węzłami.

Po wybraniu grupy bloków, pozostaje wyszukanie wolnego i-węzeła w grupie i przydzielenie go utworzonemu plikowi. Informacja, który i-węzeł w grupie bloków jest wolny, a który zajęty przechowywana jest w mapach bitowych. W odróżnieniu od deskryptorów grup NIE wszystkie mapy bitowe zajętości i-węzłów muszą być przechowywane w pamięci. Zatem może się zdarzyć, że tylko niewielka część bitmap jest w pamięci. Do wymiany bitmap w pamięci stosowany jest algorytm LRU (realizowany w funkcji load_inode_bitmap()). Po znalezieniu i-węzła bufory zawierające deskryptor grupy i mapę bitową są zaznaczane jako ”dirty” w celu ich późniejszego zapisu na dysk, a wskaźnik do wybranego i-węzła jest zwracany jako wynik funkcji.

3 Schematy algorytmów

3fi1 Schemat algorytmu efit2finewfiinode

UWAGA: Należy odróżniać i-węzły w pamięci od i-węzłów na dysku.

Algorytm musi znaleźć zarówno wolny i-węzeł w pamieci jak i na dysku.

ffffffff ffffffffffff 1

ffffffff ffff ffffffffffffffff ffffffffffffffff

1. Algorytm znajdowania optymalnej grupy dla pliku będącego katalogiem składa się z dwóch cześci:
    (a) Przeszukujemy grupy (zaczynając od grupy w której znajduje się katalog dir) w poszukiwaniu grupy spełniającej warunek ((liczba katalogów w tej grupie) * 256) < (liczby wolnych i-węzłów w tej grupie) (256 wzieło się z « 8)
    (b) Jeżeli nie znaleźliśmy takiej grupy to:
        ” wyliczamy średnią ilość wolnych i-węzłów w grupie /* ilość wolnych i-węzłów / ilość grup */
        ” przeszukujemy po kolei (zaczynając od grupy w której znajduje się katalog dir) wszystkie grupy w poszukiwaniu takiej która ma najwięcej wolnych bloków dyskowych z tych, które maja więcej niż przeciętna ilość wolnych i-węzłów.

```
Dokładniej: w pętli sprawdamy następujacy warunek:
if (liczba wolnych i-węzłów w grupie > 0 && liczba wolnych i-węzłów w grupie >= wyliczona średnia)
    if (jeszcze nie zapamietano żadnej grupy || zapamiętana wcześniej grupa ma mniej wolnych bloków dyskowych niż aktualnie badana)
        zapamiętaj aktualnie badaną grupę.
```

ffffffffffff 1 ext2_new_inode

```
ffffffff ffffffśffffffffff ffffff - wskaźnik do i-węzła katalogu w którym ma być umieszczony plik, dla którego ma być zaalokowany i-węzeł.
    mode - flaga opisująca typ pliku np. czy katalog lub plik specjalny
    err - wskaźnik do zmiennej przekazującej rodzaj błędu funkcji
ffffffffff i-węzeł który ma być przypisany nowemu plikowi
pobierz nowy i-węzeł z pamięci; /* funkcja get_empty_inode */
ffffśffff nie udało się pobrać i-węzła ffff
    return NULL; /* lista i-węzłów w pamięci jest pusta */
ffffffffff ffffśffff
zaznacz, że pobrany i-węzeł jest używany;
ffffśffff jest blokada na super bloku ffff
    czekaj na odblokowanie super bloku;
ffffffffff ffffśffff
zablokuj dla siebie super blok;
/* ffęffffffffffff ffffffffffffffffć fffffffff ffff fffffffff ff ffffóffffff ffęfffffff fffffffff ff-ffęffffł */
etykieta_powtarzaj:
ffffśffff mode="katalog" ffff
    znajdujemy najlepszą grupę;
    /* UWAGA: Sposób w jaki to robi jest cokolwiek dziwaczny więc opiszemy go na końcu tego rozdziału */
ff ffffffffffffffff ffffffffffffff
    ffffśffff jest wolny i-węzeł w grupie do której należy i-węzeł dir ffff
        to właśnie ta grupa jest przez nas szukana;
    ff ffffffffffffffff ffffffffffffff
        używamy quadratic hash by znaleźć grupę z wolnym i-węzłem
        /* UWAGA: algorytm quadratic hash opiszemy na końcu rozdziału */
        ffffśffff wciąż takiej nie znaleźliśmy ffff
            używamy przeszukiwania liniowego w celu znalezienia wolnego i-węzła
        ffffffffff ffffśffff
    ffffffffff ffffśffff
ffffffffff ffffśffff
ffffśffff nie znalaziono grupy z wolnymi i-węzłami ffff
    odblokuj super blok;
    zwolnij i-węzeł (algorytm iput);
    return NULL;
ffffffffff ffffśffff
/* Szukamy pierwszego wolnego i-węzła w znalezionej grupie */
wczytaj bitmapę zajętości znalezionej grupy /* load_inode_bitmap */
znajdujemy pierwszy wolny i-węzeł w znalezionej grupie
ffffśffff not wszystko OK ffff
    goto etykieta_powtarzaj; /* UWAGA: jest tu kilka dziwnych testów które opiszemy na końcu rozdziału */
ff ffffffffffffffff ffffffffffffff
    zaznaczamy bit dirty, w celu odnotowania, że dokonano zmian w bloku danych zawierajacym bitmapę;
    ffffśffff w super bloku jest ustawiona flaga MS_SYNCHRONOUS ffff
        zapisujemy ten blok na dysku;  /* żądanie natychmiastowego zapisu */
    ffffffffff ffffśffff
ffffffffff ffffśffff
ffffśffff indeks znalezionego i-węzła jest poza ustalonymi granicami ffff
    return NULL;
ffffffffff ffffśffff
/* ffffff ffffffffffffffff ff-ffęffffł ffffęff ffffffffffff ffffffffffffff fffffff ff ffff fffffóffffff */
zmniejsz w danej grupie liczbę wolnych i-węzłów;
ffffśffff mode = "katalog" ffff
    zwiększ liczbę katalogów w grupie;
ffffffffff ffffśffff
zaznacz, że zmieniła się zawartość grupy;
wypełnij inicjalnie i-węzeł;
wstaw i-węzeł na początek kolejki mieszającej; /* insert_inode_hash */
zaznacz, że zmieniły się dane w
```

5. **ffffffffff fffff2fffffffffffffffffff**

   ffffffffffffffffffffffffffffffff - liczba wolnych i-węzłów

   s_inodes_count - liczba i-węzłów na dysku

6. **ffffffffff fffff2ffffffffffffff**

   fffffff - dowiązanie do (dyskowego) super block-u w buforze s_sbh

   s_group_desc - dowiązanie do tablicy deskryptorów grup

   s_groups_count - liczba grup w systemie

   s_inode_bitmap - dowiązanie do bufora zawierającego tablicę wczytanych z dysku bitmap zajętości i-węzłów

   s_inode_bitmap_number - tablica z numerami wczytanych bitmap zajętości i-węzłów

   s_loaded_inode_bitmaps - liczba wczytanych bitmap zajętości i-węzłów do pamięci

   s_sbh - dowiązanie do bufora zawierającego super block

7. **ffffffffff fffff2ffffffffffffffffff**

   ffffffffffffffffffffffffffffffffffff - liczba wolnych bloków dyskowych

   bg_free_inodes_count - liczba wolnych i-węzłów

   bg_used_dirs_count - liczba katalogów

6 Skomentowane źródła

UWAGA: Tę część, drogi czytelniku, możesz spokojnie pominąć. Ale, z drugiej strony, może chcesz się dowiedzieć jak to zaimplementowano? Staraliśmy się opatrzyć kod dokładnymi komentarzami. Funkcje zawarte są w pliku fs/ext2/ialloc.c.

6.1 ffffffffffff fffff2ffffffffffffffff

```
ffffffffff ffffffff * fffff2ffffffffffffffff (ffffffff ffffffffff ffffffff * fffff, fffff ffffff, fffff * fffff) ff
    /* ffffffff ffffffffff ffffffff * fffff - wskaźnik do i-węzła katalogu tworzonego pliku
     * int mode - rodzaj pliku: katalog lub nie katalog
     * int * err - wskaźnik do zmiennej, gdzie jest zwracany błąd
     * Funkcja zwraca wskaźnik do i-węzła lub NULL w przypadku błędu.
     */
    struct super_block * sb;
    struct buffer_head * bh;
    struct buffer_head * bh2;
    int i, j, avefreei;
    struct inode * inode;
    int bitmap_nr;
    struct ext2_group_desc * gdp;
    struct ext2_group_desc * tmp;
    struct ”’’[<”Ofl]”M_n(”,””l”’)o(od)o((”’’i,AZA”3”b(1;dXACuRUSYy(inod&_[”Z”’’’’«”]fl)o(”6FfiA”’’’Go(s)37”;xRZ]GI”Fdir->u.?(”A
```

```
  inode--i--ount - --
  inode--i-nlink - --
  inode--i-de- - s---s-de--
  inode--i-uid - -urrent---suid-
  i- -test-opt -s-- -------
    inode--i-gid - dir--i-gid-
  else
    i- -dir--i-mode - -------- -
      inode--i-gid - dir--i-gid-
      i- ---------mode--
        mode -- --------
      - else
        inode--i-gid - -urrent---sgid-
  inode--i-dirt - --
  inode--i-ino - --
  inode--i--lksize - -A--------
-- --is is t-e optimal -- size --or stat-- not t-e -s -lo-k size --
  inode--i--lo-ks - --
  inode--i-mtime - inode--i-atime - inode--i--time - ------------
  inode--u-e-t--i-i-new-inode - --
  inode--u-e-t--i-i--lags - dir--u-e-t--i-i--lags-
  i- ---------mode--
    inode--u-e-t--i-i--lags -- ------------A------ - -----A----------
  inode--u-e-t--i-i--addr - --
  inode--u-e-t--i-i--rag-no - --
  inode--u-e-t--i-i--rag-size - --
  inode--u-e-t--i-i--ile-a-l - --
  inode--u-e-t--i-i-dir-a-l - --
  inode--u-e-t--i-i-dtime - --
  inode--u-e-t--i-i--lo-k-group - i-
  inode--i-op - -----
  i- -inode--u-e-t--i-i--lags - ------------
    inode--i--lags -- --------------
  insert-inode--as--inode--
  in--inode--ersion -inode- gdp- mode--
  unlo-k-super -s---
  i- -s---d--op- -
    s---d--op--initialize -inode- ----
    i- -s---d--op--allo--inode -inode- --- -
      s---d--op--drop -inode--
      inode--i-nlink - --
      iput -inode--
      -err - --------
      return -----
    -
    inode--i--lags -- --------
  -
  e-t--de-ug --allo-ating inode -lu-- inode--i-ino--
  -err - --
  return inode-
-
```

---   -unk--a load-inode--itmap

W pamięci może być jednocześnie załadowanych EXT2_MAX_GROUP_LOADED bitmap. W super blocku znajduje się tablica inode_bitmap[] o rozmiarze EXT2_MAX_GROUP_LOADED, w której znajdują się wskaźniki do bitmap w pamięci. Jeżeli liczba grup jest mniejsza od tej stałej to nie ma problemu (wszystkie bitmapy mogą znajdywać się w pamięci), w przeciwnym wypadku jeżeli nastąpi żądanie odczytania bitmapy, która nie znajduje się w pamięci, to zgodnie z algorytmen LRU wyrzuca się bitmapę najdłużej nieużywaną a na jej miejsce wczytuje się z dysku żądaną bitmapę.

stati- int load-inode--itmap -stru-t super--lo-k - s-- unsigned int -lo-k-group- - --

```
    - S- - wskaźnik do super blocku w pamięci
    * block_group - numer grupy, której odczytamy bitmapę zajętości i-węzłów.
    * Funkcja zwraca numer indeksu tablicy inode_bitmap[], pod którym znajduje się wskaźnik do bitmapy zajętości
i-węzłów grupy bloków o numerze block_group.
    */
  int i, j, retval = 0;
  unsigned long inode_bitmap_number;
  struct buffer_head * inode_bitmap;

  Sprawdzenie poprawności wartości argumentów.
    if (block_group >= sb->u.ext2_sb.s_groups_count)
      ext2_panic (sb, "load_inode_bitmap", "block_group <= groups_count
                - " "block_group = "d, groups_count = "lu",
                block_group, sb->u.ext2_sb.s_groups_count);
    if (sb->u.ext2_sb.s_loaded_inode_bitmaps > 0 &&
          sb->u.ext2_sb.s_set Eor inode"e("d"U)h(j);)150Z397"y(goto)fi(rebeat9)19fl497"x(")27"J96"x"Fk(Buf"r,)27"b(ktM63rx
```