System plików ReiserFs

Mateusz Zakrzewski

18 stycznia 2004

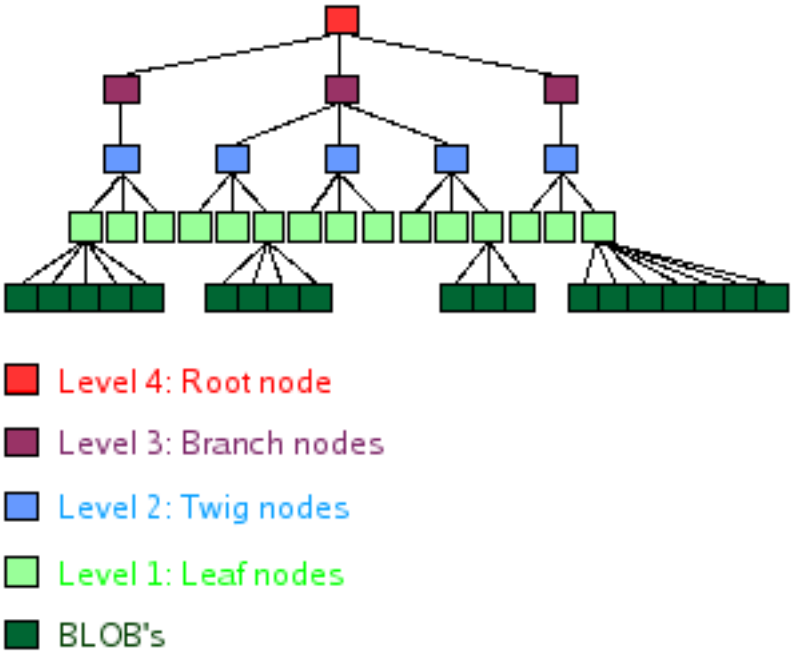
1 Najważniejsze informacje.

- Autor tego systemu plików: Hans Reiser oraz firma Namesys.
- ReiserFs to skrót od Reiser File System.
- Wersja 3 to najnowsza stabilna wersja ReiserFs'a. Jej zmiany są związane wyłącznie z usuwaniem błedów (ostatnio to nie jest konieczne).
- Wersja 4 to wersja najintensywniej rozwijana.

2 Zalety ReiserFs'a.

- Umożliwia szybki restart systemu po awarii dzięki księgowaniu.
- Jest efektywny, zwłaszcza, gdy w systemie jest dużo małych plików.
- Wykorzystuje drzewa zrównoważone, co umożliwia szybkie wykonywanie operacji wstawiania, wyszukiwania i usuwania informacji.

Struktura drzewa w systemie plików ReiserFs w wersji 3.



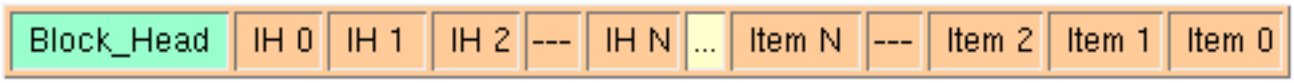
- To jest B+ drzewo
- Kazdy wezel jest tej samej wielkosci.
- Węzły, które nie zawierają metadanych to węzły *niesformatowane*. Węzły, które składają się z listy wpisów, to węzły *sformatowane*. Pozostałe węzły to węzły *wewnętrzne*. Węzły niesformatowane znajdują się na najniższym poziomie, węzły sformatowane znajdują się na pierwszym poziomie, węzły wewnętrzne znajdują się na pozostałych poziomach.
- Informacje trzymane w drzewie są podzielone na wpisy. Rozmiar każdego wpisu jest mniejszy od rozmiaru węzła. W każdym węźle sformatowanym jest przynajmniej jeden wpis. Małe pliki składają się z jednego wpisu, duże pliki składają się z wielu wpisów.
- BLOB to skrót od *Binary Large Object*. W BLOB'ach trzymane są duże pliki.

3 Zawartość węzłów drzewa.

3.1 Węzeł niesformatowany.

Węzły niesformatowane nie zawierają metadanych. Węzły wewnętrzne zawierają tylko metadane. Węzły sformatowane zawierają metadane i być może dane które nie są metadanymi, ponieważ mogą zawierać małe pliki.

3.2 Liść.



- IH oznacza nagłówek wpisu.
- Item oznacza wpis.
- Block_Head oznacza nagłówek bloku.
- Pomiędzy n-tym nagłówkiem wpisu, a n-tym wpisem jest wolne miejsce.
- Dzięki takiej strukturze węzła sformatowanego można łatwo dodać nowy wpis tzn. nie ma potrzeby przemieszczania innych wpisów.

3.3 Wpis.

Wpisy dzielą się na:

- Wpisy bezpośrednie: zawierają małe pliki (mniejsze od rozmiaru węzła) oraz końcówki większych plików.
- Wpisy pośrednie: zawierają listę wskaźników do węzłów niesformatowanych.
- Wpisy katalogowe: zawierają listę kluczy pozycji w katalogu oraz ich nazwy.
- Wpisy informacyjne: zawierają dane, które system operacyjny przechowuje w strukturze *stat*, czyli np.: właściciel pliku, prawa dostępu, czas modyfikacji pliku, ilość bloków zajmowanych przez plik itd.

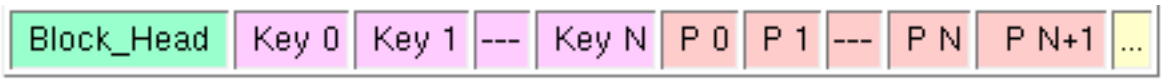
Nagłówek każdego wpisu zawiera m.in:

- Klucz.
- Długość wpisu.
- Offset wpisu wewnątrz bloku.

Ponadto każdy nagłówek wpisu pośredniego zawiera informację o tym ile jest wolnego miejsca w ostatnim węźle niesformatowanym. Ta informacja nie jest potrzebna, gdy końcówki dużych plików trzymane są we wpisach bezpośrednich, ale można wyłączyć pakowanie końcówek. Wtedy końcówki dużych plików są trzymane w częściowo wypełnionych węzłach niesformatowanych i miejsce na dysku jest mniej efektywnie wykorzystywane, ale szybciej wykonywane są operacje na drzewie.

Nagłówek katalogu zawiera informację ile pozycji w katalogu zawiera dany wpis katalogowy.

3.4 Węzeł wewnętrzny.



- Key oznacza klucz. Jeżeli chcemy znaleźć wpis w drzewie, to porównujemy klucze w węźle z kluczem wpisu, który chcemy znaleźć, aby się dowiedzieć w którym poddrzewie się on znajduje.
- P oznacza wskaźnik do poddrzewa.
- Na końcu bloku znajduje się wolne miejsce.

4 Typy danych.

4.1 Struktura key.

Nazwa	Rozmiar	Opis
Directory ID	4	Identyfikator katalogu, w którym znajduje się obiekt
Object ID	4	Identyfikator obiektu (i-węzeł)
Offset	4	Offset w pliku lub katalogu
Type	4	Typ wpisu. Możliwe wartości to: Stat: 0 Indirect: 0xFFFFFFFE Direct: 0xFFFFFFFF Directory: 500

Klucze są porównywane w ten sposób, że najpierw porównujemy pierwsze pola, jeżeli są równe, to porównujemy drugie pola itd.

Zalety takiego rozwiązania:

- Pliki znajdujące się w tym samym katalogu znajdują się blisko siebie.
- Wpisy należące do tego samego pliku znajdują się blisko siebie.
- Dane o pliku (takie jak np. właściciel pliku) znajdują się niedaleko pliku. Ponadto znajdują się one przed plikiem w porządku pre-order.
- Jeżeli plik składa się z wielu wpisów to można go całego przeczytać, przeglądając drzewo w porządku pre-order.

Dobrze jest, gdy węzły drzewa są czytane w porządku pre-order lub zbliżonym, ponieważ wtedy można to zrobić szybciej. Wynika to ze sposobu przydzielania węzłom drzewa bloków na dysku o czym jest mowa w dalszej części prezentacji. Dobrze, że dane o pliku znajdują się przed plikiem, ponieważ gdy np. czytamy plik, to najpierw musimy sprawdzić, czy mamy prawo do czytania.

4.2 Struktura block_head.

Nagłówek bloku zawiera m.in:

- Poziom tego bloku w drzewie.
- Ilość wolnego miejsca w bloku.
- Ilość wpisów w bloku.

4.3 Struktura disk_child

Struktura disk_child to wskaźnik do innego węzła. Zawiera ona numer bloku, w którym się znajduje wskazywany węzeł oraz ilość wolnego miejsca w tym bloku.

5 Położenie bloków na dysku.

Szybkość operacji wykonywanych na dysku zależy od tego jak są położone bloki, które chcemy odczytać lub zapisać. Dobrze jest, gdy bloki znajdują się blisko siebie, najlepiej na tej samej ścieżce i w odpowiedniej kolejności.

Realizacja tego celu odbywa się na czterech poziomach:

1. Przyporządkowanie logicznym numerom bloków fizycznych miejsc na dysku.
2. Przyporządkowanie węzłom drzewa numerów bloków.
3. Porządek obiektów w drzewie.
4. Operacje wykonywane na drzewie.

Przyporządkowanie numerom bloków miejsc na dysku jest takie, że możemy najszybciej czytać lub pisać gdy:

- Czytamy lub piszemy w kolejności rosnących numerów.
- Różnice pomiędzy kolejnymi numerami są małe.

Kiedy chcemy zapisać coś w drzewie i zaalokować nowy węzeł, to najpierw szukamy jego lewego sąsiada w drzewie, a następnie przeglądamy bitmapę zajętości bloków, zaczynając od tego węzła w kolejności rosnących bloków i szukamy pierwszego wolnego bloku. To rozwiązanie połączone ze sposobem przydzielania wpisom kluczy powoduje, że często węzły drzewa są czytane lub zapisywane w porządku zbliżonym do pre-order, więc dobra jest kolejność numerów czytanych lub zapisywanych bloków.

To rozwiązanie okazało się lepsze od następujących rozwiązań:

- Branie pierwszego wolnego bloku z bitmapy.
- Branie pierwszego wolnego bloku którego szukamy zaczynając od bloku, który ostatnio przydzieliliśmy i idąc w tym samym kierunku, w którym szliśmy ostatnio. Kierunek zmieniamy, gdy dojdziemy do końca bitmapy i nie znajdziemy wolnego bloku.
- Branie lewego sąsiada w drzewie i szukanie pierwszego wolnego bloku, przesuwając się w kierunku prawego sąsiada.

Inne rozwiązania są zbyt niezależne od struktury drzewa lub powodują, że bloki są czytane lub zapisywane w kolejności malejących numerów bloków, co jest mniej korzystne.

6 Księgowanie.

Kiedy chcemy wstawić wpis do drzewa i nie ma już miejsca w odpowiednim węźle, to trzeba stworzyć nowy węzeł, przekopiować część zawartości tego węzła, w którym nie było miejsca i utworzyć wskaźnik do nowego węzła. Jednak w węźle, który jest rodzicem nowego węzła może też nie być już miejsca i jego również trzeba podzielić. Może się tak zdarzyć, że trzeba będzie wstawić nowy wskaźnik do korzenia, ale w nim nie będzie miejsca, więc trzeba będzie utworzyć nowy węzeł, przekopiować część zawartości korzenia do nowego węzła, a następnie utworzyć nowy korzeń drzewa. Jeżeli tuż przed utworzeniem nowego korzenia nastąpiłaby awaria komputera (np. z powodu braku prądu), to moglibyśmy utracić ok.połowy informacji na dysku (ponieważ w korzeniu byłoby mniej wskaźników do poddrzew), gdyby nie mechanizm księgowania.

Księgowanie, to zapisywanie dodatkowych informacji na dysku po to, aby można było szybko zrestartować komputer po awarii i doprowadzić system plików do stanu spójnego, bez utraty danych. Miejsce w którym zapisywane są dodatkowe dane, to *dziennik*.

ReiserFs jest systemem plików stosującym księgowanie, więc:

- Można szybciej zrestartować komputer po awarii, nie trzeba korzystać z *fsck*. *Fsck* to program, który przegląda wszystkie metadane na dysku i stara się je naprawić. Nie zawsze jest to możliwe, poza tym działa długo tzn. od kilku minut do kilku godzin w zależności od wielkości i ilości dysków (po awarii trzeba sprawdzić wszystkie dyski).
- Można mieć pewność, że metadane będą w stanie spójnym (księgowanie w ReiserFs w wersji 3 dotyczy tylko metadanych).

- System trochę wolniej działa.
- Jest trochę mniej miejsca na dysku. Trzeba pewną część dysku przeznaczyć na dziennik. Ilość tego miejsca musi być taka, żeby rozwiązać konflikt pomiędzy szybkością działania systemu, a szybkością restartowania systemu. Jeżeli dziennik jest mały to jest konieczność częstego zwalniania miejsca w dzienniku, jeżeli jest duży, to dłużej trwa zrestartowanie systemu.

Mechanizm księgowania stosowany w ReiserFs w wersji 3 to *księgowanie w ustalonym miejscu*. Polega to na tym, że zanim dane zostaną wpisane do drzewa, są zapisywane do dziennika. Jeżeli w czasie tego zapisywania nastąpi awaria, to nic się nie stanie tzn. nie zostanie wykonana żadna operacja na drzewie. Jeżeli awaria nastąpi po zapisaniu odpowiednich informacji do dziennika, to czytając jego zawartość można dokończyć wykonywaną operację na drzewie.

7 Podsumowanie.

Cechy ReiserFs w wersji 3:

- Efektywne wykorzystanie miejsca na dysku dla małych plików.
- Efektywne wykorzystanie miejsca na dysku dla końcówek plików.

Wiąże się to z wolniejszym działaniem systemu. Jeżeli piszemy do pliku, to nie wiemy jak dużo zapiszemy i na wszelki wypadek zakładamy, że potrzebne jest nam miejsce na duży plik. Kiedy skończymy pisać, to w przypadku małego pliku musimy dokonać zmian w drzewie. Pakowanie końcówek plików można wyłączyć. Jeżeli wyłączymy pakowanie końcówek plików, to miejsce na dysku będzie mniej efektywnie wykorzystywane, za to system będzie działał szybciej.

- Efektywne działanie systemu, dla katalogów zawierających dużo plików, dzięki zastosowaniu drzew zrównoważonych.
- Możliwość traktowania systemu plików jak bazy danych, np. możemy trzymać na dysku 100000 numerów telefonów, każdy w oddzielnym pliku.
- Jeżeli końcówka pliku urośnie, to węzeł sformatowany trzeba przekształcić w węzeł niesformatowany.
- Zapisywanie danych w końcówkach się wiązać z koniecznością rozbicia węzła na dwa węzły.
- Jednolitość w traktowaniu plików, katalogów i informacji o plikach tzn. różne obiekty są tak samo wstawiane (usuwane) do (z) tej samej struktury danych.
- Efektywne księgowanie.
- Brak zalet wynikających z zastosowania nowych rozwiązań z wersji 4.

8 Zmiany wprowadzone w wersji 4.

8.1 Ekstenty

BLOB'y zastąpiono ekstentami. Ekstent to ciągły fragment dysku, dane w nim zawarte należą do jednego pliku. W ekstencie trzymane są węzły niesformatowane.

Wskaźnik do ekstentu zawiera:

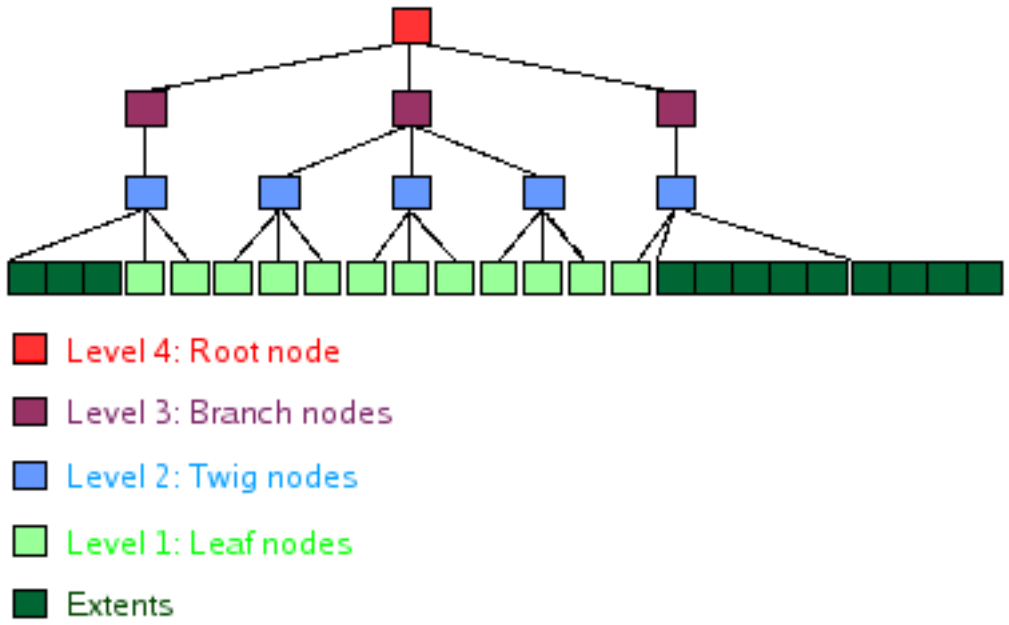
- Numer początkowego bloku.
- Długość ekstentu.

Dzięki ekstentom można:

- Zwiększyć rozgałęzienie drzewa. Aby opisać plik, który jest opisany poprzez ekstenty, potrzeba mniej wskaźników.
- Sprawić, aby drzewo było bardziej zrównoważone, ponieważ ekstenty są na tym samym poziomie, co liście.

Wynika z tego, że:

- Dostęp do danych, jest szybszy, ponieważ krótsza jest średnia droga z korzenia do węzła na najniższym poziomie.
- Jest mniejsza ilość węzłów wewnętrznych, co ułatwia buforowanie danych. Łatwiej jest zmieścić wszystkie węzły wewnętrzne w pamięci operacyjnej.



8.2 Księgowanie

- Księgowanie dotyczy nie tylko metadanych, jest więc wolniejsze, ale system i tak działa szybciej, dzięki zastosowaniu ekstentów.

- Wykorzystywane są dwie metody księgowania: księgowanie w ustalonym miejscu i przy użyciu wędrujących dzienników (wandering logs).

Metoda wędrujących dzienników polega na tym, że gdy wykonujemy jakąś operację np. piszemy do pliku, to dane są zapisywane w pewnym miejscu na dysku, a następnie nie są zapisywane w drzewie tak jak w metodzie księgowania w ustalonym miejscu, tylko w drzewie zapisywane są informacje, które mówią o tym, że to miejsce, które było traktowane jako dziennik jest teraz miejscem w którym są nowe dane. Wiąże się to z pewnymi komplikacjami, ponieważ trzeba uaktualnić wskaźniki do nowych danych i trzeba to zrobić również w sposób pozwalający na szybkie naprawienie systemu plików w razie awarii.

Metoda wędrujących dzienników jest zwykle szybsza, ponieważ nie jest konieczne dwukrotne zapisywanie tych samych danych.

8.3 Alokowanie nowych bloków.

W systemie ReiserFs w wersji 4 nowe bloki nie są alokowane wtedy, gdy wykonywana jest operacja *write*. System operacyjny buforuje dane, i gdy zaczyna brakować pamięci, dane na dysku są aktualizowane, aby można było zwolnić pamięć.

System plików przydziela wolne bloki dopiero wtedy, gdy jest zmuszony zapisać coś na dysk. Umożliwia to:

- Lepszy wybór bloków do zaalokowania. Możemy tworzyć ekstenty odpowiedniej wielkości.

- Wybór lepszej metody księgowania. Wiemy, które bloki chcemy zapisać.

- Rozwiązanie konfliktu pomiędzy szybkością usuwania elementów z drzewa, a efektywnym wykorzystaniem miejsca na dysku.

W systemie plików ReiserFs w wersji 3, gdy usuwamy wpis z drzewa, to patrzymy na węzeł z którego usuwamy i na jego obu sąsiadów w drzewie, aby sprawdzić, czy nie dałoby się z tych trzech węzłów zrobić dwóch lub jednego węzła, ponieważ chcemy się dowiedzieć, czy możemy zwolnić miejsce na dysku. Nie jest oczywiste, czy jest to optymalne rozwiązanie, bo możnaby patrzeć na węzeł, z którego usuwamy i na np. pięciu jego sąsiadów po lewej stronie i pięciu po prawej i moglibyśmy sprawdzać, czy nie dałoby się z tych jedenastu węzłów zrobić mniejszą ilość węzłów. Wtedy miejsce na dysku byłoby efektywniej wykorzystywane, ale dłużej trwałoby usuwanie wpisów.

W wersji 4, gdy usuwamy wpis, to zwalniamy miejsce na dysku tylko wtedy, gdy jest to ostatni wpis w węźle. Wpisy są pakowane przy innej okazji tzn. kiedy system plików jest zmuszony zaktualizować dane na dysku. Wtedy w pamięci operacyjnej znajduje się wiele węzłów

i można efektywnie upakować wpisy w nich zawarte, nie trzeba wykonywać niepotrzebnych i czasochłonnych operacji czytania bloków z dysku.

Ponieważ w inny sposób usuwamy z drzewa, więc nie jest to już B+ drzewo. Jest ono nazywane „tańczącym drzewem” („dancing tree”).

8.4 Program pakujący.

Częścią systemu plików ReiserFs jest program pakujący (repacker).

- Defragmentuje dysk.
- Pakuje wpisy tak, aby były gęsto upakowane w węzłach.
- Uruchamia się co jakiś czas, w czasie mniejszej aktywności systemu.

8.5 Więcej informacji:

- http://www.namesys.com
- http://www4.informatik.uni-erlangen.de/Lehre/WS01/PS_KVBK/docs/reiserfs/MosheBarReiserFS.pdf