Obsługa map bitowych zajętości bloków

20 grudnia 2000

Piotr Gaweł

1 Krótkie wprowadzenie do systemu plików

Poważnym problemem występującym przy egzystencji wielu systemów operacyjnych na jednym komputerze jest zgodnosć standardów zapisu plików. Praktycznie każdy system posiada swój własny system zapisu, co poważnie utrudnia przenoszenie danych między systemami. W tym włanie celu opracowany został Wirtualny System Plików (VFS) - obecny w systemie Linux pośrednik między funkcją systemową operującą na dysku, a sterownikiem i kontrolerem urządzenia dyskowego. Funkcja systemowa nie operuje więc na danych związanych z typem danego urządzenia i obecnym na nim systemie plików - tym zajmuje się włanie VFS. Najbardziej interesujący nas system ext2, najpowszechniejszy system stosowany w Linuxach, jest po prostu jednym z systemów obsługiwanych przez VFS.

Ogólna struktura dysku sformatowanego w systemie ext2 jest następująca: na początku występuje Boot Sector, a po nim następują kolejne tzw. grupy. Grupa jest po prostu serią następujących po sobie w ilości stałej dla danej partycji bloków. Podział na grupy ma miejsce podczas formatowania dysku i nie jest możliwy do zmiany bez ponownego formatowania.

Na początku każdej grupy znajduje się ze względów bezpieczeństwa kopia tzw. superbloku, zawierającego istotne z punktu widzenia systemu plików dane, m.in. ilość bloków na dysku, w tym ile jest wolnych, ilość bloków w grupie, pozycje pierwszego bloku zawierającego dane oraz wreszcie ilość grup bloków. Utrzymywanie kopii superbloku w każdej grupie jest używane m.in. przez program e2fsck. Za kopią superbloku znajduje się tzw. deskryptor grup, w którym znajdziemy lokalne informacje na temat każdej z grup. Jest to po prostu tablica której elementami są dane typu struktury opisującej pojedynczą grupę i zawierającej dane takie jak ilość wolnych bloków w grupie, ilość wolnych i-węzłów, adresy tablicy i-węzłów, mapy bitowej i-węzłów oraz tego, co z punktu widzenia tematu najbardziej istotne, czyli mapy bitowej zajętości bloków danych. Oczywiście, deskryptor grup, ponieważ dotyczy wszystkich grup bloków, jest przechowywany w każdej z grup w postaci identycznej wszędzie kopii.

Mapa bitowa jest, jak można się domyślić, zmienną traktowaną jako tablica bitów oznaczającą zajętość lub dostępność odpowiedniego bloku na dysku. Cel istnienia mapy jest dosyć oczywisty - ograniczyć ilość operacji dyskowych przy wyszukiwaniu wolnych bloków. Dzieki istnieniu mapy i wyrafinowanym algorytmom przydziału bloku ilość operacji dyskowych, zwłaszcza podczas odczytu, jest znacznie zmniejszona. Rozwiązanie to używane jest w wielu systemach operacyjnych. W Linuxie wolny blok oznaczany jest zerem, zajęty jedynką.

2 Jak to dziala ?

Ostatnio czytane bitmapy sa przechowywane w buforze znajdującym się w superbloku. Bufor zawiera określoną maksymalną ilość bitmap, w obecnej wersji jądra wynosi ona 8. W zwiazku z tym ogólna idea obsługi map jest następująca:

1. sprawdź czy odpowiednich danych nie ma już w buforze;
2. jeśli nie, to załaduj je do bufora, ew. wyrzuć to co w buforze najmniej potrzebne;
3. odpowiednie operacje wykonaj na buforze;
4. zaznacz bufor do zapisu;

Funkcje, które są używane do zajmowania i zwalniania bloków dyskowych to odpowiednio ext2_new_blocks oraz ext2_free_blocks. W algorytmie zwalniania bloków trudno znaleźć cokolwiek ciekawego w przeciwieństwie do algorytmu zajmowania bloków, który używa dwóch sposobów optymalizacji w celu ograniczenia ilości przesunięć głowicy dysku. Ponieważ jednak jest to tematem osobnego opracowania, nie będziemy się tutaj zajmować tymi algorytmami. Istotne są funkcje bezpośrednio obsługujące mapy, a z których te algorytmy pośrednio lub bezpośrednio korzystają. A są to :

Algorytm 1 load_block_bitmap

Dane wejściowe: adres_superbloku numer_grupy
Wynik: slot kod_bledu

```
jeśli ladowane jest to samo co ostatnio i ostatnie ladowanie zakonczylo sie sukcesem to
   zwroc 0
koniec jeśli
jeśli jest zaladowana grupa o numerze block_group to
   zwroc jej slot
w przeciwnym przypadku
   wykonaj load__block_bitmap(adres_superbloku, numer_grupy)
koniec jeśli
```

Algorytm 2 load__block_bitmap

Dane wejściowe: adres_superbloku numer_grupy
Wynik: slot lub kod_bledu

jeśli jest wolne miejsce w buforze **to**
  wykonaj read_block_bitmap (adres_superbloku, numer_grupy, numer_grupy)
  zwroc slot lub ew. kod bledu
koniec jeśli
za pomoca algorytmu LRU znajdz wolny slot
wykonaj read_block_bitmap (adres_superbloku, numer_grupy, znaleziony_slot)

Algorytm 3 read_block_bitmap

Dane wejściowe: adres_superbloku numer_grupy numer_bitmapy
Wynik: kod_sukcesu kod_bledu

jeśli jest wolne miejsce w buforze (sprawdzane za pomoca funkcji ext2_get_group_desc) **to**
  zwroc 0
w przeciwnym przypadku
  za pomoca funkcji bread zaladuj bitmape do bufora
  zwroc kod sukcesu lub ew. kod bledu
koniec jeśli

Algorytm 4 ext2_get_group_desc

Dane wejściowe: adres_superbloku numer_grupy naglowek_bufora
Wynik: adres_deskryptora_grupy

jeśli grupa o podanym numerze jest w buforze **to**
  zwroc adres
w przeciwnym przypadku
  zwroc 0
koniec jeśli

W ten sposób wywołując funkcje load_block_bitmap zawsze będziemy w stanie możliwie szybko znaleźć odpowiedni slot w buforze, do którego zostanie załadowana mapa w celu zwolnienia lub przydzielenia. A zajmuja sie tym wspomniane wczesniej funkcje ext2_new_blocks oraz ext2_free_blocks zdefiniowane, podobnie jak funkcje opisane wyzej, w pliku fs/ext2/balloc.c.

3 Podsumowanie

Warto zaznaczyć, iż pomimo oczywistych zalet ze stosowania map bitowych, objawiających się szybszym zapisem/odczytem, istnieją dwie podstawowe wady. Mapy bowiem, jak każde inne dane, zajmują dodatkowe miejsce na dysku. Trudno jednak uznać to za wadę poważną, gdyż w porownaniu do standardowej wielkości bloku dyskowego (1-4 KB) pojedynczy bit to istotnie niewiele. Ważniejszym wydaje się być inny mankament, mianowicie wzrost wraz z rozmiarem partycji rozmiaru mapy, co wiąże się z wydłużeniem średniego czasu ładowania mapy. Właśnie dlatego w niektórych systemach, jak XFS lub JFS zamiast tablic bitowych stosuje się B-drzewa.

Bibliografia

[1] Pliki zrodlowe:
include/linux/fs.h
include/linux/ext2_fs_sb.h
include/linux/ext2_fs.h
fs/ext2/balloc.c

[2] Remy Card: *Design and Implementation of the Second Extended Filesystem* http://e2fsprogs.sourceforge.net/ext2intro.html

[3] *Ext2 HomePage* http://web.mit.edu/tytso/www/linux/ext2.html

[4] *Opis systemu ext2* http://uranus.it.swin.edu.au/ jn/explore2fs/es2fs.htm