

# HardDisky

## Plotnové

- <http://blog.backblaze.com/2014/01/21/what-hard-drive-should-i-buy/>

## SSD

- ve SMARTu je potreba sledovat
  - [http://en.wikipedia.org/wiki/Wear\\_leveling](http://en.wikipedia.org/wiki/Wear_leveling)
  - Wear leveling count
    - pocet prepisu najstarsi bunky.
    - z toho se pocita normalizovana hodnota, ktera klesa od 100% u noveho disku k 0% pak uz je to otazka...
  - Media Wearout Indicator
    - Asi neco podobnyho, ale presne nevim... evidentne maj disky bud jedno nebo druhy
- Znacky
  - aktualne mam velmi dobre skusenosti s Samsungom, Samsung SSD 840 PRO Series
  - od znamych mam este info ze intely 500 a vyssia rada su gut tiez

## SW Aspekty

### Rozdeleni disku

- **MBR** (disky ≤2TB)
  - fdisk, sfdisk, cfdisk
  - kopie rozdeleni a → b: sfdisk -d /dev/sda | sfdisk /dev/sdb
- **GPT** (disky >2TB)
  - gdisk, sgdisk, cgdisk
  - kopie rozdeleni a → b: sgdisk -R=/dev/sdb /dev/sda; sgdisk -G /dev/sdb
  - jina varianta pokud predchozi nefunguje: a → b: sgdisk --backup=table /dev/sda; sgdisk --load-backup=table /dev/sdb

### RAID pomoci mdadm

- [https://raid.wiki.kernel.org/index.php/RAID\\_setup](https://raid.wiki.kernel.org/index.php/RAID_setup)
- Partition type: Linux RAID autodetect **0xFD** (0xFD00 u GPT)
- mdadm --create /dev/md0 --level=1 --bitmap=internal --raid-devices=2 /dev/sda1 missing
- mdadm --manage /dev/md0 --add /dev/sdb1
- mdadm --manage /dev/md0 --fail /dev/sdb1
- mdadm --manage /dev/md0 --remove /dev/sdb1
- mdadm /dev/md2 --fail /dev/sdc3 --remove /dev/sdc3
- mdadm --grow --bitmap=internal /dev/md0

- `mdadm --grow --bitmap=none /dev/md0`
- `mdadm --monitor --scan -l -t` (test sending of error e-mails)
- `echo "idle" > /sys/block/md0/md/sync_action` (defer active resync)
- `sysctl -w dev.raid.speed_limit_min=500000000; sysctl -w dev.raid.speed_limit_max=500000000` (unthrottle raid sync)

## Přidání dalšího disku do raidu s UEFI boot

- `mdadm --grow --raid-devices=2 --force /dev/md127` (-force je potřeba pokud byl původně raid1 založen jen s jedním diskem)
- `mdadm --manage /dev/md127 --add /dev/sdb3`
- `umount /boot/efi` - je potřeba odpojit původní EFI partiton, která je pravděpodobně na sda2
- `mkfs.vfat /dev/sdb2` - musíme připravit EFI partici na novém disku
- `mount /dev/sdb2 /boot/efi` nyní připojíme novou EFI partici do původního umístění.
- `grub-install /dev/sdb`
- standardně nepotřebujeme mít připojenou /boot/efi v FSTAB, ale je potřeba myslet na to, že při upgradu kernelu musíme OBE partice postupně připojit a zapsat na neaktuální verzi grubu !!
- pro /boot/efi přidat do fstab options nofail > `PARTUUID="eeefac33-598c-1540-acfa-e401d9d44d15" /boot/efi vfat defaults,nofail 0 1`
- overit, jestli v /boot/efi na sdb2 je nějaký obsah (musí tam být adresář EFI) pokud tam není, přepokopírujte z disku sda2 !
- nakonec pro jistotu zavolat `update-grub`

Pri tomto postupu dokážeme nabootovat do záložního kopie systému v případě potíží s primárním diskem.

## Zvětšení RAID pole

Chceme zvětšit pole např. o velikosti 500G na nové disky o velikosti 1TB. Ideální je to dělat na degradovaném poli, protože tím získáme zároveň zálohu původního pole.

- Vyměníme jeden z disků za větší. Po startu systému bude pole degradované. Na novém disku vytvoříme nové partice typu Linux RAID autodetect **0xFD** (0xFD00 u GPT) po celé délce disku, případně jak potřebujeme, pokud raidů máme více. Bude samozřejmě větší než aktivní partice v raidu.
- Novou partici přidáme do raidu - `mdadm --manage /dev/md0 --add /dev/sdb1` a počkáme, až se pole syncne.
- Nezapomeneme zapsat grub na nový disk, abychom po dokončení operace nastartovali systém - tj. `grub-install /dev/sdb!!!`

**následující operaci bude nutné provádět v jiném systému = např. nějaká live distribuce s podporou raidu, např. <https://partedmagic.com/downloads/>**

- Odebereme původní menší disk a přidáme druhý větší disk
- Nastartujeme live distribuci a složíme pole. Bude nyní degradované s novým diskem.
- Zadáme `mdadm --grow /dev/md0 --size=max` (zvětšíme pole na maximum velikosti partice)
- Kontrola filesystemu - `e2fsck -f /dev/md0`
- Zvětšíme filesystem - `resize2fs -p /dev/md0`

- Upravíme partice na druhém disku a přidáme do raid pole viz. výše.
- Nyní můžeme restartovat zpět do původního systému, synchronizace raidu bude potom pokračovat. V biosu bude nutné bootování přepnout na disk, který jsme měnili jako první. Po nastartování systému bude nutné také zapsat grub na druhý disk.

## Pokud system nespustuje na degradovanem sw raidu

V Debianu Jessie se nám stalo, že po vyjmutí jednoho z disků začal debian startovat do ramdisku s chybou, že nemůže najít rootfs. Z nějakého důvodu nedošlo k automatickému startu degradovaného pole a proto bylo potřeba provést následující úpravu ramdisku:

- Přidat skript

[/etc/initramfs-tools/scripts/init-premount/mdadm-start](#)

```
#!/bin/sh
mdadm --run /dev/md*
exit 0
```

- `chmod 755 /etc/initramfs-tools/scripts/init-premount/mdadm-start`
- `update-initramfs -u`

Případnou chybu, že `/dev/md` je adresarem můžete ignorovat.

## LVM

- Partition type: Linux LVM **0x8E**
- **PV** `pvccreate /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1`
  - overime `pvs` nebo `pvdisplay`
  - smazem `pvremove ...`
- **VG** `vgcreate fileserver /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1`
  - overime `vgs` nebo `vgdisplay`
  - autodetekce `vgscan`
  - přejmenujem `vgrename fileserver data`
  - smazem `vgremove fileserver`
- **LV** `lvcreate -name backup -size 5G fileserver`
  - overime `lvs` nebo `lvdisplay`
  - autodetekce `lvscan`
  - přejmenujem `lvrename fileserver backup zalohy`
  - smazem `lvremove /dev/fileserver/backup`
  - zvetsime
    - `pvresize /dev/sdb1` roztáhne PV přes celou zvětšenou partici
    - `lvextend -L5.5G /dev/fileserver/backup`
    - `lvextend -L+5.5G /dev/fileserver/backup` zvetsim o dalsich 5.5G
    - `e2fsck -f /dev/fileserver/backup`
    - `resize2fs /dev/fileserver/backup`
      - **POZOR: U ext4 provadej resize2fs online - nejprve mount napr do. /mnt**

- xfs\_growfs /mnt zvetsi xfs pripojene do adresare /mnt
- zmensime (**opatrne!**)
  - e2fsck -f /dev/fileserver/backup
  - resize2fs /dev/fileserver/backup 10485760
  - lvreduce -L5G /dev/fileserver/backup

## LVM Thin

- **LV** lvcreate -L 100G -n data pve
- **LV → thin-pool** lvconvert -type thin-pool pve/data
  - Thin-pool muzem nyini pridat do proxmoxu a ten si v nem bude delat thinlv pro virtualy a kontejnery
  - Nebo si v nem vytvorime vlastni thinlv
- **ThinLV** lvcreate -n thin1 -V 1T -thinpool data pve
- **Zobrazit lvm vcetne thin-pool** - lvs -a
- **Zvetseni thin-poolu o 256G** - lvextend -L+256G /dev/vg/thinpool - zmensovat nejde
- **Zvetseni metadat thin poolu** - lvextend -poolmetadatasize +100M vg/thinpool
- **Zvetseni vcetne metadat** - lvresize -size +<size[\M,G,T]> -poolmetadatasize +<size[\M,G]> <VG>/<LVThin\_pool>
- Oprava (POZOR! OPATRNE!)
  - Prohodit jiny LV k pouziti jako metadata thin poolu: lvconvert -thinpool <VG>/<THIN\_POOL\_LV> -poolmetadata <NOVY\_LV\_METADATA>
  - **Oprava LV** ~~lvconvert -repair <VG>/<LVThin\_pool>~~ ASI NENI UPLNE DOBRY NAPAD
  - <https://github.com/jthornber/thin-provisioning-tools/>
  - <https://www.unixrealm.com/?p=12000>

<https://www.redhat.com/archives/linux-lvm/2016-January/msg00010.html> So if you feel the time spend on thin\_checking doesn't pay-off - you can try to add option '-skip-mappings' (see lvm.conf field global/thin\_check\_options)

</etc/lvm/lvm.conf>

```
...
thin_check_options = [ "--skip-mappings" ]
...
```

## Ext4

Zvetsit zurnal:

```
tune2fs -O ^has_journal /dev/hdXX
tune2fs -l /dev/hdXX
tune2fs -J size=128 /dev/hdXX
```

## ZFS

- Rozdelime disky: typ partice BF(00) = Solaris Root (ale muzem pouzit i primo cely disky, v takovym pripade se partice vytvori samy)
- Vytvorime hlavni systemovy ZFS uloziste s mirrorem (neco jako VG u LVM)
  - `zpool create tank mirror /dev/sda /dev/sdb`
    - Nekdy je dobry pridat parametr `-o ashift=12`, kde hodnota `ashift` = exponent 2 pro zarovnani na sektory.  $2^9=512\text{B}$  sektory,  $2^{12}=4096\text{B}$  sektory,... Nejde to menit po naformatovani, nicmene default by mel bejt vic nez rozumny reseni, takze bych se spis vyhnul rejpani se v tomhle
    - Také se doporucuje pridavat disky podle id, např.: `/dev/disk/by-id/wwn-NECONECO` kvůli spolehlivější detekci
  - `#zpool set listsnapshots=on tank` neni potreba, tyka se jen vypisu "zfs list", jinak se snapshoty daj normalne i bez toho vypsati `ls -la /tank/.zfs/snapshot/`
  - `#zfs set recordsize=16k tank` (nastavit recordsize na stejnou velikost jako pouziva db. aplikuje se jen pro nove soubory: default=128k, innodb-data=16k, innodb-log=128k, myisam=8k, postgresql=8k, sysbench=16k)
  - `#zfs inherit recordsize tank` (nastavit recordsize na vychozi hodnotu)
  - `zfs set atime=off tank` (vypne atime)
  - `#zfs set logbias=throughput tank` (vypne dvoji zapis dat pres SLOG, nejsem si jisty, ze je to vzdy dobre pro vykon, zalezi na konfiguraci SLOGu)
  - `zfs set compression=on tank` (default kompresni algoritmus by mel bejt vcelku rychleji a napr. eliminuje dlouhy retezce nul)
  - `#zfs set dedup=on tank` #deduplikace zere MOOC ramky = asi 8GB ram na 1TB dat! pomoci prikazu `zdb -S tank` lze zjistit jestli se deduplikace vyplati (pokud to vypise `dedup=2.00` nebo vic, tak se vyplati. Pokud vynasobime total allocated blocks cislem 320, vyjde nam potrebna ramka)
  - `zpool list` a `zpool status`
  - `zfs get all tank`
- V tomto ulozisti vytvorime ZFS pro LXC a pripojime
  - `zfs create tank/vps`
  - **Pokud pouzivame proxmox, tak tady zkoncime a zbytek naklikame v proxmoxu!**
  - `zfs set mountpoint=/var/lib/lxc tank/vps`
  - mountpoint nesmi existovat, ZFS si automaticky adresar vytvori pri pripojeni a smaze pri odpojeni.
- Vytvorime novej virtual s quotou (tohle uz dela LXC automaticky)
  - `zfs create tank/vps/test`
  - `zfs set quota=10G tank/vps/test`
- `Fsck: zpool scrub tank` (prubeh sledujem v `zpool status`)
- Vsechno smazem (neni dobry napad): `#zpool destroy tank`
- `zpool status` Vypise vsechny disky v poolu, jestli nedochazi k datovym chybam, jestli probiha scrub, resilver, atd...
- Log a cache na SSD
  - Typy pomocnejch devicu
    - "log", take nazyvany "zil" nebo "slog" je zurnal synchronnich zapisu
    - "cache" neboli "l2arc" ("level 2 arc") je cache pro cteni
    - ani u jednoho nevadi vypadek, v cache jsou jen data precteny z rotacniho disku a vsechno co je v logu je i v RAMce, log by se pouzil jen pri vypadku napajeni, kdy bysme o RAMku prisli
  - Dimenzovani

- velikost logu staci cca 0.6 GB na kazdy gigabit sitovky co mame na systemu. 16GB by tedy melo s obri rezervou stacit i pro pripady ze budem mit 2x10G sitovku. Flushuje se jednou za 5 sekund, takze staci kdyz se do nej vejdou vsechny synchronni zapisy co probihaj za 5 sekund. (= ano, vic nez 16GB tezko budem v soucasnosti potrebovat, i to je fakt hodne, prakticky staci asi 1GB nebo min)
- na cache je potom dobre vyuzit vsechno misto co nam na SSD zbyde po logu (cim vic, tim lip. dava smysl, aby byla cache radove vetsi nez log i L1ARC, ale jejich velikosti spolu primo nijak nesouvisi)
  - log a cache vzdy pridavame jako /dev/disk/by-id/ nemaji metadata!!!
  - zpool add tank log /dev/loop0
  - zpool add tank cache /dev/loop1
  - zpool iostat -v vypise obsazeni logu a cache (a vseh jednotek v poolu)
  - zpool remove tank /dev/loop0 /dev/loop1
- Vymena disku v RAIDu (konzultovat s harviem!!!)
  - zpool detach tank sdb2
  - zpool attach tank sda2 sdb2 (asi dava spis smysl attachnout celej prazdnej disk, oddily si vytvori ZFS samo)
- Aktivace automatickeho zvetsovani zrcadla (asi dobry udelat uz pred vymenou disku za vetsi)
  - zpool set autoexpand=on tank
- ZVOL (= neco jako LVM uvnitr ZFS poolu, doporučený na swapy a image virtualu!)
  - zfs create -V 5gb tank/vol (vytvori jednotku /dev/zvol/tank/vol, taky znamy jako /dev/zd0, parametr -s udela zvol bez rezervace diskovoho prostoru v poolu = thin-provisioning)
  - zfs list -t volume vypiseme si zvoly (bez -t volume to vypise vse v poolu)
  - zfs destroy tank/vol
- Autodetekce existujiciho ZFS
  - zpool import nebo zpool import -a pro exportnuty
  - zfs mount -a
- SWAP na ZFS
  - Swapovani do souboru na ZFS neni podporovano, ale snad se da pouzít ZVOL jako swap
  - Nicmene jsou tam problemy (ZFS nekdy potrebuje pri zapisu do zvolu samo swapovat, coz je dost nemily predpoklad pro zapis do swapu = zpusobuje zamrznuti)
    - nekdo tvrdi, ze pomuze ZVOLu nastavit logbias=throughput a compress=zle, nicmene jsem netestoval
    - urcite to chce mit nastaveny hung\_task a reboot pri panicu
- Vylepseni pro backup destinace
  - zfs set readonly=yes tank/backup - zabranime systemu menit zalohy (recv stale funguje)
  - zfs set volmode=none tank/backup - schovame ZVOLy pred systemem (aby nam je nedetekoval treba mdraid nebo lvm)
  - Aby nemusela protistrana mit pristup na root, je mozne povolit userovi prava na konkretni dataset pomoci zfs allow
    - <https://github.com/oetiker/znapzend#running-by-an-unprivileged-user>
    - Sending end: destroy,hold,mount,send,snapshot,userprop
    - Receiving end: create,destroy,mount,receive,userprop
      - Eg.: zfs allow -u backup-pve1 create,destroy,mount,receive,userprop tank/backup/pve1
      - Verify: zfs allow tank/backup/pve1
- Testovani ZFS
  - Ztest NESLOUŽÍ k testování zfs modulu v jádře!!! Pro otestovani systemu je naprosto nevhodny.

- `ztest -f /tmp -V` vytvori v /tmp blockfly s testovacim ZFS a spusti na nem unit testy (musi tam byt dost mista).
- Jde prodlouzit cas v sekundach, defaultne `T 300`
- Dalsi zdroje
  - ZoL manual <https://pthree.org/2012/04/17/install-zfs-on-debian-gnulinux/>
  - Arch Linux ZFS <https://wiki.archlinux.org/index.php/ZFS>
  - Things Nobody Told You About ZFS <http://nex7.blogspot.cz/2013/03/readme1st.html>

Je dobry omezit kolik RAM muze sezrat ARC (defaultne si bere 1/2 veskery ramky na Linuxu a 3/4 na Illumosu):

[/etc/modprobe.d/zfs.conf](#)

```
#echo '(1024^3)*10' | bc
#update-initramfs -u -k all
options zfs zfs_arc_max=10737418240
```

## ZFS replikace

- zdroj# `zfs snapshot tank/vps/subvol-300-disk-1@mujsnapshot`
- zdroj\$ `su -c "zfs send tank/vps/subvol-300-disk-1@mujsnapshot" | pv | ssh strojcilovy sudo zfs recv -F tank/vps/subvol-300-disk-1`
- cil#

## NILFS2

- `mkfs.nilfs2 -L LABEL /dev/sdx1`
- `lscp /dev/sdx1` Vypiseme checkpointy (historii)
- `chcp ss /dev/sdx1 94` Nastavime checkpoint #94 jako snapshot (= nebude smazan cleanerem a pujde pripojit)
- `mount -o ro,cp=94 /dev/sdx1 /mnt/` Pripojime readonly snapshot #94
- `chcp cp /dev/sdx1 94` Po odpojeni zrusime snapshot #94 (udelame z nej zpatky checkpoint)

## SATA HotSwap

- `readlink /sys/block/sda` (zjisti na jaky sbernici je disk **sda**)
- `echo 1 > /sys/block/sda/device/delete` (odpoji disk **sda**)
- `echo "- - -" > /sys/class/scsi_host/host0/scan` (najde disky na sbernici **host0**)

## Fyzická identifikace disku

### Pomocí LED

Pokud má stroj modré identifikační led diody u jednotlivých šuplíčků disku, je možné je rozsvítit nebo

zhasnout následujícím příkazem z balíčku `ledmon`:

- `ledctl locate=/dev/sda`
- `ledctl locate_off=/dev/sda`

Je možné použít i hierarchické označení disku (např. `/dev/disk/by-id/[drive-id]`, atd...)

## Host Protected Area (HPA)

Precti si o tom něco než to začneš používat!

- Zakázat v GRUBu: `libata.ignore_hpa=1`
- Overit: `cat /sys/module/libata/parameters/ignore_hpa`
- Zjistit stav HPA u disku: `hdparm -N /dev/sd?`
- Nastavit HPA u disku na hodnotu XXX: `hdparm -N XXX /dev/sdx` (HPA se zruší nastavením hodnoty XXX na max co disk umí)
  - Pokud se má hodnota XXX zachovat permanentně i po rebootu, tak se musí uvádět s "p" jako `pXXX`, jde to udělat jen jednou za power-cycle.

## Migrace ext3 na ext4

Minimální požadavky: **kernel 2.6.30; grub 1.96+20090808; e2fsprogs 1.41.6; mount 2.16**

- `umount /dev/md5`
- `fsck.ext3 -ftv /dev/md5`
- `tune2fs -O extents,uninit_bg,dir_index /dev/md5`
- `fsck.ext4 -yDtv /dev/md5`

Podrobnosti na

[http://www.debian-administration.org/article/643/Migrating\\_a\\_live\\_system\\_from\\_ext3\\_to\\_ext4\\_filesystem](http://www.debian-administration.org/article/643/Migrating_a_live_system_from_ext3_to_ext4_filesystem)

## Migrace Windows 10 na menší SSD

Migroval jsem úspěšně 1TB HDD na 960GB SSD. Delat to z Windows je nemožné (nezkoušel jsem placené Windows nástroje třetích stran, ale všechny freeware nástroje i programy dodávané k SSD selhaly, přímo ve Windows není nic co by to zvládlo). Na Linuxu to jde, ale je to trochu mákačka. Můj postup byl následující:

- Na Windows 10 v nabídce Start najít CMD a "spustit jako správce"
- Na Windows pustit `chkdsk /f`, nabídne to naplánování opravy FS při dalším rebootu, rebootnout a zčekovat. Linux tohle neumí.
- Když je disk opravenější, může člověk naboťovat live Linux, použil jsem ArchBang. Má předinstalované vše potřebné, bezi z RAM a vejde se na flashku.
- Pomocí `cfdisk` si otevru starý a nový disk. Všechny oddíly vytvořím stejně na GPT(!), ale ten největší zmenším aby se to tam vše veslo. Velikosti zadávám v sektorech, stejně jako na původním disku. Používá se k tomu suffix "S", např.: `123456789S`. Bohužel to neumím naskriptovat, takže jsem to dělal docela ručně:

- Nastavim stejný typ u všech partic, zapisu a závru cfdisk.
- Pomocí sgdisk nastavím GUID disku (-U) i jednotlivých partici (-u) a jejich atributy (-A) tak, aby byly stejné jako na původním disku. Jinak to nebude bootovat!!! Je to mravenci práce, dělá se to po jednom, detaily v man sgdisk. Jáke GUID má původní disk jsem vycítal z cfdisku.
- Podívám se jak velká je cílová partic se systémem a rezíznu zdrojový systém aby se tam s rezervou vesel. Napr.: `ntfsresize -size 850G /dev/sda2`. Musí tam být volné místo aby to šlo. Toto selže, pokud jsi na začátku neudělal `chkdsk /f`
- Po rezíznutí znovu rebootnu do windows a znovu udělám `chkdsk /f` včetně rebootu jako předtím, aby se disk opravil. Pak se vrátím do linuxu.
- Pomocí `dd` přepíšu obsah jednotlivých partic kromě té největší systémové.
- Největší systémovou přepíšu pomocí `ntfsclone`, napr.: `ntfsclone --overwrite /dev/cíl /dev/zdroj`. Funguje to jako `dd`, ale nekopíruje to bloky nealokované fs, takže na prázdném fs to ušetří pulku času. Nechám běžet přes noc.
- `sync`
- Znovu rebootnu do windows a udělám `chkdsk /f+reboot`, vrátím se do linuxu
- Protože jsem si nechal pár GB rezervu a chci ji získat zpátky, tak na cílovém systému pustím `ntfsresize /dev/sdb2` na systémový oddíl, abych filesystem roztáhl opět přes celou novou partici
- Znovu rebootnu do windows a udělám `chkdsk /f+reboot`, windows by měly v pořádku bootovat
- Protože oba disky mají stejnou GUID, není možné je oba mít ve windows připojené najednou. Původní disk nechám 1-3 měsíce, než se zahřeje SSD a pak na něm přepíšu GPT s novým oddílem, ten naformátuju a mám externí disk na zálohy, nebo cokoliv jiného.

## Poznámky

- **Test rychlosti**
  - `hdparm --direct -t /dev/sd?`
- **Číslo ATA portu v dmesgu - převod na device**
  - `ata=4; ls -l /sys/block/sd* | grep $(grep $ata /sys/class/scsi_host/host*/unique_id | awk -F'/' '{print $5}')`
- **Souhrn co zapisuje na disk**
  - `iotop -aoP`
- **Kdo používá mountpoint**
  - `fuser -mv /mnt/point`
- **Záchrana dat**
  - <http://www.forensicswiki.org/wiki/Ddrescue>

From:  
<https://wiki.spoje.net/> - **SPOJE.NET**

Permanent link:  
<https://wiki.spoje.net/doku.php/howto/hw/disky?rev=1715098992>

Last update: **2024/05/07 18:23**

