B4B35OSY: Operační systémy

Souborové systémy

Michal Sojka[1]

7. prosince 2017

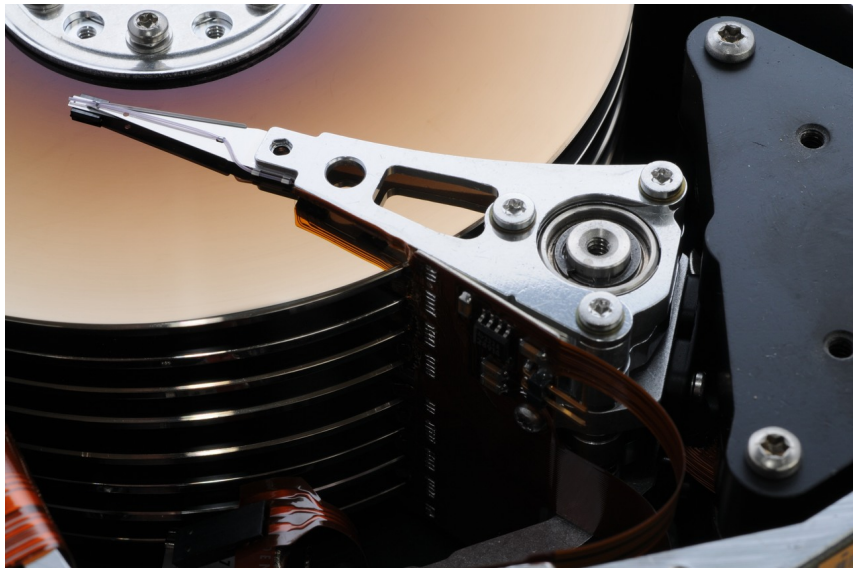
[1]michal.sojka@cvut.cz

Obsah I

Obsah

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
  - MBR, GPT
  - Většina souborových systému využívá jeden logický disk

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

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

Adresáře

- Adresář je seznam dvojic (*«jméno souboru»*, *«umístění»*)
- Jméno:
  - V UNIXu všechny znaky kromě „/“, NUL
  - Ve Windows nesmí obsahovat /\:*"?<>|
- Umístění: viz dále
- Třídění seznamu:
  - 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“?
  - Otevře se kořenový adresář „/“ (vždy se ví, kde se najde)
  - Najde se v něm záznam „jedna“ a zjistí se jeho *umístění*
  - Otevře se adresář „jedna“ a najde se záznam „dva“ a jeho *umístění*
  - Otevře se adresář „dva“, najde se záznam „tři“ a jeho *umístění*
  - Otevře se soubor „tři“
- Procházení cesty a adresářů po cestě může trvat dlouho
  - Proto je volání `open` odděleno od `read` / `write`
  - 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 použitý
  - 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

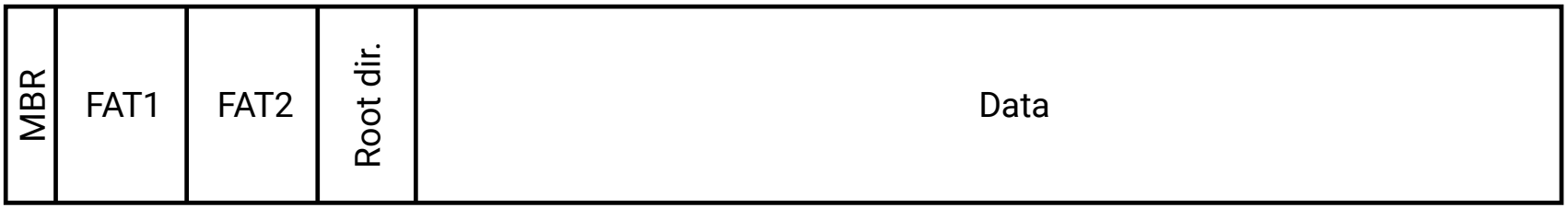
Základní možnosti uložení obsahu souboru

- Obsah souboru je typicky uložen ve více než jednom bloku
- Jak se zjistí, které bloky to jsou?
  1. Alokace souvislého úseku bloků
     - Podobné alokaci paměti
     - Rychlý přístup k datům (lokalita)
     - Neflexibilní, způsobuje fragmentaci a nutnost přemisťovat soubory
  2. Spojové seznamy
     - Každý blok ukazuje na další, adresář ukazuje 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 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
     - Může být potřeba použít více indexových bloků

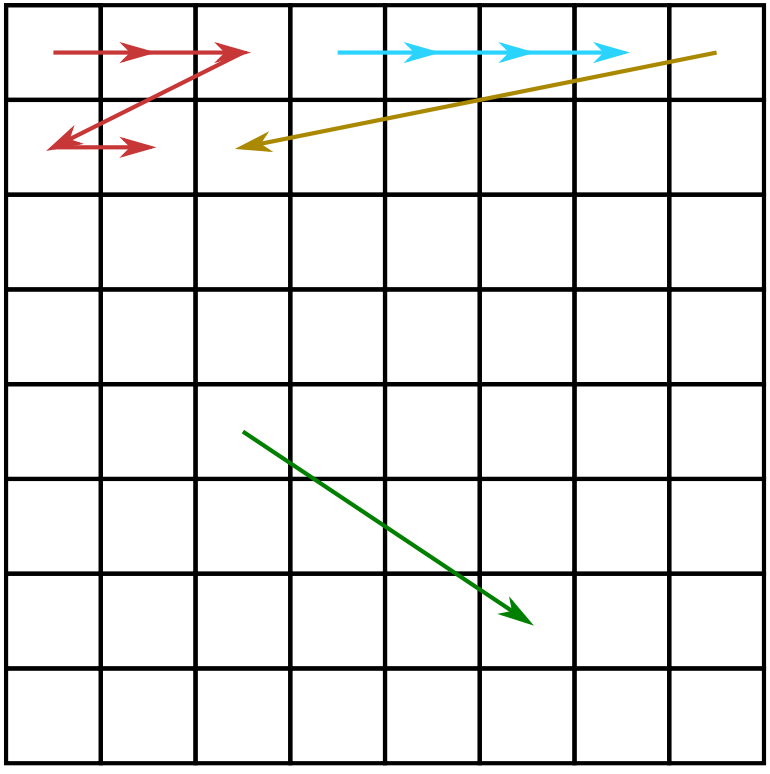
Souborový systém FAT

File Allocation Table

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



Tabulka FAT



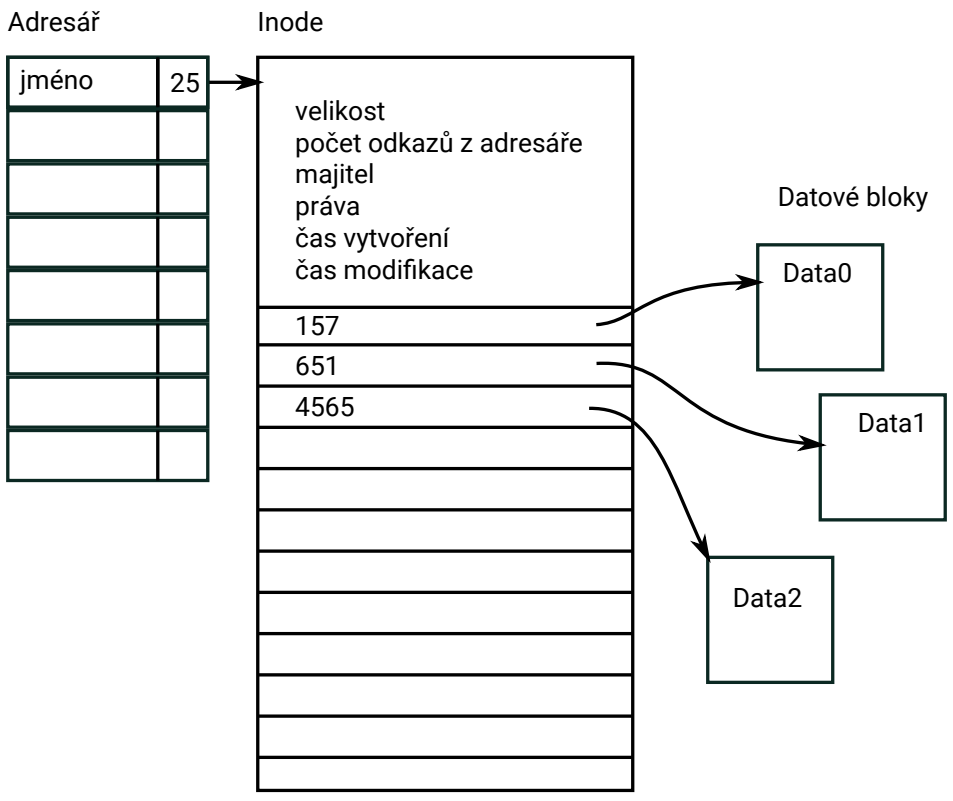
- Jedna položka FAT tabulky má 12/16/32 bitů a odpovídá clusteru na disku
- Hodnota v tabulce udává číslo následujícího clusteru (konec šipky) nebo 0 značící konec souboru.
- Číslo 1. clusteru se najde v položce adresáře
- Pro urychlení přístupu je tabulka uchovávána v paměti

Nevýhody

- Fragmentace
- Omezená velikost
- Nutnost procházet bloky sekvenčně

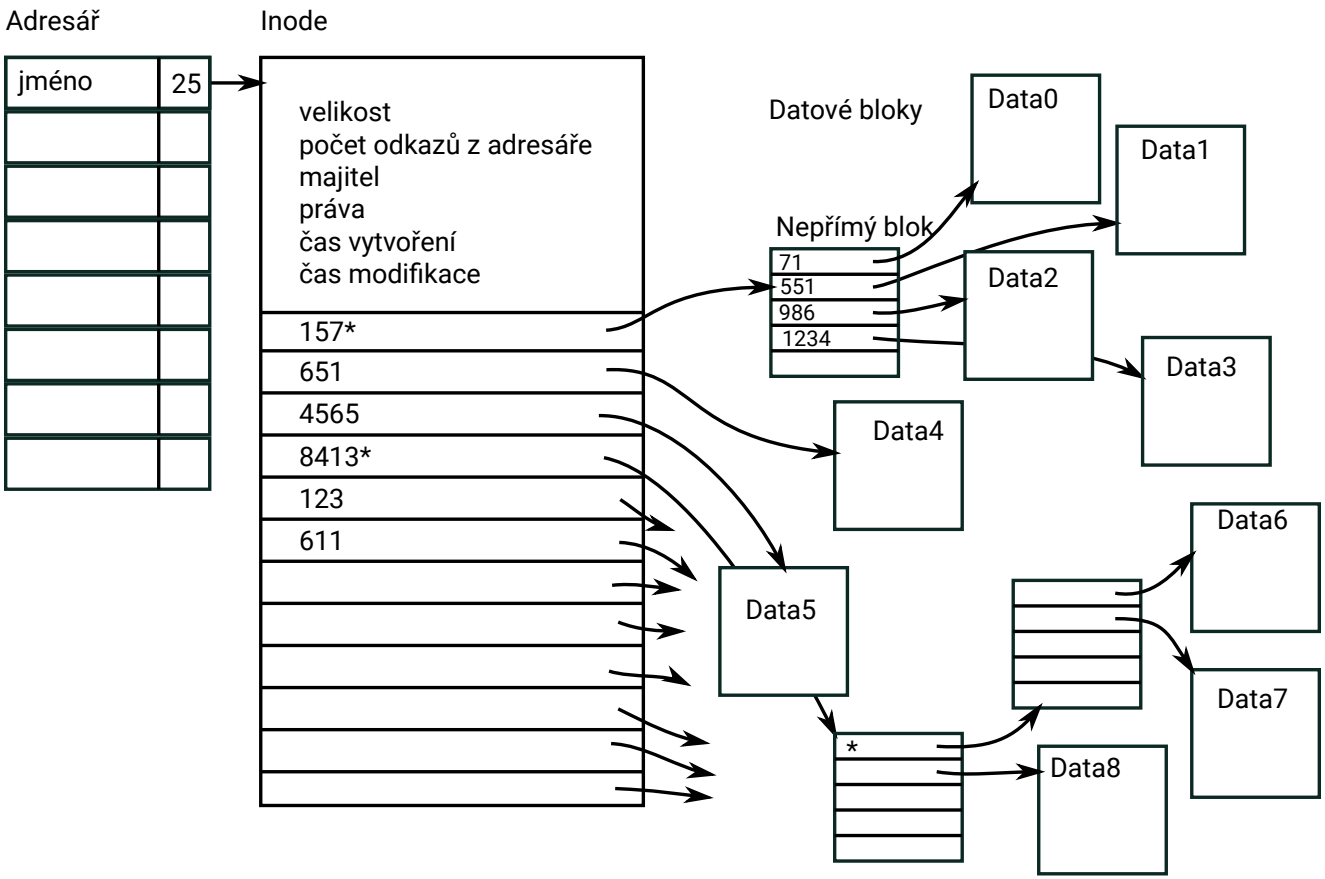
Indexový souborový systém

- Metadata o jednotlivých souborech jsou uložena v datové struktuře zvané inode.
- inode obsahuje pevný počet odkazů na datové bloky
- Několik inode se vejde do 1 bloku (velikost inode bývá např. 128 B)



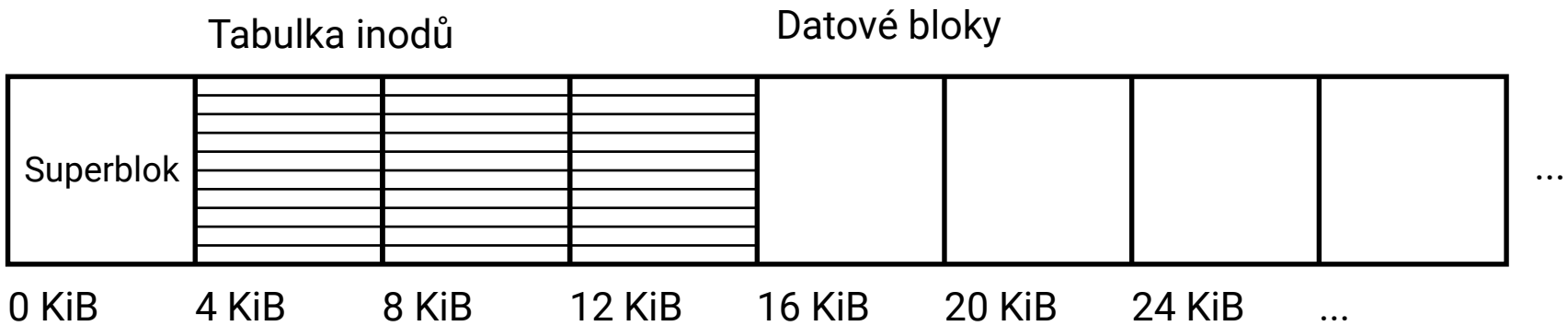
Nepřímé bloky

- 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



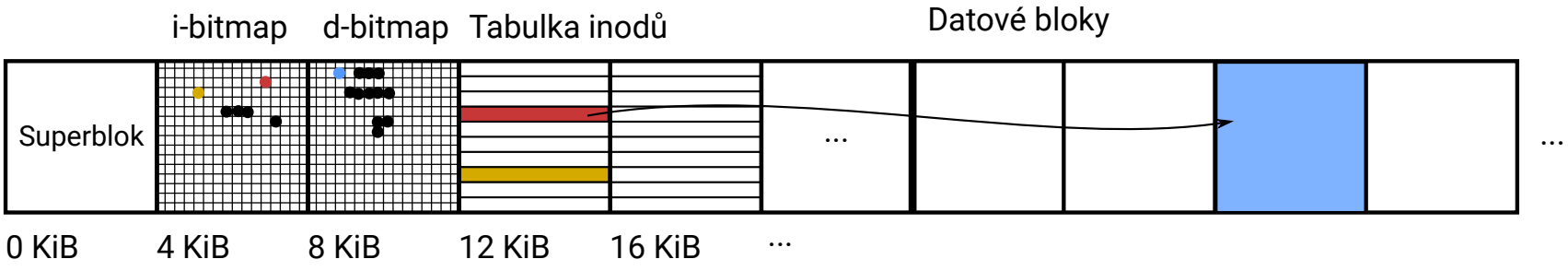
Rozložení na disku

- 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 inode, …
  - počet volných bloků a inode
  - odkaz na záložní kopii superbloku
- Kořenový adresář: např. v inode č. 0



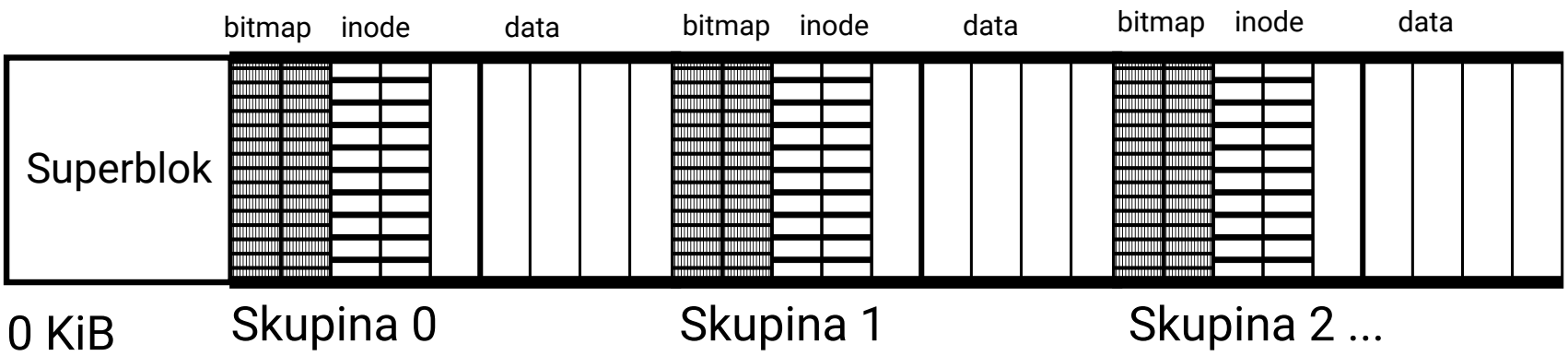
Hledání volného místa

- 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



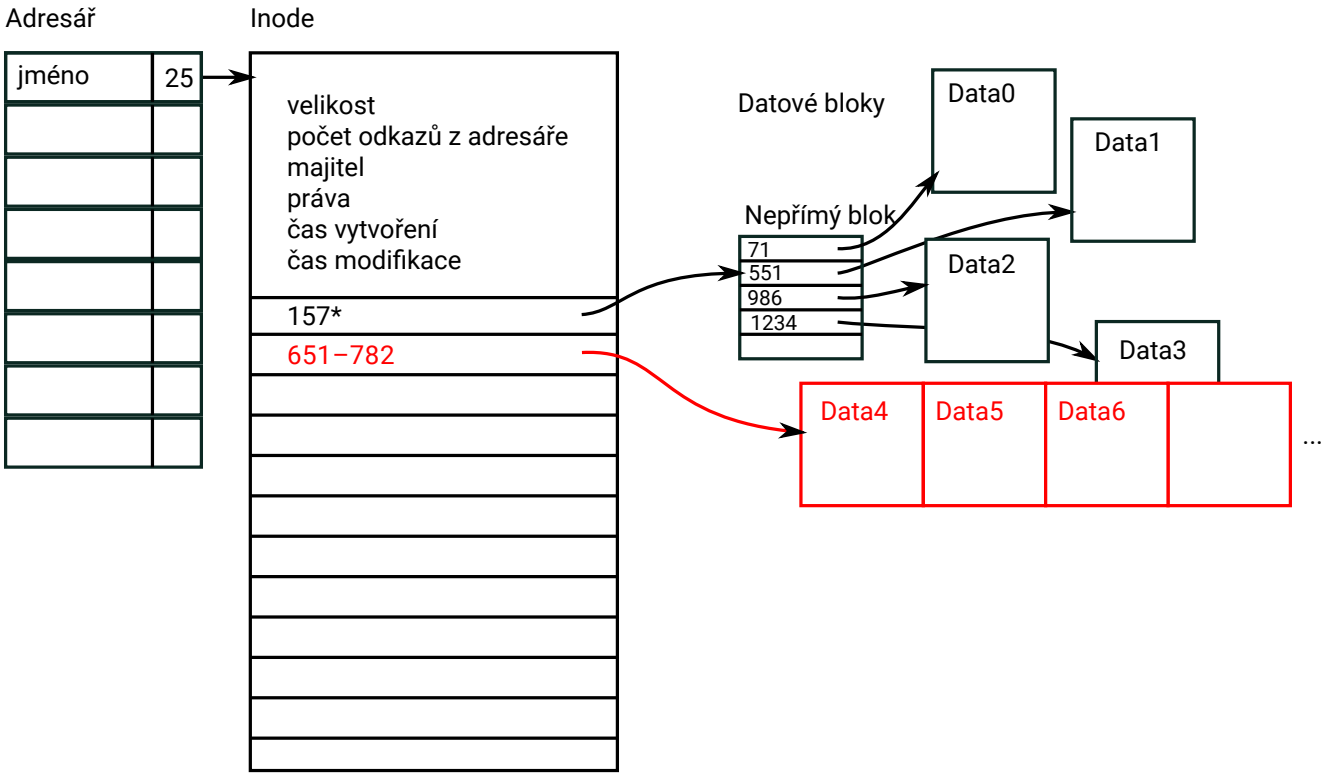
Skupiny (ext2–4)

- 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 a koncem disku
- Řešení: skupiny

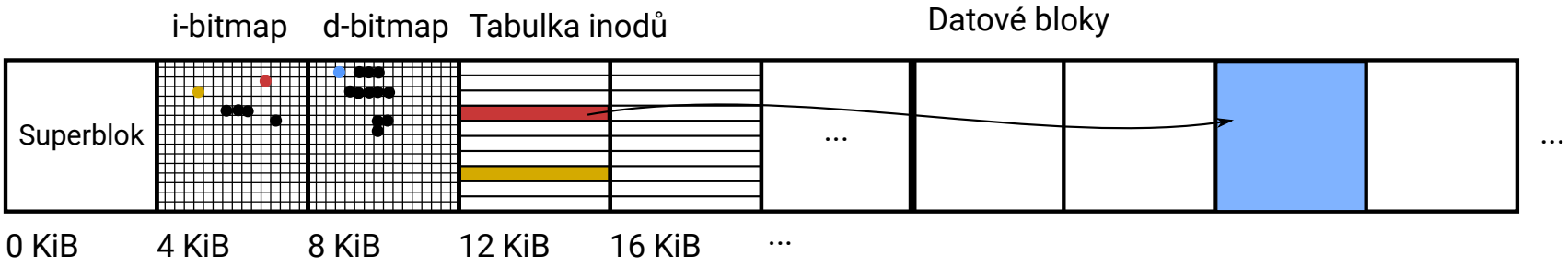


Extents

- Tabulky bloků nejsou efektivní pro velké soubory, velká režie
- Moderní souborové systému 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?
  - bitmapa, inode/nepřímé bloky, data
  - inode, data, bitmapa
  - bitmapa, data, inode

Řešení problémů s integritou souborového systému

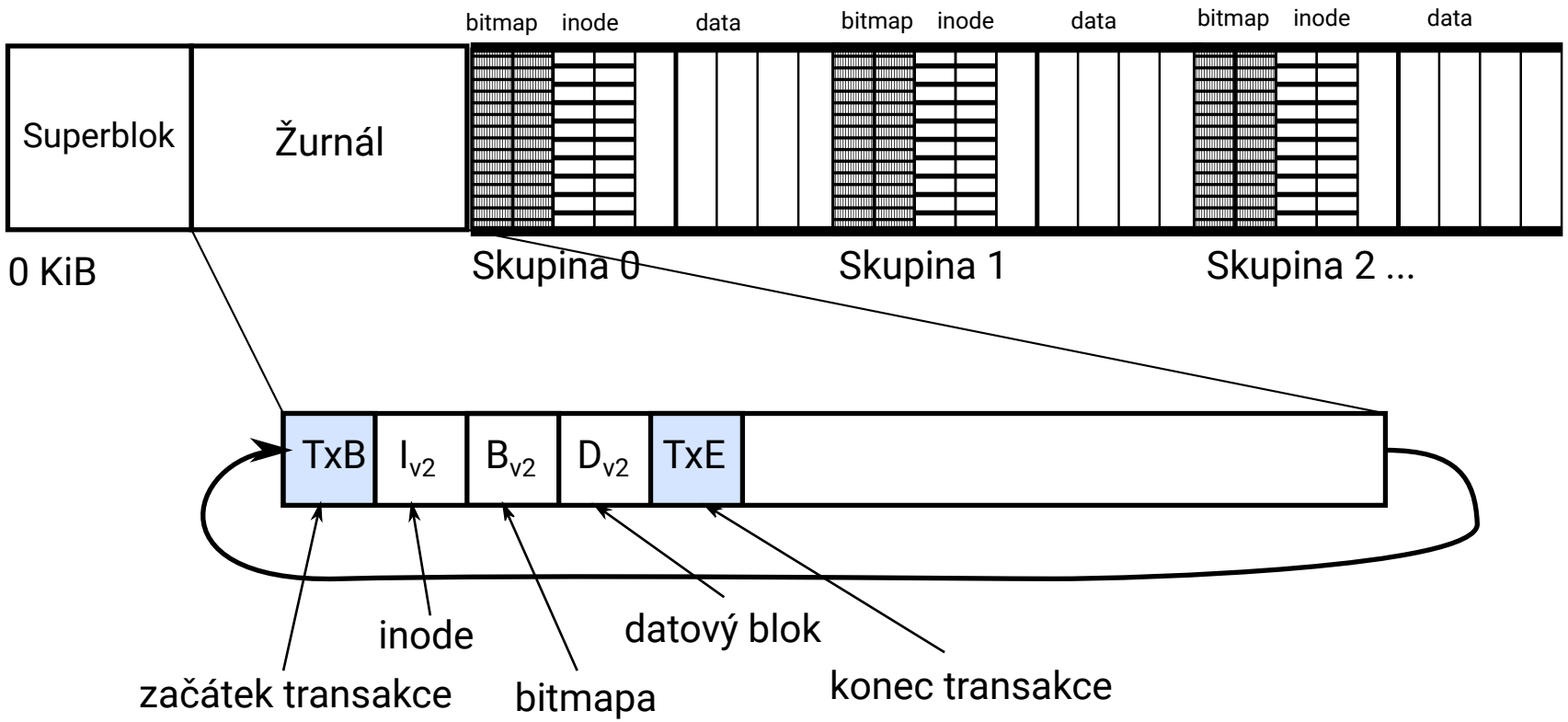
- Kontrola souborového systému při startu počítače
  - projdu všechny inode a nepřímé bloky
  - zjistím, jestli bitmapa volných inode souhlasí se stavem tabulky inode
  - zjistím, jestli bitmapa datových bloků souhlasí s informacemi v inode
  - zjistím, jestli dva inode neodkazují na stejné bloky
  -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

Obsah

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

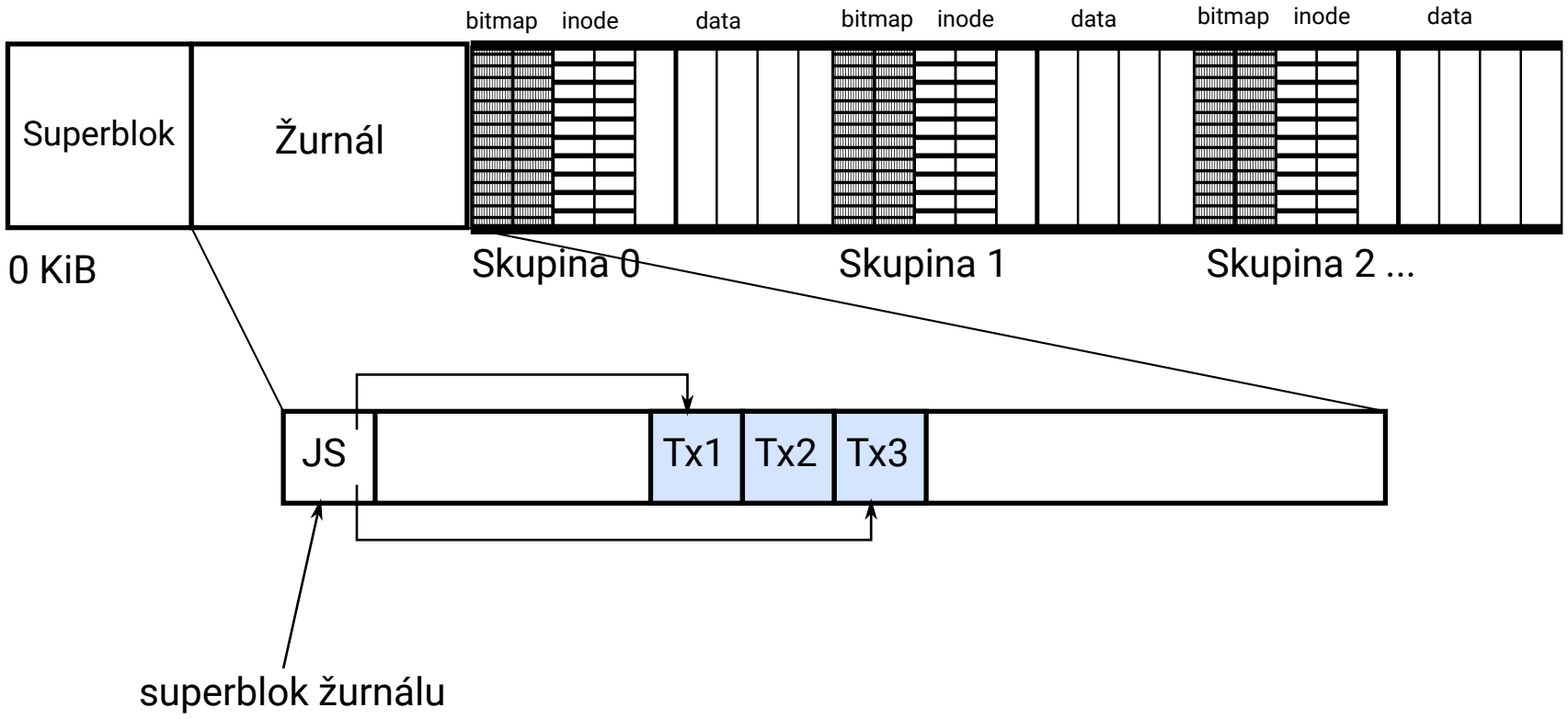
Možné scénáře pádu systému

- Zapíše se pouze část transakce
  - Souborový systém (SS) je konzistentní a obsahuje původní data
- Zapíšeme celou transakci, ale neaktualizují se bloky SS
  - Při startu OS aktualizujeme bloky podle informací v žurnálu
- Zapíšeme celou transakci, aktualizujeme bloky systému, ale neodstraníme transakci ze žurnálu
  - Při startu OS se bloky přepíší ze žurnálu – žádná změna, už zapsané byly

Možné scénáře pádu systému

- Zapíše se pouze část transakce
  - Souborový systém (SS) je konzistentní a obsahuje původní data
- Zapíšeme celou transakci, ale neaktualizují se bloky SS
  - Při startu OS aktualizujeme bloky podle informací v žurnálu
- Zapíšeme celou transakci, aktualizujeme bloky systému, ale neodstraníme transakci ze žurnálu
  - Při startu OS se bloky přepíší ze žurnálu – žádná změna, už zapsané byly
- Zapíše se pouze *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 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 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 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
  - *odered* (výchozí nastavení): data se zapisují přímo, metadat skrze žurnál po zapsání dat
  - *write-back*: zapsání dat nemusí proběhnout před zápisem metadat
- Typická velikost žurnálu: 128 MiB

Obsah

Vlastnosti Flash pamětí

- Zapisovat lze pouze do vymazaného bloku
- Zapsat na jedno místo lze pouze jedou
- 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

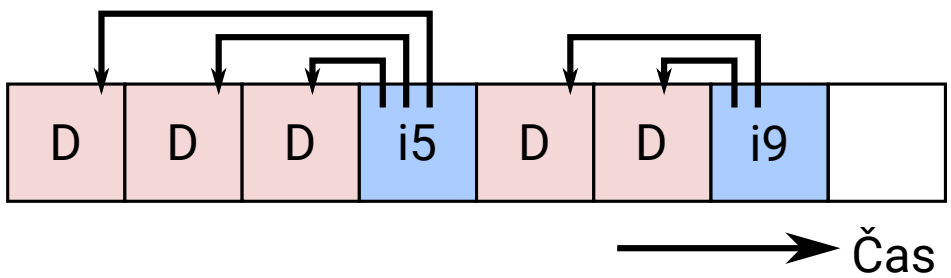
Řešení

1. Nepoužívat Flash čipy samostatně, ale v kombinaci s řadičem, implementující „Flash Translation Layer“ (FTL)
   - Mapuje logická čísla sektorů zaslaných OS na bloky flash pamětí tak, aby nedocházelo k nežádoucím jevům
   - Implementováno v SSD dicích, SD kartách, USB pamětech apod.
   - SD karty/USB pamětí 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

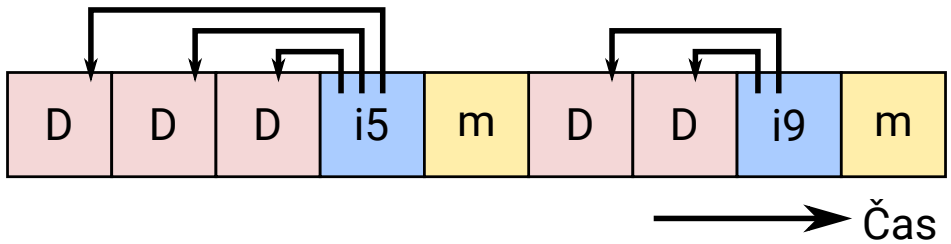
Log-Structured file systems

- 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?

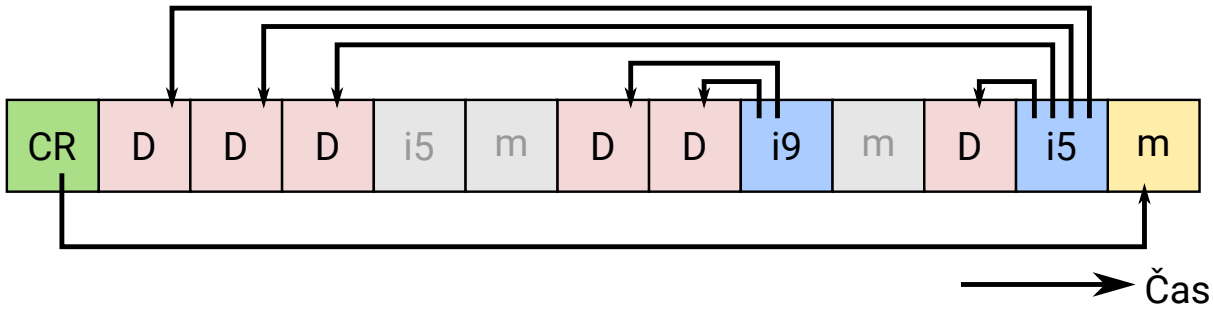
Mapa inodů



- 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 – nevadí to?
- Využíváno např. F2FS (Samsung)