

B4B350SY: Operační systémy

Souborové systémy

Michal Sojka¹



2019-11-28

¹michal.sojka@cvut.cz

- 1 Úvod
- 2 Souborové systémy
 - FAT
 - Souborový systém založený na inode
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Obsah

- 1 Úvod
- 2 Souborové systémy
 - FAT
 - Souborový systém založený na inode
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Co je souborový systém?

- Způsob organizace dat na pevném disku
- Data uložená v pojmenovaných souborech
- Soubory v adresářích (složkách)
- Hierarchická struktura adresářů

Požadavky na souborový systém

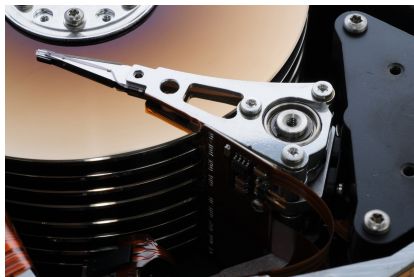
- Efektivita (nízká režie)
- Rychlost
- Nízká fragmentace – souvisí s rychlostí
- Spolehlivost

Terminologie

- Data = obsah souborů
- Metadata = pomocné informace ukládané souborovým systémem (bitmapy, inody, superblok, ...)

Pevný disk

- Trvalé uložení dat (i bez napájení)
- Rotační, Flash
- Posloupnost bloků (sektorů) určité velikosti
- Každý blok je identifikován číslem
- Oddíly (partitions)
 - Fyzický disk lze rozdělit na víc logických disků
 - Na začátku disku je tabulka definující typ, (jméno), počáteční a koncový sektor oddílu
 - Master Boot Record (MBR) – pozůstatek MS-DOSu, 1. sektor na disku (512 B), obsahuje místo pro 4 oddíly.
 - GUID Partition Table (GPT) – modernější, více informací, „neomezený“ počet oddílů.
- Většina souborových systémů využívá jeden logický disk



Otázky

- Jak ukládat adresáře?
- Jak zjistit, ve kterých blocích jsou data daného souboru?
- Jak alokovat bloky na disku při vytváření/zvětšování souborů?
- Jak se vypořádat s chybami a pády systému?
- Jak optimalizovat souborové systémy pro rotační disky a Flash paměti?

Obsah

- 1 Úvod
- 2 Souborové systémy
 - FAT
 - Souborový systém založený na inode
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Adresáře

- Adresář je seznam dvojic («jméno souboru», «umístění»)
- Jméno:
 - V UNIXu všechny znaky kromě / a NUL
 - Ve Windows nesmí obsahovat znaky /\:*"?'<>|
- Umístění: kde jsou uložena data daného souboru – viz dále
- Třídění seznamu (položek v adresáři):
 - Seznam není uložen setříděný; třídění provádí až program zobrazující adresář uživateli podle jím zadaných kritérií
 - Třídění podle názvu, data přístupu, typu souboru
 - Pomalé otevírání souborů ve velkých adresářích
 - Vyhledávací B-strom
 - Rychlejší

Překlad cesty k souboru

- Co se děje při otevírání souboru „/jedna/dva/tři“?
 - 1 Otevře se kořenový adresář „/“ (vždy se ví, kde se najde – superblok)
 - 2 Najde se v něm záznam „jedna“ a zjistí se jeho *umístění*
 - 3 Otevře se adresář „jedna“ a najde se záznam „dva“ a jeho *umístění*
 - 4 Otevře se adresář „dva“, najde se záznam „tři“ a jeho *umístění*
 - 5 Otevře se soubor „tři“
- Procházení cesty a adresářů po cestě může trvat dlouho
 - Proto je volání `open` odděleno od `read / write`
 - Cesta se prochází jen při `open`
 - Položky adresářů se ukládají do vyrovnávací paměti (dentry cache v Linuxu)

Rozložení dat na disku

- Souborový systém definuje **velikost bloku** (např. 4 KiB)
 - Prostor na disku je vždy alokován v násobcích velikosti bloku
- **Superblok** určuje umístění kořenového adresáře a další informace o souborovém systému
 - Vždy na předem známém místě (např. 1. blok na disku)
 - Často uložen ve více kopiích
- Informace o **volných blocích**
 - OS musí mít přehled, který blok je volný a který obsazený
 - Podobné jako v alokátorech paměti – např. freelist
 - Typicky bitová mapa (1 bit na blok)
 - Kopie v paměti pro urychlení přístupu (cache)
- Bloky ukládající **obsah souborů**
 - Existuje mnoho způsobů, jak je organizovat

Základní možnosti uložení obsahu souboru

- Obsah souboru je typicky uložen ve více blocích (do jednoho se nemusí vejít)
- Jak se zjistí, které bloky to jsou?

Základní možnosti uložení obsahu souboru

- Obsah souboru je typicky uložen ve více blocích (do jednoho se nemusí vejít)
- Jak se zjistí, které bloky to jsou?
 - 1 Soubor je vždy uložen v souvislém úseku bloků
 - Podobné alokaci paměti
 - Rychlý přístup k datům (lokalita)
 - Neflexibilní, způsobuje fragmentaci a nutnost přemísťovat soubory

Základní možnosti uložení obsahu souboru

- Obsah souboru je typicky uložen ve více blocích (do jednoho se nemusí vejít)
- Jak se zjistí, které bloky to jsou?
 - 1 Soubor je vždy uložen v souvislém úseku bloků
 - Podobné alokaci paměti
 - Rychlý přístup k datům (lokalita)
 - Neflexibilní, způsobuje fragmentaci a nutnost přemísťovat soubory
 - 2 Spojové seznamy
 - Každý blok obsahuje kromě dat i odkaz na další blok, adresář odkazuje na 1. blok souboru
 - Výhodné pro sekvenční přístup k souborům, nevýhodné pro vše ostatní
 - Nemožnost „mapovat“ data z disku přímo do paměti
 - Jeden špatný sektor na disku (porucha) může způsobit „ztrátu“ zbytku souboru

Základní možnosti uložení obsahu souboru

- Obsah souboru je typicky uložen ve více blocích (do jednoho se nemusí vejít)
- Jak se zjistí, které bloky to jsou?
 - 1 Soubor je vždy uložen v souvislém úseku bloků
 - Podobné alokaci paměti
 - Rychlý přístup k datům (lokalita)
 - Neflexibilní, způsobuje fragmentaci a nutnost přemísťovat soubory
 - 2 Spojové seznamy
 - Každý blok obsahuje kromě dat i odkaz na další blok, adresář odkazuje na 1. blok souboru
 - Výhodné pro sekvenční přístup k souborům, nevýhodné pro vše ostatní
 - Nemožnost „mapovat“ data z disku přímo do paměti
 - Jeden špatný sektor na disku (porucha) může způsobit „ztrátu“ zbytku souboru
 - 3 Indexové struktury
 - „Indexový blok“ obsahuje ukazatele (čísla bloků) na mnoho jiných bloků
 - Vhodnější pro náhodný přístup, stále poměrně dobré pro sekvenční přístup
 - Může být potřeba použít více indexových bloků

Obsah

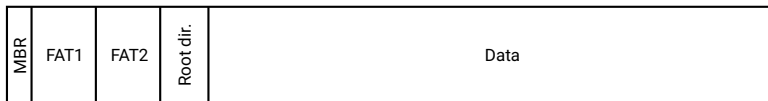
- 1 Úvod
- 2 Souborové systémy
 - FAT
 - Souborový systém založený na inode
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Souborový systém FAT

File Allocation Table

Starý souborový systém s mnoha nedostatky a omezeními.

- Základní jednotka „cluster“ (4–32 KiB)
- FAT12: 2^{12} clusterů, FAT16: 2^{16} , FAT32: 2^{28}
- Rozložení disku



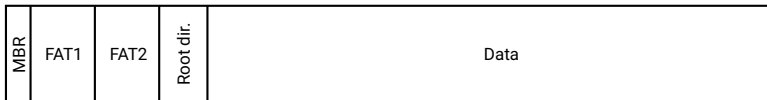
- MBR – master boot record (info o soub. systému, tj. velikost, jméno, počet kopií FAT tabulek atd.)
- FAT1, 2 – dvě kopie FAT tabulky (redundance)

Souborový systém FAT

File Allocation Table

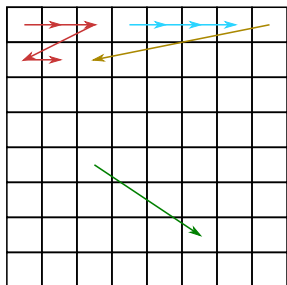
Starý souborový systém s mnoha nedostatky a omezeními.

- Základní jednotka „cluster“ (4–32 KiB)
- FAT12: 2^{12} clusterů, FAT16: 2^{16} , FAT32: 2^{28}
- Rozložení disku



- MBR – master boot record (info o soub. systému, tj. velikost, jméno, počet kopií FAT tabulek atd.)
- FAT1, 2 – dvě kopie FAT tabulky (redundance)

Tabulka FAT



- Jedna položka FAT tabulky má 12/16/32 bitů a odpovídá jednomu clusteru na disku
- Hodnota položky udává číslo následujícího clusteru (konec šipky) nebo -1 značící konec souboru.
- Číslo 1. clusteru se najde v položce adresáře
- Pro urychlení přístupu je tabulka uchovávána v paměti

Nevýhody

- Fragmentace
- Omezená velikost (např. $2^{32} \times 4\text{KiB}$)
- Nutnost procházet FAT položky sekvenčně (zpomaluje náhodný přístup u velkých souborů)

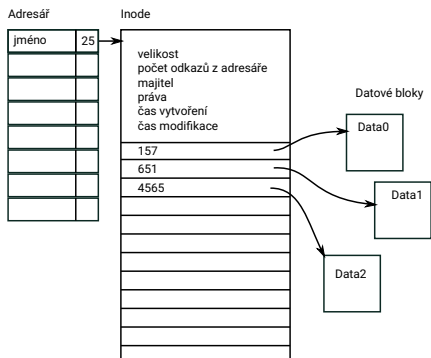
Obsah

- 1 Úvod
- 2 Souborové systémy**
 - FAT
 - Souborový systém založený na inode**
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Indexový souborový systém

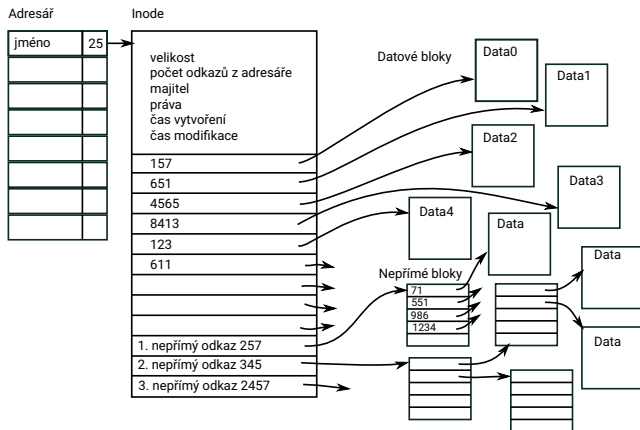
Základ mnoha UNIXových souborových systémů.

- Metadata o jednotlivých souborech jsou uložena v datové struktuře zvané **inode**.
- Položka adresáře obsahuje kromě jména souboru i číslo (pořadí) inode
- inode obsahuje pevný počet odkazů na datové bloky
- Několik inode se vejde do 1 bloku (velikost inode bývá např. 128 B)



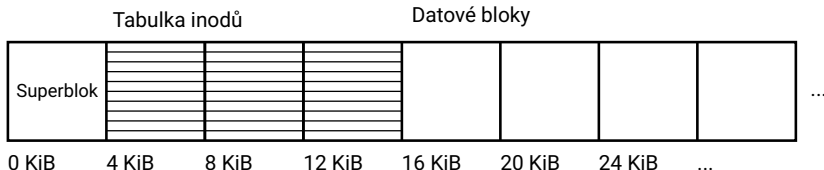
Nepřímé bloky

- Co když je soubor větší, než počet odkazů na datové bloky v inode?
- Odkaz na další bloky nepřímo, přes blok odkazů
- Nepřímé bloky mohou být i v dalších úrovních



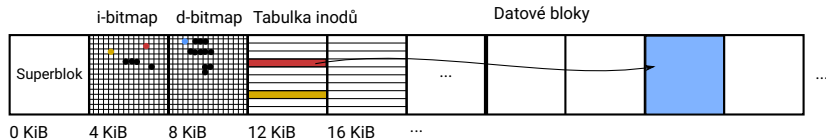
Rozložení na disku

- Pevný počet inodů
- inode lze nalézt na základě jeho indexu v tabulce
- inode je zkratka *index node*
- Superblok – informace o souborovém systému
 - celková délka, počet inodů, ...
 - počet volných bloků a inode
 - odkaz na záložní kopii superbloku
- Kořenový adresář: např. v inode č. 0



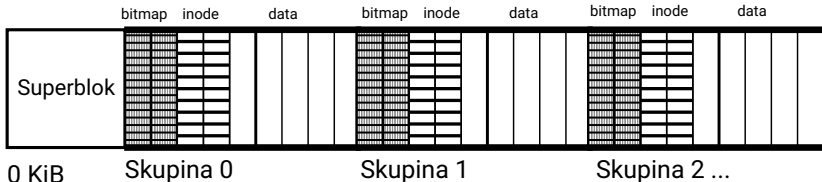
Hledání volného místa

- Jak poznat, který inode je volný (např. při vytváření nového souboru)?
 - Sekvenčním procházením všech inode
- Jak poznat, který datový blok je volný?
 - Těžko
- Bitové mapy pro inode a datové bloky
 - každý bit udává obsazenost inodu nebo datového bloku



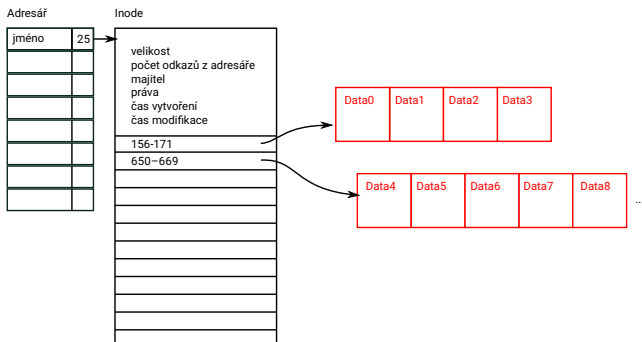
Skupiny (ext2-4)

- Při práci se souborem je potřeba pracovat s bitmapou, inodem a datovými bloky
- Disky (zejména rotační, ale částečně i SSD) přistupují rychleji k blokům blízko sebe
- Co když datové bloky budou až na konci disku?
 - Hlavičky disků musí pořád jezdit mezi začátkem (bitmapy, tab. inode) a koncem disku (data)
- Řešení: skupiny
 - Souborový systém se snaží alokovat datové bloky ve stejné skupině jako inode souboru

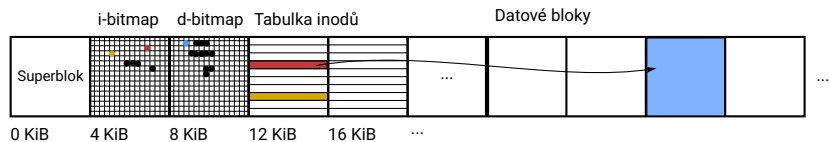


Extents

- Tabulky bloků nejsou efektivní pro velké soubory, velká režie
- Moderní souborové systémy mohou odkazovat místo na jednotlivé bloky na celé souvislé skupiny
- Odkazovaná skupina s více než jedním blokem se nazývá **extent**
- Implementováno v: ext4, NTFS, btrfs, ...

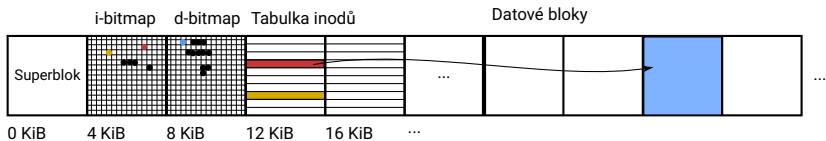


Konzistence dat



- Při zápisu do souboru je potřeba měnit: bitmapy, inode/nepřímé bloky a data
- Hardware disku garantuje atomický zápis pouze jednoho sektoru
- V jakém pořadí bloky zapisovat na disk?
- Co se stane, když dojde k pádu či vypnutí systému v průběhu zapisování při následujících pořadích zápisu?
 - bitmapa, inode/nepřímé bloky, data
 - inode, data, bitmapa
 - bitmapa, data, inode

Konzistence dat



- Při zápisu do souboru je potřeba měnit: bitmapy, inode/nepřímé bloky a data
- Hardware disku garantuje atomický zápis pouze jednoho sektoru
- V jakém pořadí bloky zapisovat na disk?
- Co se stane, když dojde k pádu či vypnutí systému v průběhu zapisování při následujících pořadích zápisu?
 - bitmapa, inode/nepřímé bloky, data
 - inode, data, bitmapa
 - bitmapa, data, inode
- Vždy může vzniknout v datech nějaká nekonzistence!

Možná řešení problémů s integritou souborového systému

- 1 Kontrola souborového systému při startu počítače
 - projdu všechny inode a nepřímé bloky
 - zjistím, jestli bitmapa volných inode souhlasí se stavem tabulky inode
 - zjistím, jestli bitmapa datových bloků souhlasí s informacemi v inode
 - zjistím, jestli dva inode neodkazují na stejné bloky
 - ...
 - **Pomalé**, zejména na velkých discích!
- 2 Žurnálování
- 3 Copy-on-write

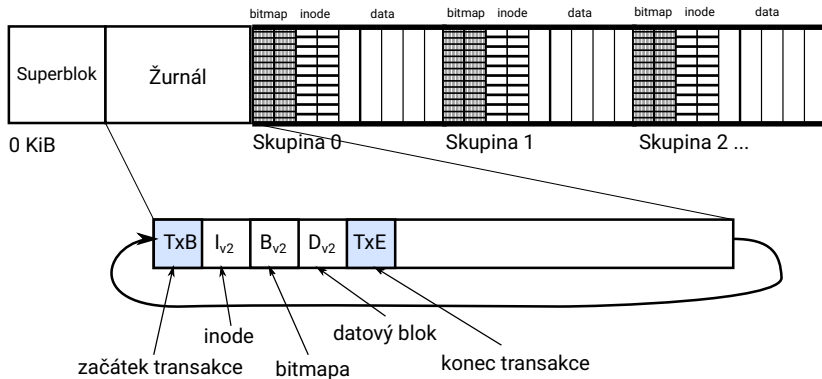
Obsah

- 1 Úvod
- 2 Souborové systémy
 - FAT
 - Souborový systém založený na inode
- 3 **Žurnálování**
- 4 Souborové systémy pro Flash paměti

Žurnálovací systém souborů

- Před tím, než se začne souborový systém modifikovat, se uloží seznam potřebných modifikací na vyhrazené místo – **žurnál**
- Pokud dojde k pádu systému, zkontroluje se žurnál, změny disku v něm nalezené se provedou dodatečně
- Žurnálování se někdy nazývá „dopředné logování“
- Implementováno: NTFS, ext3, ...

Struktura žurnálovacího systému souborů (ext3)



Bezpečný způsob změny souborového systému

1 Commit – zapsání transakce do žurnálu

- TxB : obsahuje id transakce a čísla bloků měněného inode, bitmap a dat
- I_{v2} : nová verze bloku s inode
- B_{v2} : nová verze bloku bitmapy
- D_{v2} : nový datový blok
- TxE : id transakce, kontrolní součet

2 Checkpoint – provedení změn

- 1 aktualizace bloků v souborovém systému (inode, bitmapy, data)
- 2 odstranění transakce z žurnálu

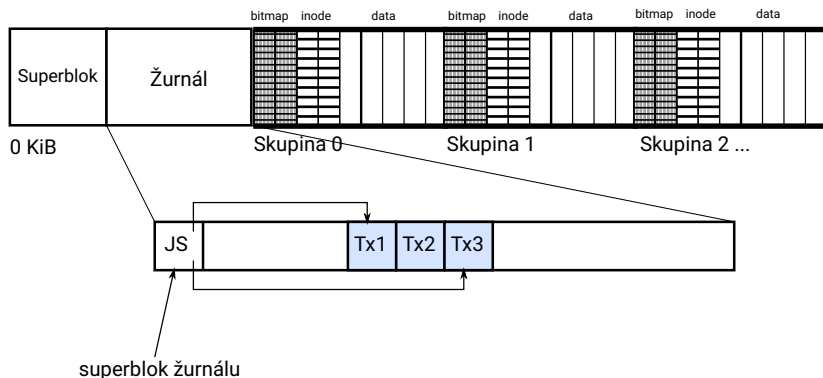
Možné scénáře pádu systému

- 1 Zapiše se pouze část transakce
 - Souborový systém (SS) je konzistentní a obsahuje původní data
 - Při startu OS se zjistí, že transakce v žurnálu není kompletní (viz další slide) a ignoruje se.
- 2 Zapišeme celou transakci, ale neaktualizují se bloky SS
 - Při startu OS aktualizujeme bloky podle informací v žurnálu
- 3 Zapišeme celou transakci, aktualizujeme bloky systému, ale neodstraníme transakci ze žurnálu
 - Při startu OS se bloky přepíše ze žurnálu – žádná změna, už zapsané byly

Možné scénáře pádu systému

- 1 Zapiše se pouze část transakce
 - Souborový systém (SS) je konzistentní a obsahuje původní data
 - Při startu OS se zjistí, že transakce v žurnálu není kompletní (viz další slide) a ignoruje se.
- 2 Zapišeme celou transakci, ale neaktualizují se bloky SS
 - Při startu OS aktualizujeme bloky podle informací v žurnálu
- 3 Zapišeme celou transakci, aktualizujeme bloky systému, ale neodstraníme transakci ze žurnálu
 - Při startu OS se bloky přepíše ze žurnálu – žádná změna, už zapsané byly
- 4 Zapiše se pouze část transakce (např. TxB , I_{V2} a TxE)
 - **Problém!**
 - HW disků se snaží provádět optimalizace a může změnit pořadí vykonávání příkazů zaslaných OS
 - OS musí disku posílat speciální příkazy (tzv. bariéry), aby se data skutečně zapsala v potřebném pořadí
 - Bariéra garantuje, že příkazy zaslané před bariérou budou vykonány před příkazy zaslanými po bariéře

Nevyřízené transakce



- V jednom okamžiku může vypadat žurnál např. takto
- Commit transakce do žurnálu nebo její smazání se provede atomickým zápisem superbloku žurnálu

Rychlost žurnálu

- Pomalé
 - Commit: zápis metadat a dat do žurnálu
 - Checkpoint: aktualizace inode, bitmapy a dat podle transakce
 - Vše se zapisuje na disk dvakrát!

Rychlost žurnálu

■ Pomalé

- Commit: zápis metadat a dat do žurnálu
- Checkpoint: aktualizace inode, bitmapy a dat podle transakce
- Vše se zapisuje na disk dvakrát!

■ Rychlejší:

- Zapsání dat přímo do daného bloku + bariéra
- Commit metadat: Když jsou data uložena, zapsání transakce pro změnu metadat do žurnálu
- Checkpoint: Aktualizace inode a bitmap podle transakce
- Jaké chyby v SS mohou nastat při pádu systému?

Rychlost žurnálu

■ Pomalé

- Commit: zápis metadat a dat do žurnálu
- Checkpoint: aktualizace inode, bitmapy a dat podle transakce
- Vše se zapisuje na disk dvakrát!

■ Rychlejší:

- Zapsání dat přímo do daného bloku + bariéra
- Commit metadat: Když jsou data uložena, zapsání transakce pro změnu metadat do žurnálu
- Checkpoint: Aktualizace inode a bitmap podle transakce
- Jaké chyby v SS mohou nastat při pádu systému?

■ Ještě rychlejší:

- Zapsání dat přímo do daného bloku
- Commit metadat: zapsání transakce pro změnu metadat do žurnálu
- Checkpoint: Aktualizace inode a bitmap podle transakce (doufáme, že data zapsána také)
- Jaké chyby v SS mohou nastat při pádu systému?

Souborový systém ext4/jbd2

- Uživatel si může zvolit, jaký mód žurnálování se použije
 - **journal**: všechna data i metadata se zapisují skrze žurnál
 - **ordered** (výchozí nastavení): data se zapisují přímo, metadata skrze žurnál po zapsání dat
 - **write-back**: data se zapisují přímo, ale jejich zápis nemusí proběhnout před zápisem metadat (skrze žurnál)
- Typická velikost žurnálu: 128 MiB

Obsah

- 1 Úvod
- 2 Souborové systémy
 - FAT
 - Souborový systém založený na inode
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Vlastnosti Flash paměti

- Zapisovat lze pouze do vymazaného bloku
- Zapsat na jedno místo lze pouze jednou
- Mazací blok bývá mnohem větší (např. 4 MiB) než blok souborového systému (4 KiB)
- Každý blok garantuje pouze určitý počet přepsání – např. 100 tisíc

Důsledky pro „tradiční“ souborový systém?

- Často se měnící data (např. bitmapy, či FAT tabulka) drasticky snižují životnost paměti
- Změna jednoho bytu v souboru znamená smazání a znovu zapsání 4 MiB
- Garance poskytované žurnálovacím souborovým systémem neplatí pro Flash
 - Commit žurnálu musí vymazat 4 MiB okolo
 - Pokud systém havaruje mezi smazáním a zápisem, přijdeme o data v žurnálu

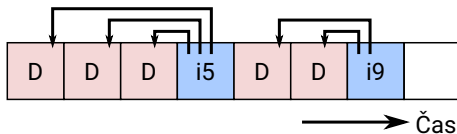
Řešení

- 1** Nepoužívat Flash čipy samostatně, ale v kombinaci s řadičem, implementující „Flash Translation Layer“ (FTL)
 - Mapuje logická čísla sektorů zaslaných OS na bloky flash paměti tak, aby nedocházelo k nežádoucím jevům (např. neustálé přepisování stejného bloku)
 - Implementováno v SSD discích, SD kartách, USB pamětech apod.
 - SD karty/USB paměti mají FTL často optimalizovaný pro souborový systém FAT.
 - Pokud se použije jiný souborový systém, je to pomalé a paměť dlouho nevydrží
- 2** Použít speciální souborové systémy pro Flash paměti
 - UBIFS, JFFS2, NILFS, ...

Protokolovací souborové systémy

Log-Structured file systems

- Data se čtou převážně z vyrovnávací paměti (page cache)
- Stačí se zaměřit na operace zápisu – snaha je zapisovat data rovnoměrně po celé oblasti disku
- Zápis velkých souvislých bloků dat je velmi efektivní (není třeba znovu zapisovat nezměněná data v mazacím bloku)
- Stav celého souborového systému je dán zaznamenaným protokolem událostí

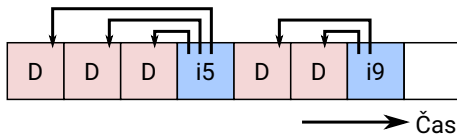


- Jak najdeme inody?

Protokolovací souborové systémy

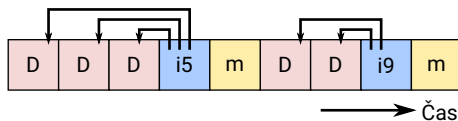
Log-Structured file systems

- Data se čtou převážně z vyrovnávací paměti (page cache)
- Stačí se zaměřit na operace zápisu – snaha je zapisovat data rovnoměrně po celé oblasti disku
- Zápis velkých souvislých bloků dat je velmi efektivní (není třeba znovu zapisovat nezměněná data v mazacím bloku)
- Stav celého souborového systému je dán zaznamenaným protokolem událostí



- Jak najdeme inody?
 - Sekvenčně projdeme celý disk a najdeme je. Pomalé :-)

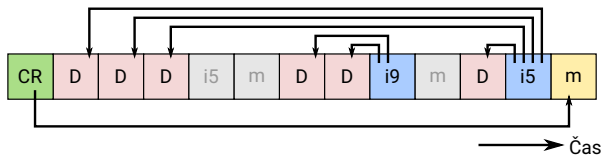
Mapa inodů



- Mapa inodů obsahuje tabulku pro převod čísel inodů na čísla bloků
- Jak zjistíme, která verze mapy je poslední?

Kontrolní region

Check region (CR)



Čtení souboru

- Přečti kontrolní region
- Najdi pozici mapy inodů (m)
- Najdi inode
- Přečti datové bloky

Zápis souboru

- Zapiš datové bloky
- Zapiš změněnou kopii inode
- Zapiš změněnou kopii mapy inodů
- Aktualizuj kontrolní region

■ CR se pořád přepisuje – nevdí to?

- Nevdí, pokud máme např. SD kartu optimalizovanou pro FAT (viz slide 34)
- Využíváno např. F2FS (Samsung)