Uniksowe systemy plików

„Nigdy nie mogłem zrozumieć,
co znaczą te przeklęte kropki”
— *Winston Churchill*

Plan prezentacji

Wstęp czym jest system system plików, czym dziennikowanie

VFS Wirtualny System Plików

Lokalne systemy plików przegląd

ReiserFS coś godnego uwagi

Podsumowanie krótkie porównanie omówionych lokalnych systemów plików

Sieciowe systemy plików przegląd

System plików – co to jest?!?

system plików – sposób organizacji danych na nośniku pamięci trwałej.

- rozmieszczenie danych, umieszczenie ich w jakiejś strukturze danych
- utrzymanie informacji o danych, takich jak ich rozmiar, czas utworzenia lub modyfikacji itd.
- często dbanie o prawa dostępu do danych

Dziennikowanie

- *clean bit*, wyłączanie komputera
- awarie -> niespójności, brudny *cleanbit*:
  - banalna przerwa w dostawie prądu
  - wyjątkowo poważny błąd systemowy
  - królik przegryza kabel
- niespójności -> fsck – długotrwałe sprawdzanie
- dziennikowanie:
  - dziennik zmian powiązanych w transakcje
  - przepisywanie zmian na dysk
  - awaria -> odtworzenie zmian z dziennika

VFS

Virtual File System

Historia

- *Unix SVR3* – FSS *(File System Switch)*
- *SunOS* – VFS *Virtual File System*

FSS + Sun VFS = SVR4 VFS

Struktury VFS

- VFSSW – *Virtual Filesystem Switch Table*
- struktura *vfsops* – operacje zależne od typu systemu plików
- struktura *ustat* i *statvfs* – informacje o systemie plików

VFSSW

Tablica rozdzielcza wirtualnego systemu plików jest strukturą jądra systemu, która przechowuje po jednym wpisie dla każdego obsługiwanego rodzaju systemu plików.

Element	Opis
*char *vsw_name*	napis zawierający nazwę typu systemu plików
*int (*vsw_init)()*	wskaźnik do funkcji inicjowania systemu plików
*struct vfsops *vsw_vfsops*	wskaźnik do tablicy operacji zależnych od typu systemu plików

vfsops

Operacje zależne od typu systemu plików *(fragment)*:

Element	Związane makro	Opis operacji
int (vfs_mount)()	*VFS_MOUNT*	zamontowanie systemu plików
int (vfs_unmount)()	*VFS_UNMOUNT*	odmontowanie systemu plików
int (vfs_root)()	*VFS_ROOT*	odczytanie v-węzła dla korzenia systemu plików
int (vfs_statvfs)()	*VFS_STATVFS*	odczytanie informacji statycznych o systemie plików
int (vfs_sync)()	*VFS_SYNC*	zapisanie zawartości buforów dyskowych na dysku
int (vfs_vget)()	*VFS_VGET*	znalezienie v-węzła odpowiadającego identyfikatorowi pliku
int (vfs_mountroot)()	*VFS_MOUNTROOT*	zamontowanie głównego systemu plików

Informacje o systemie plików

Struktura *ustat*:

Element	Opis
daddr_t f_tfree	łączna liczba wolnych bloków
o_ino_t f_tinode	łączna liczba wolnych i-węzłów
char f_fname[6]	napis zawierający nazwę systemu plików
char f_fpack[6]	napis zawierający nazwę woluminu systemu plików

Informacje o systemie plików

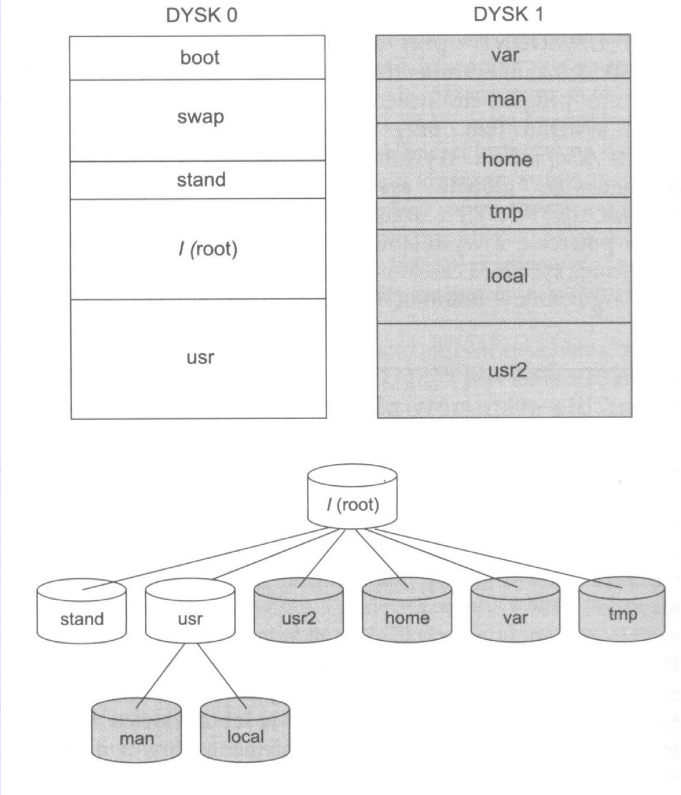
Struktura *statvfs (fragment)*:

Element	Opis
u_long f_bsize	rozmiar bloku systemu plików
u_long f_frsize	rozmiar fragmentów (jeśli są obsługiwane)
u_long f_blocks	łączna liczba bloków w systemie plików (jednostką jest *f_frsize*)
u_long f_bfree	łączna liczba wszystkich wolnych bloków w systemie plików (jednostką jest *f_frsize*)
u_long f_ffree	łączna liczba wolnych i-węzłów
u_long f_favail	liczba wolnych i-węzłów dostępnych dla zwykłego użytkownika
char f_basetype[FSTYPSZ]	napis zawierający nazwę typu systemu plików
u_long f_namemax	maksymalna długość nazwy pliku

V-węzły

- Jeden v-węzeł odpowiada w VFS każdemu otwartemu plikowi
- V-węzeł przechowuje wskaźnik na odpowiedni i-węzeł
- Wszystkie operacje na pliku są wykonywane przez jego v-węzeł

Hierarchia plików



Odwzorowywanie ścieżek

- Struktura *pathname*
- Funkcja *lookupname()*
- Funkcja *lookuppn()*

Algorytm funkcji *lookuppn()*

zacznij poszukiwanie od bieżącego katalogu

if (pierwszym znakiem ścieżki jest '/') {
usuń ukośnik ze ścieżki
zacznij przeszukiwanie od katalogu głównego
}

loop {
if (długość ścieżki wynosi zero) {
wybierz bieżący katalog
return success
}

if (VOP_LOOKUP() == *error*)
return error

while (*v_vfsmountedhere* != *NULL*)
przejdź do korzenia tego systemu plików
if (ten człon jest dowiązaniem symbolicznym) {
utwórz nową strukturę *pathname*
odczytaj dane z tego dowiązania i zapisz je w *pathname*
next loop

VFS a konkretny system plików

Jądro jest niezależne od konstrukcji konkretnych systemów plików:

- Łatwość implementacji operacji niezależnych od konkretnych systemów plików, takich jak *df*
- Konieczność tworzenia osobnych narzędzi, np. do formatowania

vfs_sync

- Funkcja systemowa *sync* otrzymuje znacznik, jakiego rodzaju synchronizacja jest żądana.
- Demon *fsflush* co sekundę wykonuje *częściową operację zapisania danych na dysku*.
- *Pełne zapisanie danych* jest wykonywane na żądanie, np. poleceniem *sync*.

Lokalne systemy plików

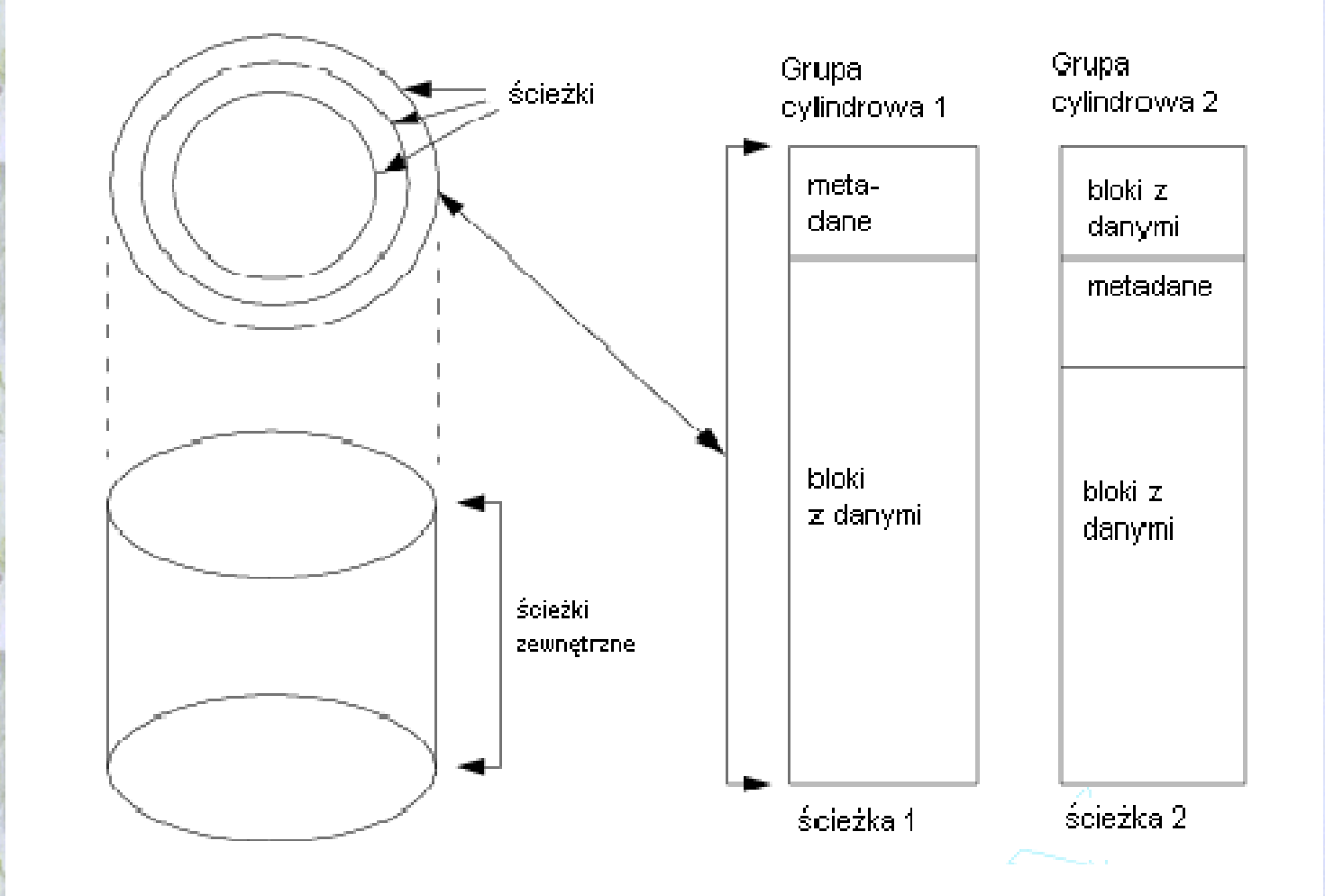
- UFS - jeden z pierwszych systemów plików
- VxFS - komercyjny system plików
- Ext2 i jego następca Ext3
- XFS - bardzo wydajny
- ReiserFS - najnowsza wersja

UFS a struktura dysku

UFS znany jako Berkley Fast File System

- *Grupy cylindrowe* -> cylindry dyskowe

- W grupie cylindrowej znajdują się:
  - kopia superbloku
  - stała liczba i-węzłów (1 i-węzeł / 2096 b)
  - bitmapy opisujące wolne bloki i i-węzły
  - dane dot. zużycia i-węzłów
  - bloki z danymi


ścieżki
Grupa
cylindrowa 1
Grupa
cylindrowa 2
meta-
dane
bloki z
danymi
metadane
bloki
z danymi
bloki z
danymi
ścieżki
zewnętrzne
ścieżka 1
ścieżka 2

UFS - ciekawostki

- podział bloków na mniejsze 2, 4 lub 8 części zwane *fragmentami*
- UFS dziś -> firma Sun (Solaris)
  - księgowanie
  - wsparcie dla baz danych

VxFS

Rozwój systemu plików VERITAS wraz z implementacją SVR4.0

ekstenty - przyległe do siebie bloki, do których alokowany jest jeden plik

Algorytm alokowania ekstentów dla pliku - mechanizm wejścia/wyjścia. Stosowane prealokowanie - biorę więcej niż potrzebuję.

zbiory plików: zbiór główny i strukturalny

VxFS - wszędzie pliki

- Tablica lokacji (*Object location table* - OTL)
- Etykieta (superblok i jego kopie)
- Plik nagłówkowy dla zbioru plików
- Lista i-węzłów
- Jednostka alokacji i-węzłów
- Dziennik
- Pliki zawierające informacje o ekstentach

VxFS - reklama :)

- natychmiastowe odmontowanie
- administracja online
- minimalizacja informacji alokowanych przy inicjowaniu systemu (32 i-węzły na początek)

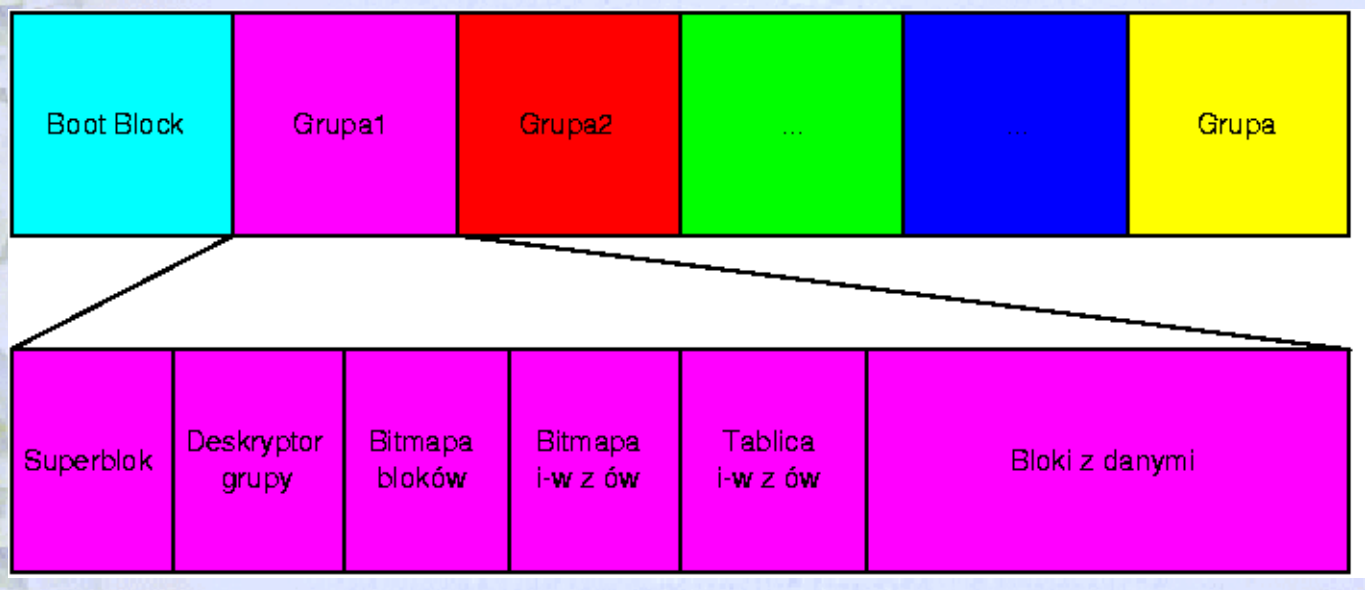
System EXT

Historia Ext3:

1. Minix - system plików do 64MB
2. Ext - listy - zarządzanie wolnymi blokami i i-węzłami
3. Xia i Ext2
4. Ext2 lepszy :-) - standard Linuxa
5. Ext3 - ulepszony ext2!!!

EXT2 - deskryptory, bitmapy, super !

Struktura danych na dysku w ext2 i ext3:



EXT3 = EXT2 + JBD

Warstwa księgowania (*Journal Block Device* JBD).

Ext3 dostarcza trzy tryby księgowania:

- *journal* - dane i metadane
- *ordered* - metadane po zapisaniu na dysk, domyślny
- *writeback* - metadane (możliwość niespójności)

EXT3 - dodatki

- indeksowany format katalogu
- możliwość rezerwacji przestrzeni dyskowej dla administratora
- dwie semantyki dostępu do pliku, BSD lub SVR4

XFS

- księgowanie, transakcyjność operacji
- obsługa dużych plików i dużej liczby plików
- użycie B+drzew

XFS - dane na dysku

System plików podzielony na grupy alokacji (ang. *Allocation Groups* - AG). AG to autonomiczne jednostki systemu plików.

W skład struktur grupy alokacji wchodzą:

- superblok
- struktura wolnych obszarów
- drzewo i-węzłów
- wykaz bloków zajmowanych przez drzewo wolnych obszarów

XFS - co innego?

Księgowanie w XFS jest asynchroniczne. Najpierw bufor dziennika potem dysk.

Listy uprawnień - prawa dostępu dla Baltazara Gąbki.

Reiser File System

- jeden z najbardziej popularnych systemów plików
- ojcem Hans Reiser
- obecnie ReiserFS 3, w fazie zaawansowanych, ostatnich testów Reiser4

Dlaczego Reiser

testy jeden z najszybszych systemów plików

dziennikowanie teraz ulepszone

system atomowy operacje wykonują się w całości albo wcale

tańczące drzewa szybkie drzewa zrównoważone

ściskanie małych plików daje dużą wydajność pamięciową

wtyczki modularność

bezpieczeństwo militarne

Warstwy

warstwa semantyczna nazywanie obiektów, decydowanie, co z nimi zrobić

warstwa przechowywania ulokowanie obiektów na dysku, w strukturze danych

Pliki

wtyczka dla plików wyłapuje interakcje z plikami, zbiór metod

metoda sposób interakcji z wtyczką

identyfikator wtyczki określa typ pliku i zbiór operacji, odszukiwany przy próbie interakcji z plikiem

tablica wtyczek przy pomocy identyfikatora odszukiwana wtyczka

Katalogi

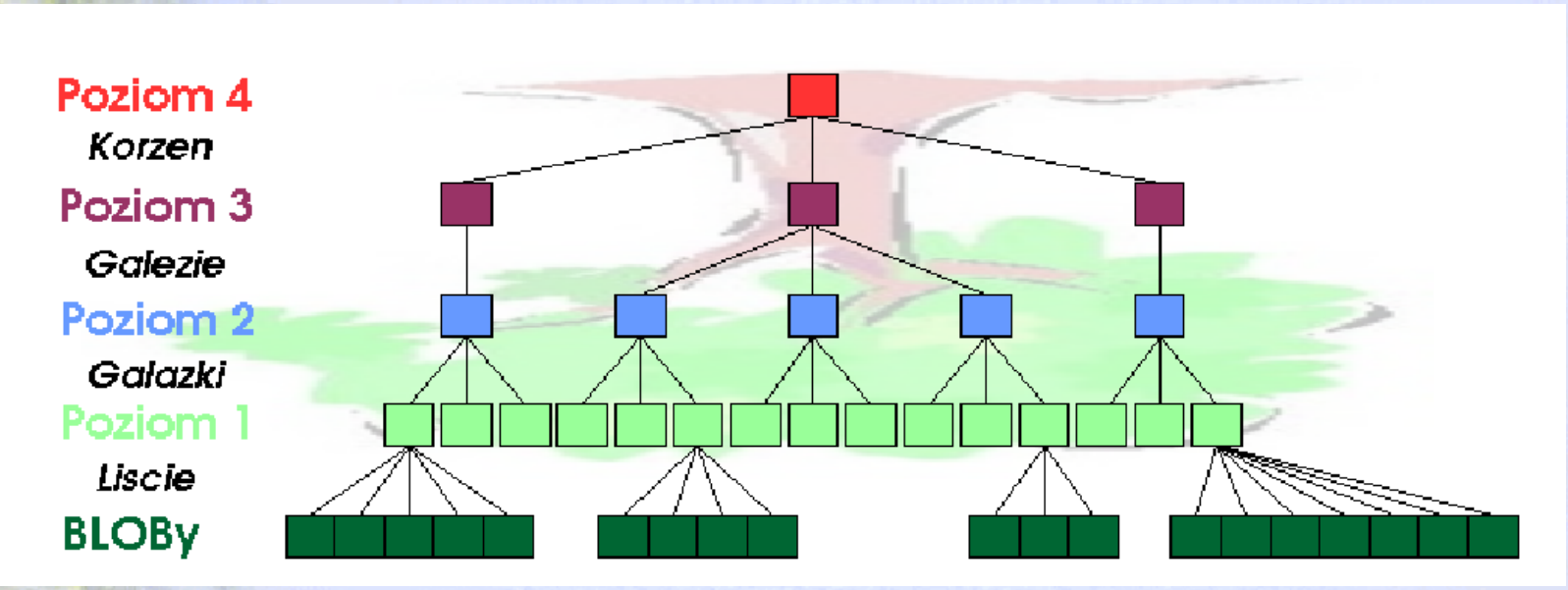
katalog zbiór wpisów, które zawierają nazwę i klucz

obiekt jako plik i katalog jednocześnie podstawa do zbudowania funkcjonalności obecnej w strumieniach

wtyczka plikowa dla ciała pliku

wtyczka katalogowa do odszukiwania wtyczek plikowych do obsługi metadanych

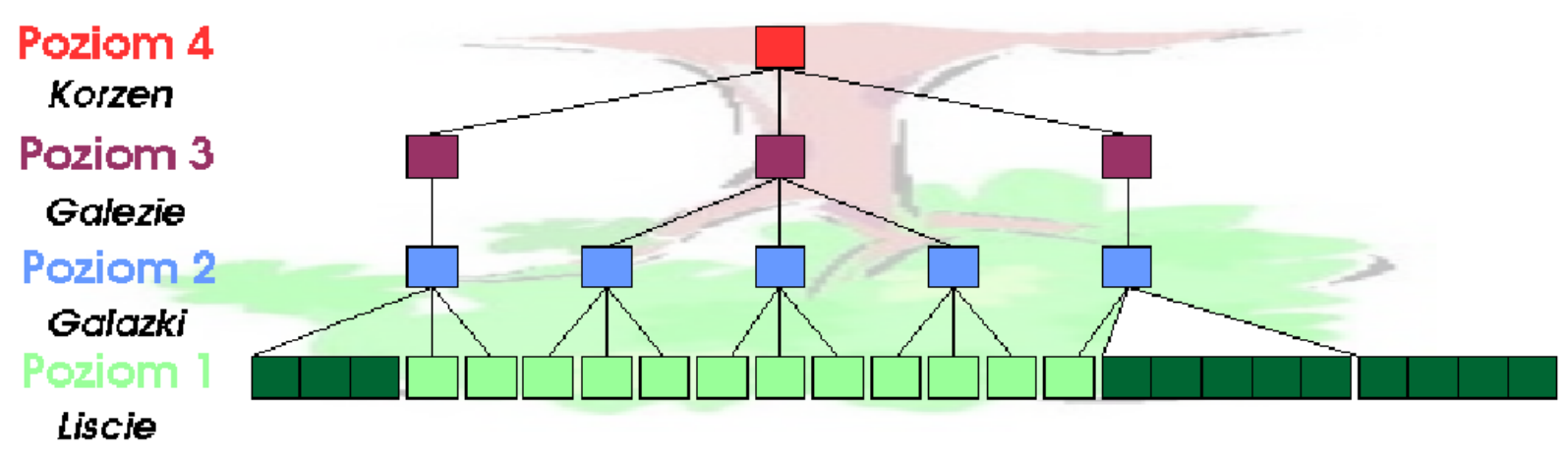
Drzewo w ReiserFS 3



B+drzewa dane w liściach

BLOB w liściach wskaźniki do węzłów z obiektem

Drzewo w Reiser4



Powrót do klasycznej definicji drzewa zrównoważonego

Gałęzie, gałązki, liście

sformatowany liść *z ang. formatted leaf* przechowuje dane, zawiera tzw. *items*

Nagłówek węzła	Wpis 0	Wpis 1	...	Wpis n	Wolne miejsce	Nagłówek wpisu 0	Nagłówek wpisu n	...	Nagłówek wpisu 0

niesformatowany liść *z ang. unformatted leaf*

- zawiera blok rozszerzony
- *wskaźniki rozszerzone*: numer pierwszego bloku rozszerzonego i jego długość

.....

Gałęzie, gałązki, liście

węzły wewnętrzne *z ang. internal nodes* wskaźniki do poddrzew oddzielone kluczami. Klucz równy pierwszemu kluczowi w pierwszym węźle sformatowanym poddrzewa, na które wskazuje wskaźnik.

- gałązki *twigs* rodzice liści

Nagł. węzła	Wpis 0 Wsk. 0	Wpis 1 Rozsz. wsk. 1	Wpis 2 Wsk. 2	Wpis 3 Rozsz. wsk. 3	.	Wpis n Wsk. n	Wolne miejsce	Nagł. wpisu n	.	Nagł. wpisu 0

- gałęzie *branches* pozostałe węzły.

Nagłówek węzła	Wpis 0 Wskaźnik 0	...	Wpis n Wskaźnik n	Wolne miejsce	Nagłówek wpisu n	...	Nagłówek wpisu 0

Drzewa w Reiserze mają minimalną wysokość 2.

Rozmiar węzłów

- taki sam dla każdego węzła
  - łatwa alokacja
  - wygoda dla dyskowych algorytmów korekcji błędów
- równy rozmiarowi strony
- krojenie na kawałki
- przechowywanie plików w całych blokach – marnowanie miejsca, wydłużenie czasu operacji wejścia-wyjścia – Reiser upycha
- wyrównywanie dobre dla dużych plików – mmap() – Reiser wyrównuje pliki większe niż 16k

Rozdrabnianie

Zawartość (ciało)	...	Etykietka (nagłówek)			
		Klucz	Offset	Długość	Identyfikator wtyczki

Rodzaje pojemników

informacyjny (*static_stat_data*) właściciel, prawa dostępu, czas ostatniego dostępu... (jend.: wszystkie informacje)

katalogowy (*cmpnd_dir_item*) wpisy katalogowe, klucze do plików, których wpisy dotyczą (jedn.: wpis)

pośredni (wskaźnikowy) (*pointers, extent pointers*) lista wskaźników (zwykłych lub rozszerzonych) do bloków na dysku

bezpośredni (*bodies*) części plików, które nie są wystarczająco duże, aby je przechowywać w niesformatowanych liściach (jedn.: bajt)

Jednostka – w całości w pojemniku, nie możemy jej rozlać do wielu pojemników

Tańczące drzewa

Zapisywanie w ostatniej chwili

Reiser4 jako atomowy system plików

- strata, gdy coś złego stanie się z komputerem, podczas gdy w RAMie mamy dane, które nie dotarły na dysk
- nie zawsze opłaca nam się zachowywać to, co już zostało zapisane, np. transfer 10 zł = potrącenie + dopisanie
- atom – zbiór operacji: wszystkie albo żadna
- *wywołania systemowe* są w Reiserze atomowe
- Reiser pozwala denifiować nowe atomowe operacje (wtyczki)
- używa specjalnych algorytmów, które pozwalają uczynić operacje atomowymi przy niewielkim dodatkowym koszcie

Przepychacz

- 80% danych na dysku pozostaje nie zmienianych przez długi okres czasu
- wydajnie by było, gdyby były dobrze rozmieszczone
- Reiser4 oferuje przepychacz, z ang. *repacker*
  - przechodzi drzewo od lewej do prawej, spychając wszystko w lewo i od prawej do lewej, spychając w prawo
  - defragmentuje drzewo i upycha dane

Wtyczki

- plikowe
- katalogowe
- haszujące
- bezpieczeństwa
- obsługi rozdrabniania obiektów
- przypisywania kluczy
- wyszukiwania węzłów

Podsumowanie. Który lepszy?

1TB = 1024 GB = 2^{40} B

System plików	UFS	VxFS	Ext3	XFS	Reiser4
Maks. rozmiar systemu plików		32TB	4TB	18 mln. TB	16 TB
Maks. rozmiar pliku		2TB	2 GB	9 mln. TB	16 TB
Rozmiar bloku	4KB, 8KB	1KB, 2KB, 4KB, 8KB	1KB, 2KB, 4KB	512B - 64KB	4KB
Księgowanie	Nie	Tak	Tak	Tak	Tak

System plików	UFS	VxFS	Ext3	XFS	Reiser4
Dynamiczna alokacja i-węzłów	Nie	Tak	Nie	Tak	Tak
Ekstenty	Nie	Tak	Nie	Tak	Tak
Struktury do zarządzania wolnym miejscem	Bitmapa	Bitmapa	Bitmapa	B+-drzewa	B+-drzewa

Trochę praktyki

- Dwa testy pokazujące szybkość działania operacji *write*.
- Usuwanie plików w Uniksie (funkcja *unlink*).
- Implementacja „Kosza” – biblioteka *libtrash*.

Testy operacji *write*

Przykładowe wyniki pierwszego testu:

ext2: 0.45s user 2.49s system 20% cpu 14.518 total

reiserfs: 0.72s user 4.11s system 32% cpu 14.831 total

Przykładowe wyniki drugiego testu:

ext2: 1.60s user 14.82s system 86% cpu 19.064 total

reiserfs: 1.59s user 3.87s system 68% cpu 8.012 total

Demonstracja działania *unlink*

Wniosek:

Faktycznie plik jest usuwany dopiero po zakończeniu naszego programu.

Linuksowy „Kosz”

Zalety:

- Bardzo łatwa instalacja, bez rekompilacji jądra
- Przezroczyste działanie
- Możliwości konfiguracji (wybór katalogów)

Linuksowy „Kosz” c.d.

Zasadnicza wada:
Ponieważ *libtrash* nie modyfikuje jądra, nie działa w przypadku bezpośrednich wywołań funkcji systemowych.
Dlatego nie ma 100% pewności, że usunięty plik znajdziemy w koszu.

Użyjmy dłuższego kabla – sieciowe systemy plików

- Protokoły sieciowe (RPC i XDR)
- RFS
- NFS
- Coda
- Podsumowanie
- Przyszłość

Każdy może dać się wykorzystać, czyli RPC

RPC – *Remote Procedure Call*

- Najpopularniejsza wersja: SunRPC
- Zastosowania:
  - NIS (Network Information Server)
  - XML-RPC (transport: http, dane: XML)

RPC: zrób to sam

1. Dostarczamy specyfikację naszego protokołu (.x).

2. Zatrudniamy program `rpcgen`.

3. Otrzymujemy:

   - pliki nagłówkowe dla serwera i klienta
   - kod dla serwera
   - bibliotekę dla klienta
   - przykładowe programy

Zadanie domowe: ściągnąć specyfikację NFSv4, wygenerować co trzeba i zaimplementować protokół.

Sieciowa Wieża Babel – pomoże XDR

XDR – *Extendend Data Representation*

Przykłady:

- Liczby całkowite – bajt skrajnie lewy jest najbardziej znaczący.
- Napisy – długość + wyrównana do 4 bajtów treść.

Trochę komercji, czyli RFS

RFS – *Remote File Sharing*

- Autor: AT&T.
- Cel: stworzyć środowisko sieciowej pracy dla dystrybucji Uniksa dzieła AT&T.

Problemy z licencjonowaniem (RFS nie jest publicznie dostępny) i przenośnością.

Mimo wszystko lubimy RFS, gdyż: ma wiele korzystniejszych niż NFS rozwiązań (semantyka, bufor).

Charakterystyka RFS

- Z założenia: pełna zgodność z semantyką Uniksa
  - Serwer pamięta środowisko klienta
  - Buforowanie danych
  - Dostęp do plików specjalnych
- Wyznaczony serwer zarządzający zasobami
- Sieć podzielona na domeny
- Komunikacja oparta na obwodach wirtualnych

RFS vs NFSv3: `mount`

RFS	NFSv3
Klient odpytuje serwer nazw o lokalizację zasobu.	Użytkownik sam określa nazwę serwera.
Klient łączy się z odpowiednim serwerem (jeśli potrzeba ustanawia obwód wirtualny).	Klient łączy się z odpowiednim serwerem.
Serwer weryfikuje i potwierdza operację.	Serwer weryfikuje i potwierdza operację.
Klient zapamiętuje wskaźnik do obwodu wirtualnego, nazwę serwera i zasobu.	Klient zapamiętuje adres sieciowy serwera i uchwyt do katalogu.

RFS vs NFSv3: `open`

RFS	NFSv3
Analiza nazwy ścieżkowej.	Analiza nazwy ścieżkowej.
Serwer tworzy deskryptor i włącza go do środowiska klienta.	Serwer tworzy uchwyt do pliku.
Serwer otwiera plik.	
Do klienta wraca deskryptor i numer wersji pliku.	Do klienta wraca uchwyt.
Klient sprawdza świeżość swojego bufora.	

RFS vs NFSv3: `write`

RFS	NFSv3
Klient wysyła zlecenie zapisu (przesyła deskryptor).	Klient wysyła zlecenie zapisu (przesyła uchwyt).
Serwer tłumaczy deskryptor danego klienta na odwołanie do konkretnego pliku.	Serwer wykorzystuje uchwyt do odnalezienia pliku.
	Serwer otwiera plik w imieniu klienta, sprawdza prawa dostępu.
Serwer powiadamia innych.	
Serwer wykonuje zapis.	Serwer wykonuje zapis.
Do klienta wysyłane jest potwierdzenie.	Do klienta wysyłane jest potwierdzenie.

Przewaga RFS nad NFS

- Semantyka Uniksowa
- Niezawodny bufor
- Łatwiejsza autoryzacja

Świeży powiew, czyli Projekt NFSv4

- Sun przekazał pracę nad protokołem Internet Engineering Task Force (IETF)
- Najważniejsze cele projektowe dla NFSv4:
  - szybkie działanie w sieciach rozległych
  - zwiększenie bezpieczeństwa
  - modularna budowa
  - kompatybilność wstecz
  - praca w dowolnym systemie operacyjnym

Projekt trwa. Specyfikacja jest w trakcie opracowywania (najnowsza wersja: RFC3530 z 04.2003).

Co nowego w NFSv4?

- stanowy serwer
- blokady
- delegowanie
- operacje złożone (ang. compound)
- zrezygnowanie z operacji `mount`
- bezpieczeństwo (RPCSEC_GSS)
- migracje i replikacje

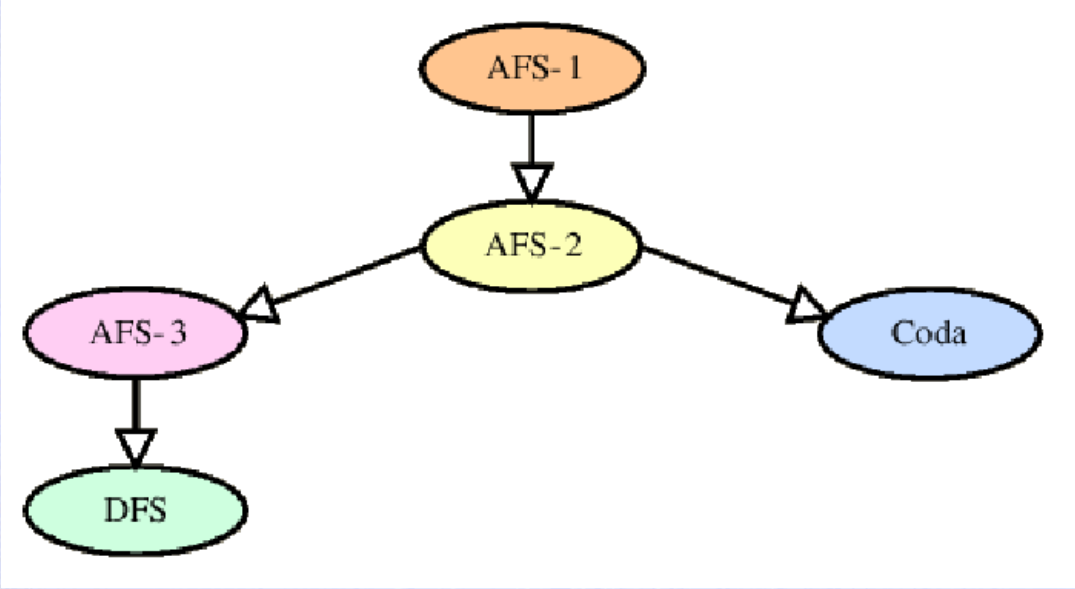
NFSv4 pierze i sprząta, czyli atak na Internet

- zastąpi ftp
  - łatwe uwierzytelnianie
  - bezpieczeństwo przesyłanych danych
- zastąpi e-dyski
- umożliwi tworzenie internetowych kopii zapasowych

A przede wszystkim... jako środowisko pracy w dużych sieciach

Poznajmy rodzinę AFS

Wszystko zaczęło się w Carnegie-Mellon University w Pittsburghu w Penslwanii. Najpierw powstał Andrew's File System. Projekt był przejmowany przez różne firmy, rozwijany...



Aż wreszcie wymyślono Codę.

Mahadev Satyanarayanan = wczoraj i dziś Cody

Projekt Coda został zapoczątkowany w 1987r. na CMU. Opieką nad nim zajmuje się prof. M.Satyanarayanan.

Zaczęło się od Macha, potem przeniesiony na NetBSD, FreeBSD, Linuxa oraz Windows 95.

Projekt żyje – najnowsze wydanie pochodzi z 17.10.2003r.

„Załóżmy, że...” czyli poznajemy Codę

Załóżmy, że:

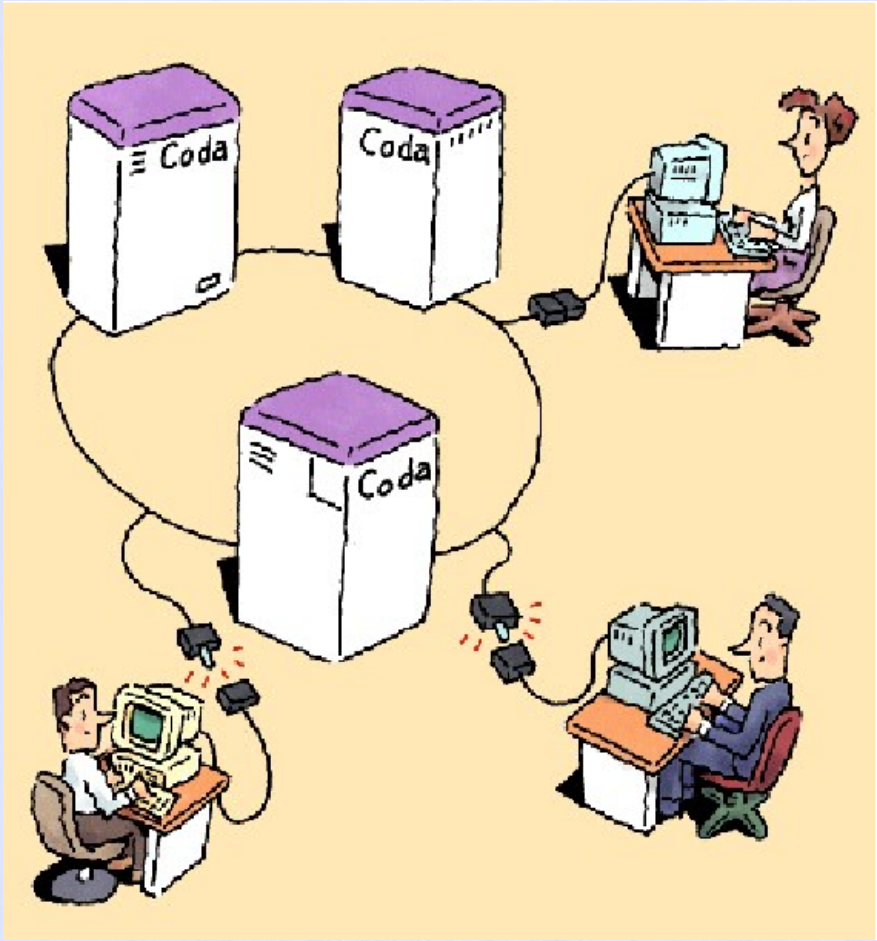
- nad jednym plikiem pracuje w danym momencie co najwyżej jeden użytkownik,
- współdzielone pliki są zazwyczaj niewielkie,
- użytkownik chce pracować niezależnie od awarii serwera i bieżącego położenia swoich plików,

a otrzymamy koncepcję Cody.

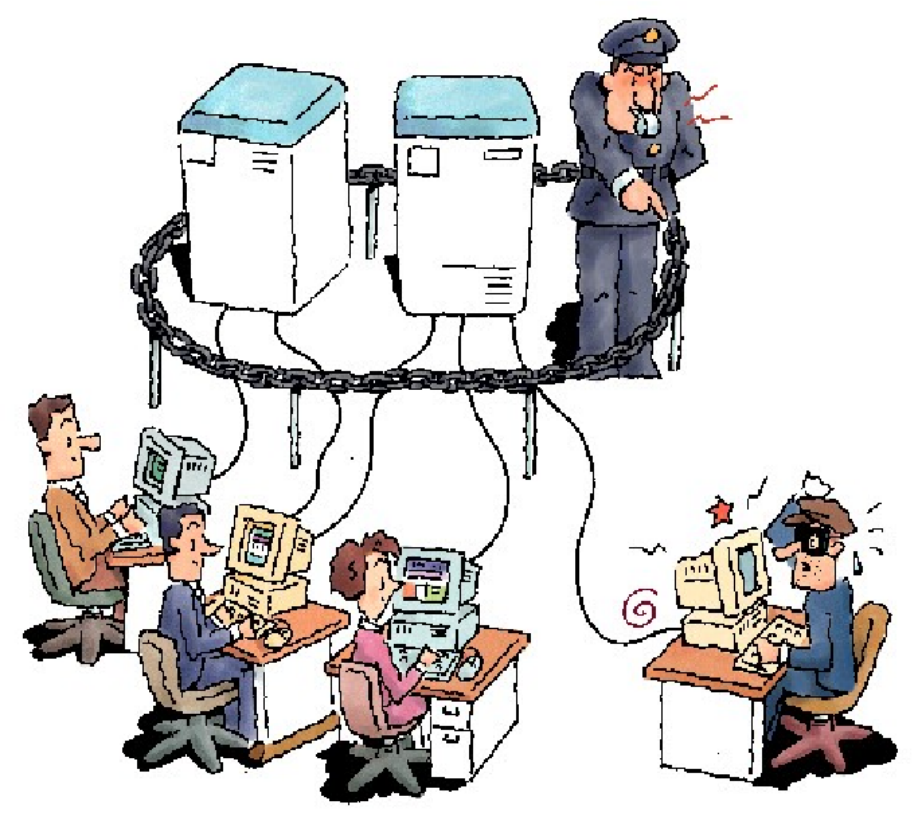
Architektura Cody

Serwery Klient Wektory wersji

W Codzie możemy pracować off-line



Coda dba o bezpieczeństwo danych



Kerberos

ACL (Access Control List)

Plamy na honorze Cody

Coda sprawdza się, o ile nie korzystamy z jej ciekawych funkcji.

- Wydajność
  - lepsza niż przy NFS wydajność przy pojedynczym serwerze
  - drastyczny spadek wydajności przy wielu serwerach
- Konflikty wersji

Czy je jeszcze pamiętasz? (podsumowanie)

	RFS	NFSv3	NFSv4	Coda
Projekt	AT&T	Sun	IETF	UCM
Serwer czy zasób?	zasób	serwer	serwer	zasób
Semantyka Uniksowa	tak	nie	tak (?)	nie!
Praca jednoczesna	tak!	tak, ostrożna	tak (?)	nie
Migracje	nie	nie	tak (?)	tak

Przyszłość – jakiej chcemy?

Czego oczekujemy od sieciowego systemu plików?

- szybkości działania
- bezpieczeństwa
- możliwości wykorzystania w dużych sieciach (w szczególności w Internecie)

Przyszłość – co dostaniemy?

Jest całkiem prawdopodobne, że systemowi NFSv4 uda się zaspokoić wszystkie nasze potrzeby (w zakresie sieciowych systemów plików oczywiście).

Na zakończenie...

„Stoję na stanowisku i zawsze zgodnie z tym postępuję, że jeśli tego, co się chce powiedzieć, nie można powiedzieć w dwadzieścia minut, to powinno się pójść i napisać o tym książkę.”

— *Derek Charles Moore-Brabazon*