Souborové systémy v Linuxu

Red Hat
Lukáš Czerner
May 16, 2016

Agenda

1. Co je to souborový systém
2. Základní pojmy
3. Rozhraní souborových systémů
4. Interní struktury
5. Konzistence při výpadku
6. Pokročilé funkce
7. Nové typy zařízení
8. Jak se zapojit
9. Otázky

Part I

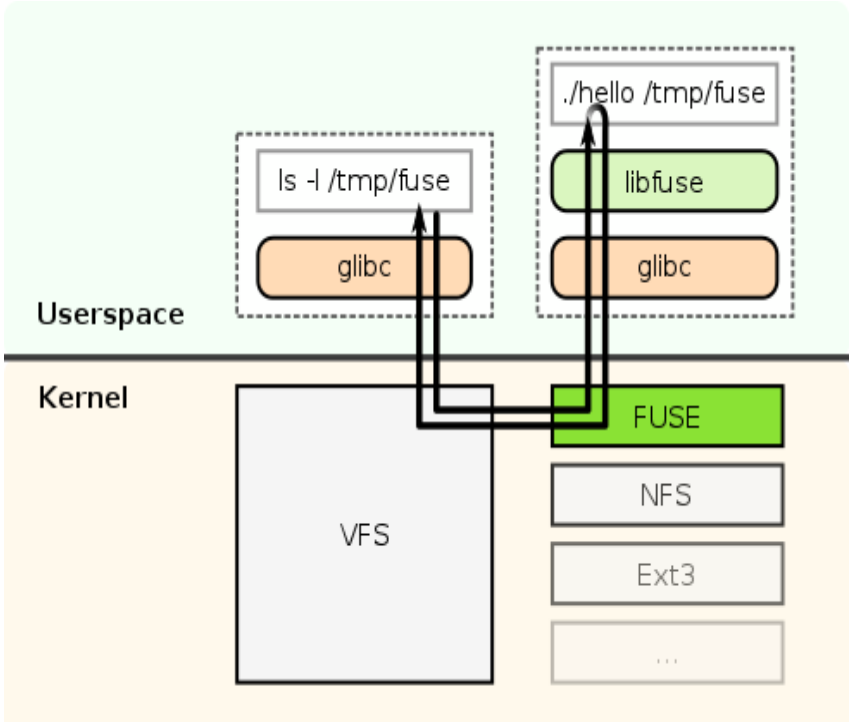
Co je to souborový systém ?

Co je to souborový systém ?

- Způsob organizace dat na nosném médiu ve formě souborů a adresářů
- **Snadný přístup**
  - Uživatelská data snadno přístupná pojmenovaných souborech
  - Soubory seskupené v pojmenovaných adresářích
  - Snadno pochopitelná stromová struktura
- **Virtualizace adresového prostoru média**
  - Adresový prostor souboru vs. logický prostor média
  - Prostory jednotlivých souborů jsou na sobě nezávislé
- **Řízení přístupu**
  - Práva ke čtení, zápisu
  - Kvóty pro omezení množství dat

Typy souborových systémů

- **Souborové sysémy v uživatelském prostoru**
  - použitím jaderného modulu FUSE
  - **GlusterFS**, **sshfs**
- **Souborové systémy v jaderném prostoru**
  - Distribuované, síťové, lokální
  - Speciální souborové systémy
  - Pseudo souborové systémy



Utility v uživatelském prostoru

- **Nástroje pro vytvoření souborového systému**
  - mkfs.ext4, mkfs.xfs, ...
  - Vytvoření souborového systému s danými parametry
- **Nástroje pro kontrolu souborového systému**
  - fsck.ext4, xfs_repair, ...
  - Kontrola, oprava, optimalizace
- **Nástroje pro správu souborového systému**
  - btrfs, resize2fs, tune2fs, xfs_growfs, debugfs
  - Vše od změny velikosti, přes export metadat až k detailní úpravě interních struktur

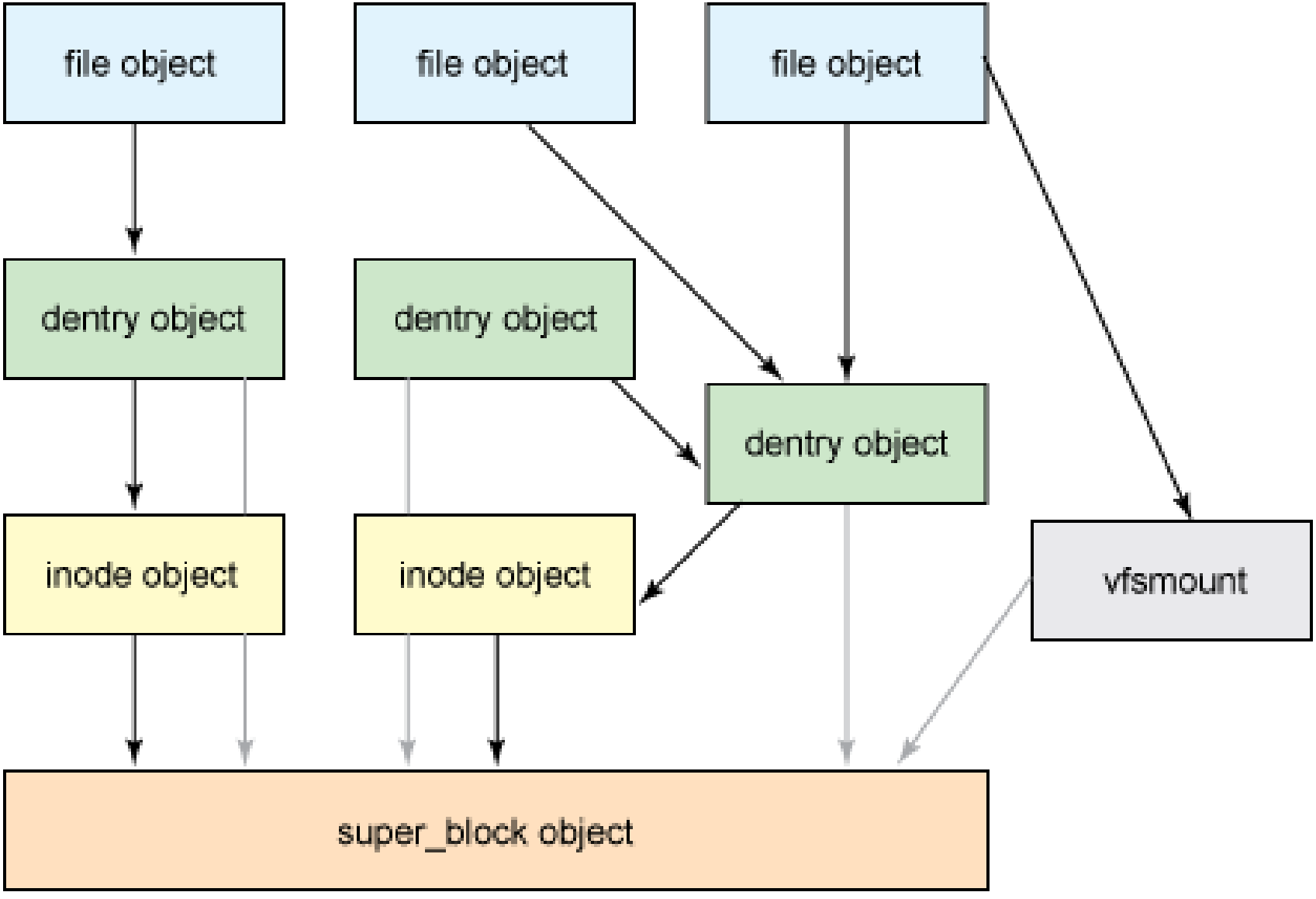
Part II
Pojmy

Důležité struktury

- **Inode** - Index node
  - Struktura reprezentující všechny typy souborů - v paměti
  - **i_mode** - typ souboru
  - **i_ino** - číslo inode
  - **i_nlink** - počet odkazů na inode
  - **i_size** - velikost inode
  - *další viz. include/linux/fs.h:528*
- **Dentry** - Directory entry
  - Struktura mapující jméno souboru na číslo inode - v paměti
  - **d_parent** - ukazatel rodičovskou dentry
  - **d_name** - struktura obsahující jméno záznamu
  - **d_inode** - odkaz na inode - může být NULL
  - *další viz. include/linux/dcache.h:108*

Důležité struktury - pokračování

- **File**
  - Reprezentuje otevřený soubor
  - **f_path** - struktura reprezentuje cestu k souboru
  - **f_inode** - odkaz na příslušnou inode
  - **f_mode** - mód otevřeného souboru
  - **f_pos** - aktuální pozice v souboru
  - *další viz. include/linux/fs.h:776*
- **Superblock**
  - Identifikuje daný souborový systém na médiu - v paměti
  - **s_dev** - číslo identifikující zařízení
  - **s_blocksize** - velikost bloku
  - **s_type** - struktura popisující typ souborového systému
  - **s_magic** - *magické* číslo identifikující typ souborového systému
  - *další viz. include/linux/fs.h:1821*


file object
file object
file object
dentry object
dentry object
dentry object
inode object
inode object
vfsmount
super_block object

Další pojmy

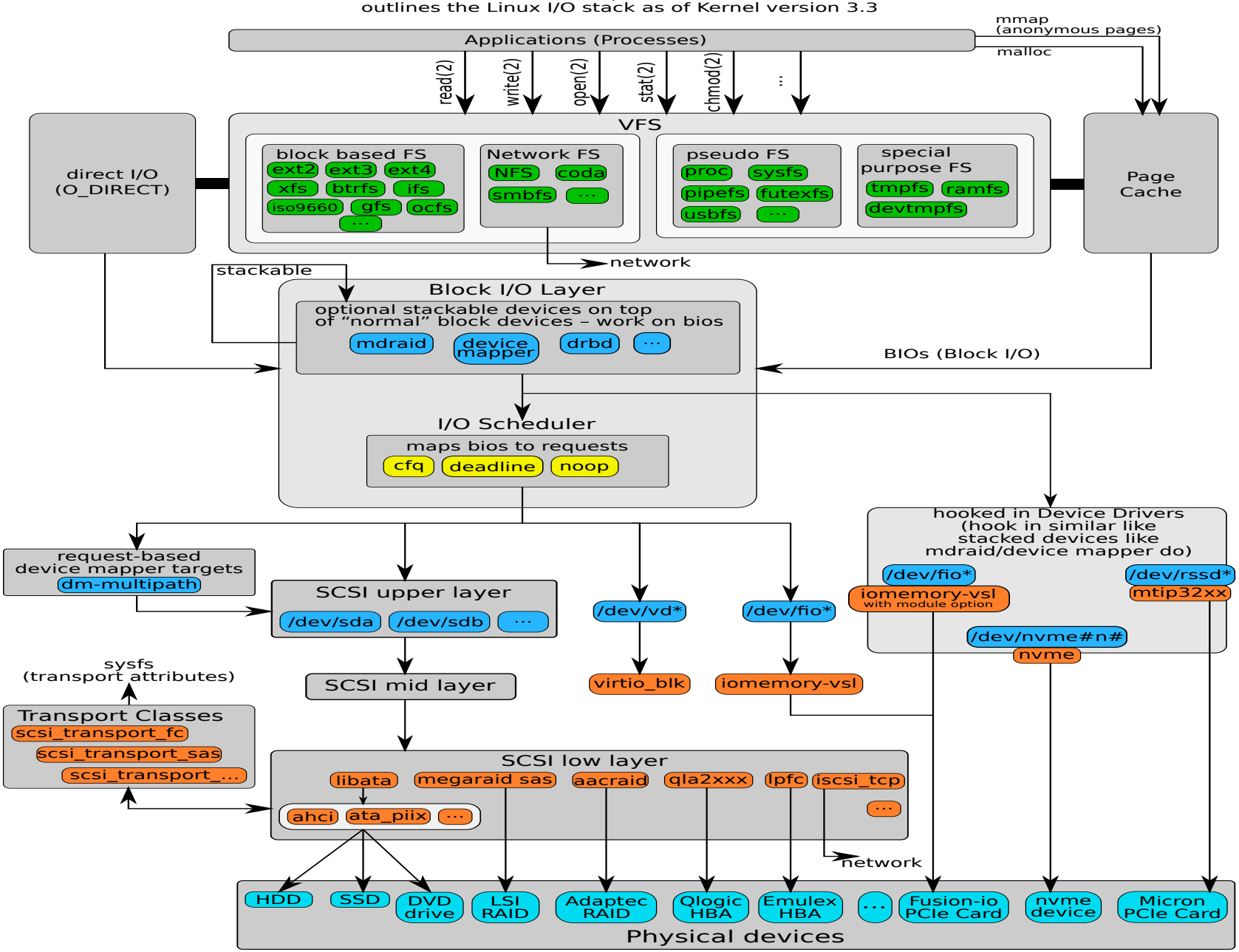
- **Blok**
  - Nejmenší alokovatelná jednotka souborového systému
- **Stránka**
  - Přesun dat mezi pamětí a záznamovým médiem

Part III
Rozhraní souborových systémů

The Linux I/O Stack Diagram

version 1.0, 2012-06-20

outlines the Linux I/O stack as of Kernel version 3.3



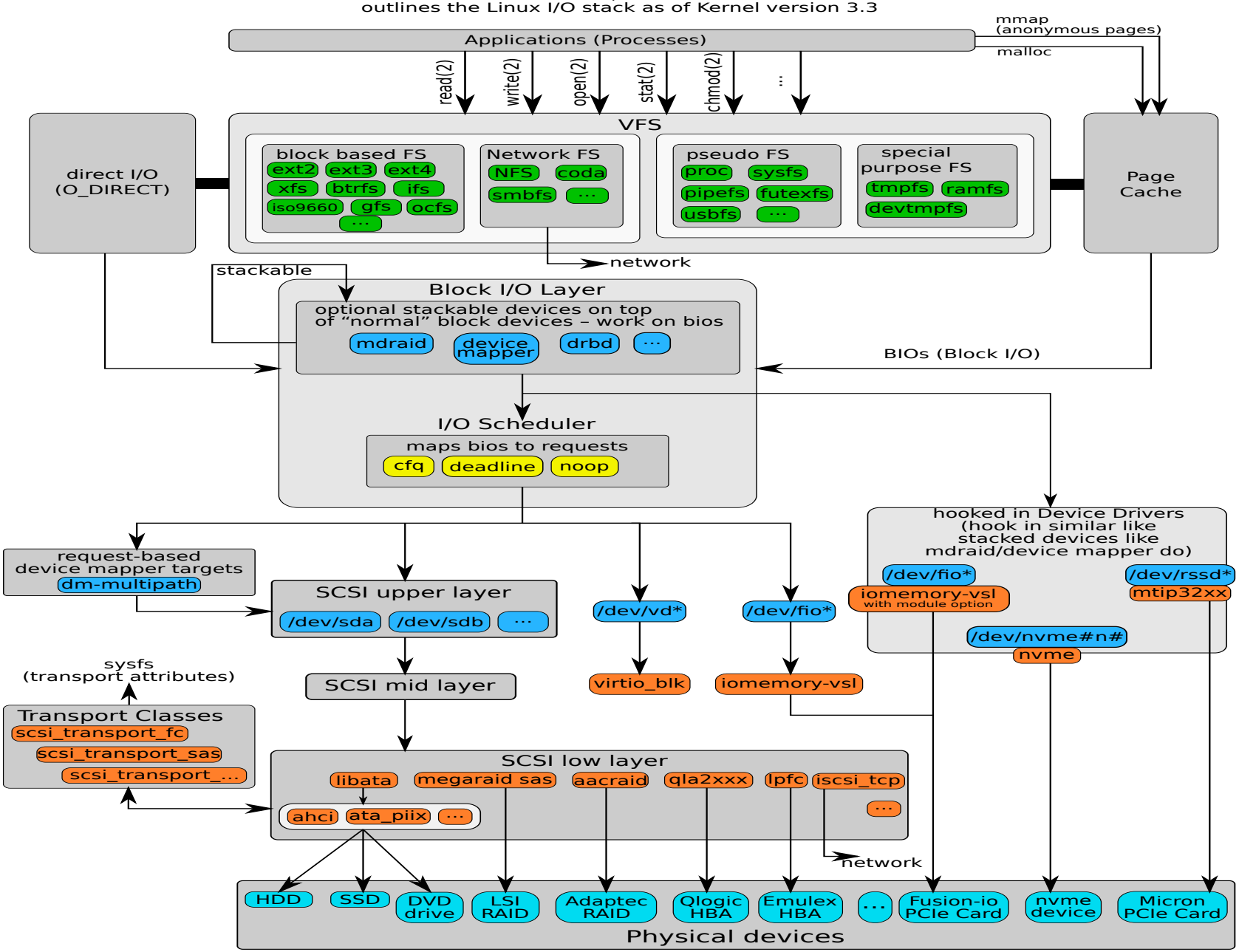
Rozhraní souborových systémů

- **Systémová volání**
  - Standardní rozhraní pro komunikaci s jádrem
  - read, write, stat, open, close, unlink, fallocate, ...
- **Input/output control - ioctl**
  - Původně sloužilo pro komunikaci s HW zařízením
  - Dnes se zneužívá jako "levné" rozhraní pro cokoliv
  - FIFREEZE, FITRIM, EXT4_IOC_RESIZE_FS, XFS_IOC_ZERO_RANGE, ...
- **procfs, sysfs**
  - Speciální soubory většinou slouží pro exportování informací do uživatelského prostoru */sys/fs/ext4/features/lazy_itable_init*
  - Někdy však i pro nastavení parametrů */sys/fs/ext4/sda1/extent_max_zeroout_kb*

The Linux I/O Stack Diagram

version 1.0, 2012-06-20

outlines the Linux I/O stack as of Kernel version 3.3



VFS - Virtual File System Switch

- **Pohled uživatelského prostoru**
  - Abstraktní vrstva poskytující jednotné uživatelské rozhraní mezi uživatelskými aplikacemi a různými souborovými systémy
  - Aplikace nemusí vědět zda přistupuje k lokálnímu, nebo síťovému souborovému systému
- **Pohled jádra**
  - Abstraktní vrstva poskytující jednotné rozhraní mezi jádrem a souborovými systémy
  - Poskytuje funkce společné pro všechny souborové systémy
  - Usnadňuje vývoj nového souborového systému
- **Objektově orientovaný přístup**
  - file_operations
  - inode_operations
  - dentry_operations
  - super_operations
  - address_space_operations

File operations

```
struct file_operations {
    ....
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t,
                      loff_t *);
    int (*mmap) (struct file *, struct vm_area_struct *);
    int (*open) (struct inode *, struct file *);
    int (*fsync) (struct file *, loff_t, loff_t, int datasync);
    int (*fasync) (int, struct file *, int);
    long (*fallocate)(struct file *file, int mode, loff_t offset,
                      loff_t len);
    ...
};
```

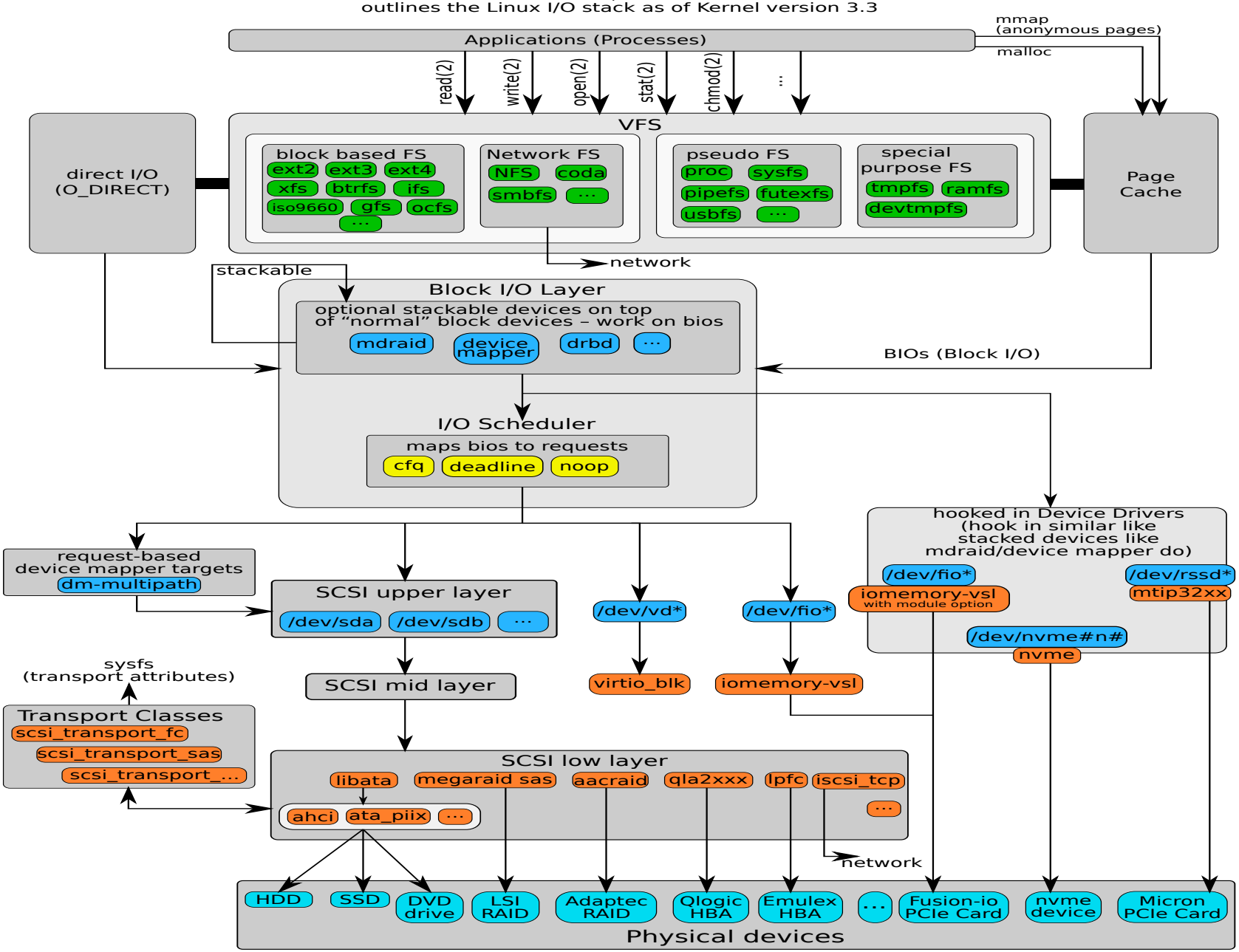
VFS - Virtual File System Switch

- **Dentry cache**
  - Spravuje hashovací tabulku adresářových záznamů
  - Při vytváření cesty k inode jsou uloženy všechny prvky cesty
  - Při vytvoření záznamu v dentry cache je zároveň vytvořen příslušný záznam v inode cache
- **Inode cache**
  - Spravuje hashovací tabulku inode
  - Urychluje přístup k inode
  - Novou, prázdnou inode naplňuje sám souborový systém

The Linux I/O Stack Diagram

version 1.0, 2012-06-20

outlines the Linux I/O stack as of Kernel version 3.3



Page Cache

- Disková cache - urychluje opakovaný přístup k datům na médiu
- Používá se pro všechny přenosové operace (kromě operace přímého přístupu direct I/O)
- Se stránkami se opět pracuje pomocí algoritmu LRU
- **Inode address_space**
  - Obsahuje mimo jiné odkaz na strom stránek příslušících danému souboru
  - **address_space_operations** pro manipulaci se stránkami
    - readpage, writepage, invalidatepage
- Ulehčuje práci souborovému systému - práce se stránkami
- **Příznaky stránek**
  - PG_dirty, PG_uptodate, PG_locked, PG_active, ...

Page cache

```
# free -m
             total       used       free     shared    buffers     cached
Mem:          7685       1606       6079          0        171       1098
```

- **Uvolnění inode cache a dentry cache**

```
# echo 2 > /proc/sys/vm/drop_caches
# free -m
             total       used       free     shared    buffers     cached
Mem:          7685       1606       6079          0        171        219
```

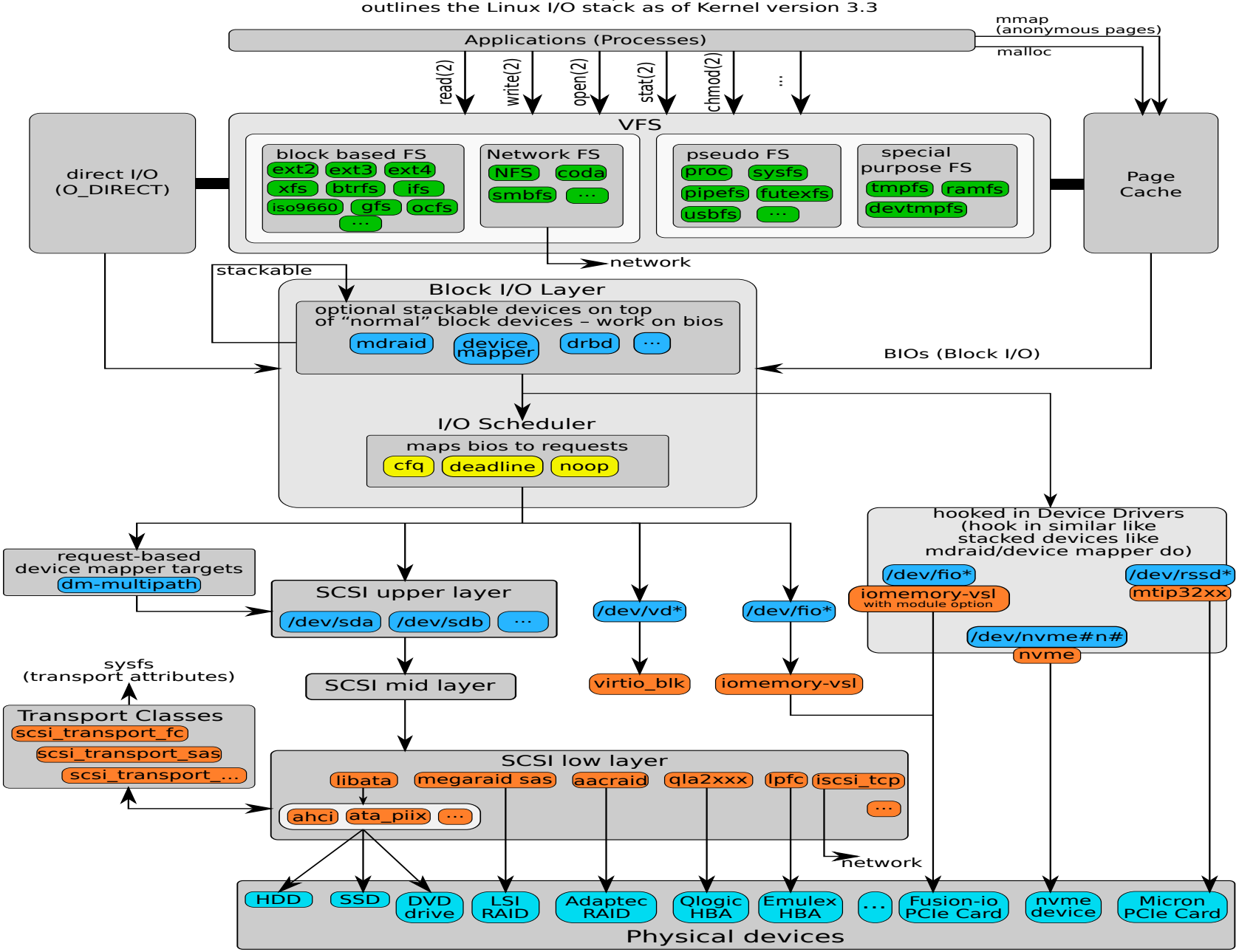
- **Uvolnění page cache**

```
# echo 1 > /proc/sys/vm/drop_caches
# free -m
             total       used       free     shared    buffers     cached
Mem:          7685       1606       6079          0          0        209
```

The Linux I/O Stack Diagram

version 1.0, 2012-06-20

outlines the Linux I/O stack as of Kernel version 3.3



Bloková vrstva

- Zprostředkovává rozhraní mezi souborovými systémy a ovladači blokových zařízení
- **Fronty**
  - Vkládání požadavků do fronty k odeslání do ovladače zařízení
- **I/O plánovače**
  - Připojují a odpojují frontu za účelem seskupování požadavků do dávek
  - Přehazování a spojování požadavků podle potřeby a možností
  - no-op, deadline, cfq
  - `/sys/block/sda/queue/scheduler`
- **Virtuální bloková zařízení**
  - mdraid
  - device mapper

Part IV
Interní struktury

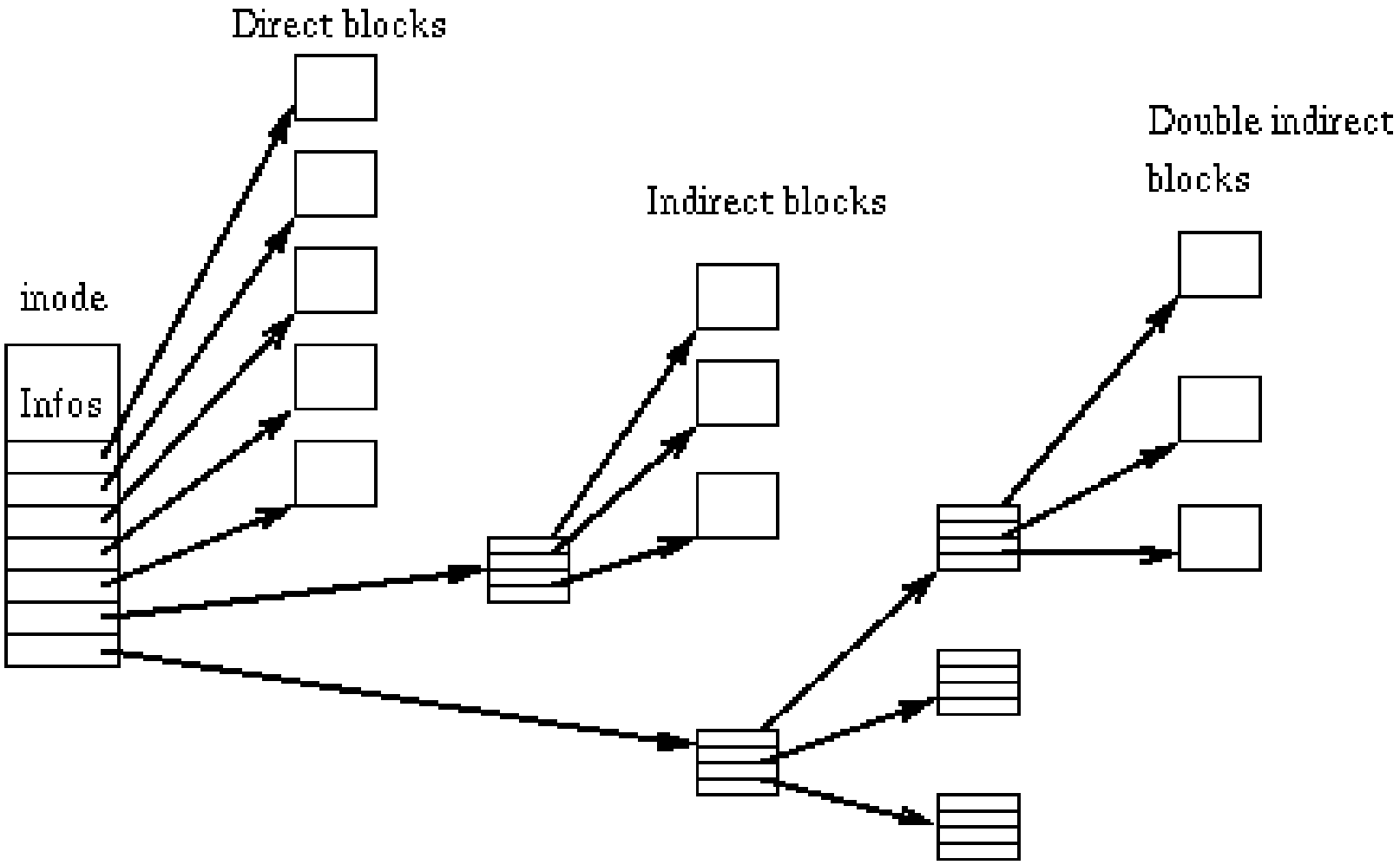
Interní struktury

- **Správa volného místa**
  - Historicky - spojitá alokace, spojový seznam, spojový seznam s tabulkou
  - Bitmapy, rozsahy bloků uložené ve stromech
  - Logický adresový prostor bývá často rozdělen do alokačních skupin
- **Alokátor**
  - Kritická část souborového systému
  - Alokuje boky se zásoby volných bloků
  - Snaha o prostorovou optimalizaci - zajištění lokality dat a metadat
  - Definuje rozložení na disku
- **Diskový formát**
  - Definuje formát a rozložení metadat na disku
  - Statické / dynamické rozložení

Adresování bloků v souboru

- **Přímé/nepřímé adresování bloků**
  - Jednoduchá implementace
  - Efektivní pro malé soubory

Direct/Indirect Block Addressing

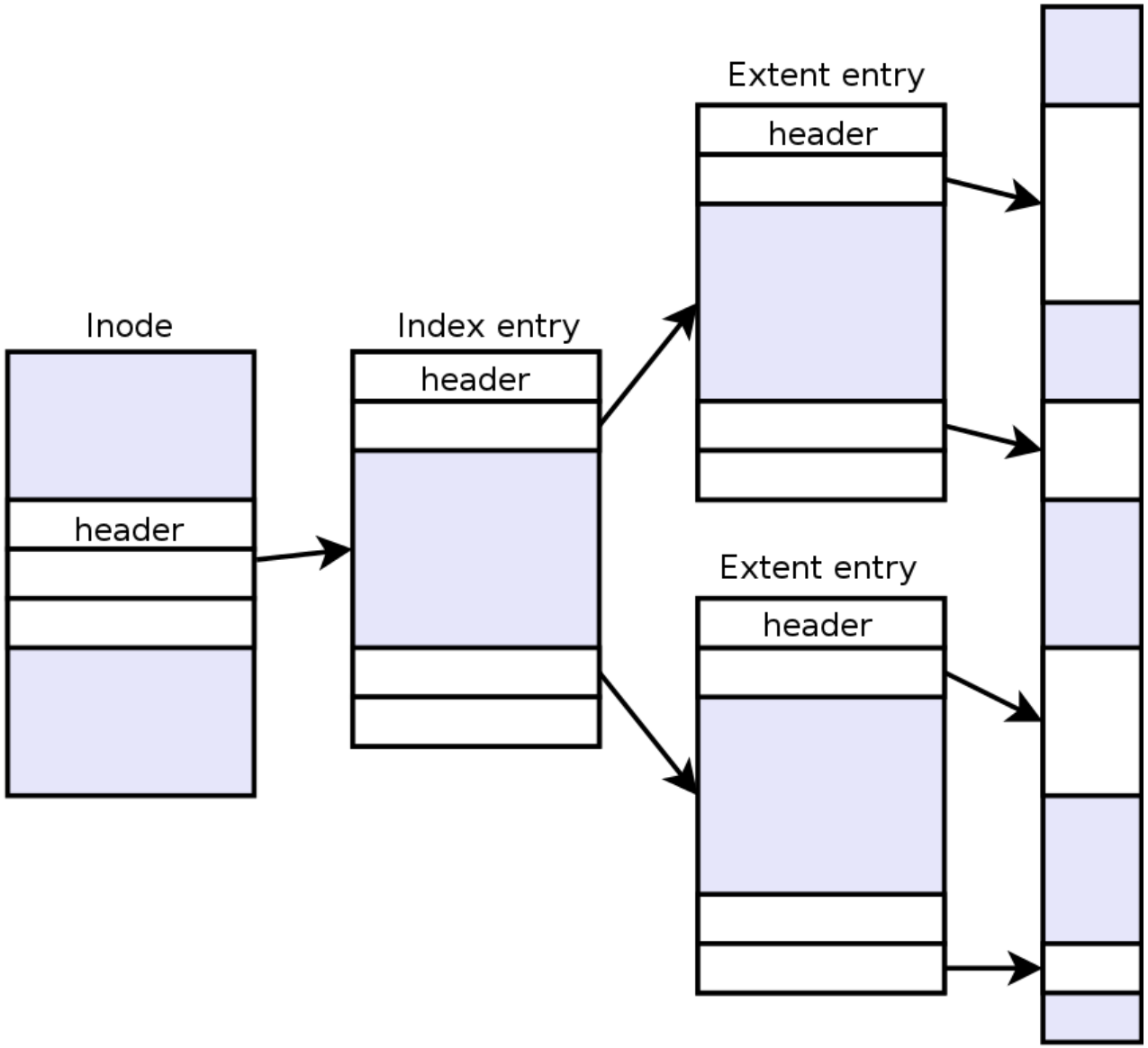


redhat.

Adresování bloků v souboru

- **Přímé/nepřímé adresování bloků**
  - Jednoduchá implementace
  - Efektivní pro malé soubory
- **Extent tree**
  - Mnohem složitější implementace
  - Efektivnější pro velké soubory
  - Menší množství metadat - záleží na fragmentaci souboru

Extent Tree


Inode
header
Index entry
header
Extent entry
header
Extent entry
header

redhat.

Part V
Konzistence při výpadku

Co se stane při výpadku

- Změny, které jsou v page cache nebudou zapsány na disk
- Změny, které jsou ve write cache zažízení při výpadku napájení nebudou zapsány na disk
- Některé bloky budou zapsány jen částečně
- **Nástroj kontroly systému by měl být schopen problém opravit, ale...**
  - Velikost úložných zařízení neustále roste
  - Velké časová a paměťová náročnost
- Je potřeba zajistit neustálou konzistenci **metadat** na disku

Žurnál

- Cyklický buffer na disku
- **Změny jsou seskupeny do transakcí**
  - T_RUNNING - transakce je otevřná a přijímá další změny
  - T_LOCKED - transakce již nepřijímá změny ale stávající ještě nejsou dokončeny
  - T_FLUSH - transakce je kompletní a je zapisována do žurnálu
  - T_COMMIT - transakce je kompletně zapsána na disk a změny můžou být přesunuty na své místo na disku
  - T_FINISHED - bloky transakce jsou kompletně zapsány na svém místě, bloky můžou být uvoněny z žurnálu
- Žurnál může být na jiném zařízení, než je samotný souborový systém
- Žurnál může být použit pro více souborových systémů zároveň
- Souborový systém může používat více žurnálů, např. jeden pro alokační skupinu

Copy-on-write

- Nejlzajímavější vlastnost souborového systému **btrfs**
- Bloky nejsou přepisovány na místě, ale je vždy vytvořená nová verze
- Při výpadku návrat k poslednímu konzistentnímu stavu
- **btrfs**
  - Několik druhů stromů s různými typy metadat
  - **root tree** obsahuje kořeny všech ostatních stromů
  - **root tree** kořen je uložen v superbloku
  - Po zápisu nového **root tree** do superbloku začíná nová transakce

Soft updates

- Striktně uplatňuje řazení změn metadat
- Složitá implementace
- Některé typy metadat nemusí být řazeny
  - Alokace bloků
- Stále vyžaduje kontrolu konzistence po pádu
  - Uvolnit nikým nepoužité alokované místo
- Výkonnější zejména pro fsync
- FreeBSD - souborový systém UFS

Part VI
Pokročilé funkce

Pokročilé funkce

- **Delayed allocation**
  - Není nezbytně nutné alokovat místo pro soubor ve chvíli kdy je do něj zapsáno
  - Bloky jsou pouze rezervovány
  - Alokace bloků probíhá až při zápisu na disk (samozřejmě asynchronně)
  - Seskupení menších alokací do větších celků - lepší pro alokátor
  - Některé bloky nemusí být zapsány vůbec pokud jsou předtím uvolněny
- **Data/Metadata Checksumming**
  - Ochrana metadat a dat před neplánovanými změnami díky chybám v paměti, nebo na disku
  - První krok ke kontorle konzistence dat/metadat za běhu
  - Jeden z taháků souborového systému btrfs
  - Metadata chsksumming dnes umí i ext4 a xfs

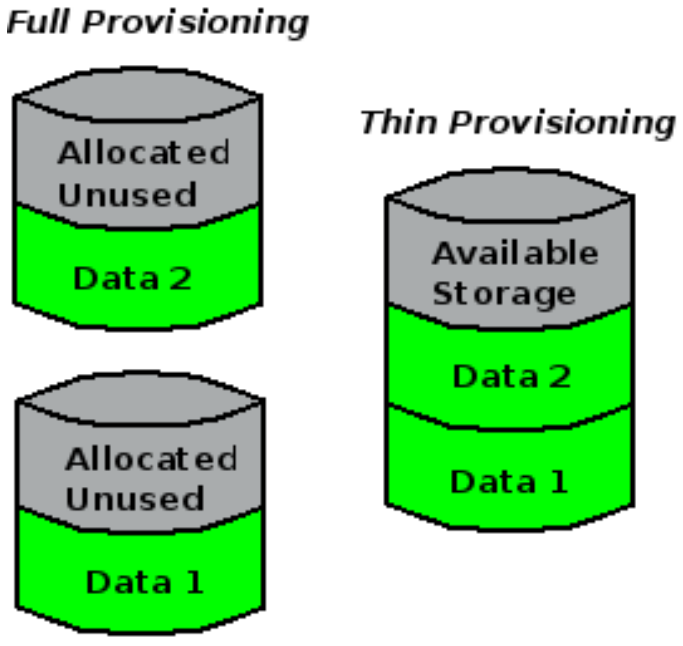
Pokročilé funkce pokračování...

- **Kontrola za běhu**
  - Kontrola metadat
  - Kontrola dat - file system scrub
  - Možnost zabránit nechtěným změnám ještě předtím než budou zapsány na disk
  - Oprava vadných metadat po přečtení z disku
    - Pro efektivnost vyžaduje selfdescribing metadata
- Snapshotting, multi disk support, ...

Part VII
Nové typy zařízení

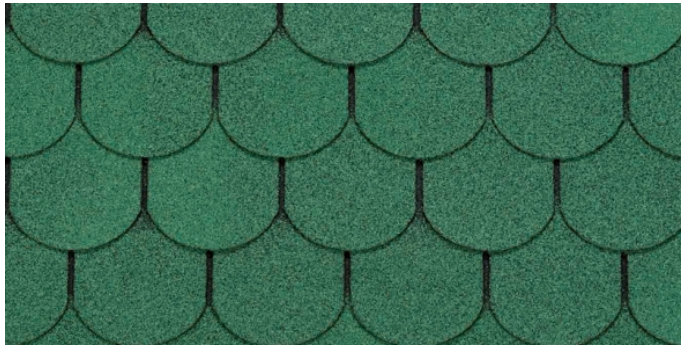
Thin provisioning

- Alokace na požádání
- Efektivní využití prostotu na úložném zařízení
- **Dnes dostupné:**
  - Specifická implementace výrobce ve formě velký a drahých úložišť
  - dm-thinp



Shingled Magnetic Recording - SMR Drives

- Vyšší hustota zápisu a tím i vyšší kapacita než u konvenčních magentických disků
- Omezení ve způsobu zápisu
- Přepsání jedné sopy poškodí přilehlé stopy
- Rozděleno do zón
- **Implementace**
  - Drive-Managed SMR
  - Host-Aware SMR
  - Host-Managed SMR / Restrictive mode



Persistent Memory

- Cena a výkon jako DRAM
- Zachovává stav po vynutí napájení
- Adresovatelná po Bytech
- Můžete si to představit jako DRAM s baterií
- **Komplikace**
  - Nechceme aby data byla ukládána v page cache
  - Nechceme aby bloková vrstva jakkoliv zasahovala
  - Velmi odlišné od toho co známe v oblasti úložných zařízení
  - Potřebujeme nový souborový systém ?

Persistent Memory

- Je to problém **za hranicí** pouhé **optimalizace**
  - Konvenční disk 75 - 100 IOPS
  - SSD 5k - 100k IOPS
  - PCIe SSD 100k - 1 000k IOPS
  - Persistent Memory může dosáhnout až **desítky milionů** IOPS
- To nejlepší co můžeme udělat je jít z cesty
- Naštěstí již máme nějaké rozhraní v souborových systémech
  - mmap
  - XIP - Execute in place
  - Ext2 již podporuje XIP
  - Pracuje se na podpoře XIP pro ext4

Part VIII
Jak se zapojit

redhat.

Jak se zapojit

- **Linux Kernel code** http://kernel.org
- **Kernel mailing lists** http://vger.kernel.org
  - linux-fsdevel
  - linux-ext4
  - linux-btrfs
  - linux-xfs
- **Kernel newbies** http://kernelnewbies.org/
- **Diplomky / Bakalářky**
  - http://fedora.cz/diplomky/
  - Valsní nápady na projekt v souborových systémech
  - Lukáš Czerner <lczerner@redhat.com>

Part IX
Otázky

Resources

- **Linux Weekly News** http://lwn.net
- **Linux Kernel code** http://kernel.org
- **Linux IO stack diagram**
  - http://www.thomas-krenn.com/en/oss/linuxiostack-diagram.html
- **Fuse diagram**
  - https://en.wikipedia.org/wiki/File:FUSE_structure.svg
- **VFS diagram**
  - http://www.ibm.com/developerworks/library/lvirtual-filesystemswitch/
- **Developers Conference** http://devconf.cz/
- Lukáš Jelínek. **Jádro systému Linux**, Computer press, 2008

The end.

Thanks for listening.