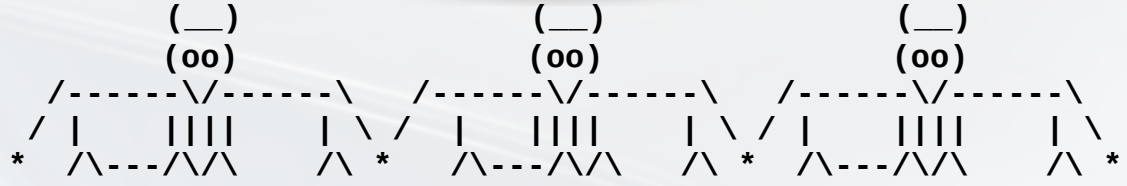
Giochiamo ai COW-boy
zfs – btrf



Alberto Maria Fiaschi

Prerequisiti

- Dispositivo a blocchi: una periferica leggibile/scrivibile per blocchi di grandezza fissa.
- Partizione : una suddivisione logica di un dispositivo a blocchi.
- File System: Una struttura per gestire informazioni memorizzate in un “supporto” di solito persistente. Un filesytem contiene dati (le nostre informazioni), e metadati (come ritrovare i nostri dati e altre proprietà)
- LVM: Substrato per una gestione dinamica delle partizioni.
- RAID: Distribuzione delle informazioni su più dischi per favorire le prestazioni o la sicurezza dei dati.

Cosa si intende per COW

- Con Copy On Write si intende una metodologia per cui i dati modificati non vengono sostituiti dove si trovano ma scritti in una nuova locazione.
- Di solito viene letto il dato originario, modificato in RAM e scritto altrove. Quindi sul disco sono presenti le vecchie e le nuove informazioni ….

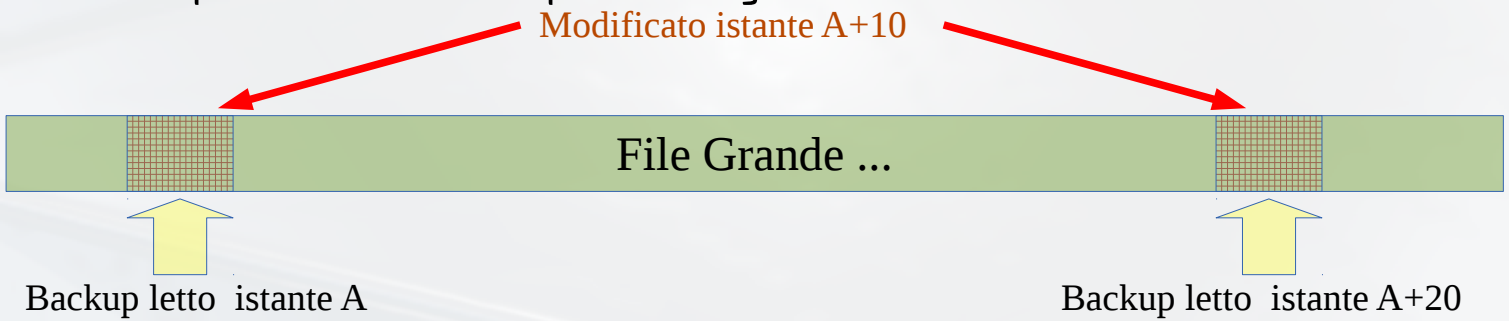
Ma non è meno efficiente, non è preferibile riutilizzare le stesse risorse ?

vantaggi del COW: riuso e coerenza

- Il fine è il riutilizzo delle zone di memoria che contengono le stesse informazioni. Se ho in memoria due copie dello stesso dato ne conservo una sola. Nel momento in cui una delle due copie diverge, allora la duplico con le differenze.
- In caso di crash i vecchi dati e metadati vengono preservati non essendo mai sovrascritti. Nei files system tradizionali basati su sovrascrittura di solito viene garantito la coerenza dei metadati tramite journaling . Per esempio ext4 ha le seguenti opzioni di mount : `Journal, Ordered (default), Writeback.`

Vantaggi del COW: Snapshot

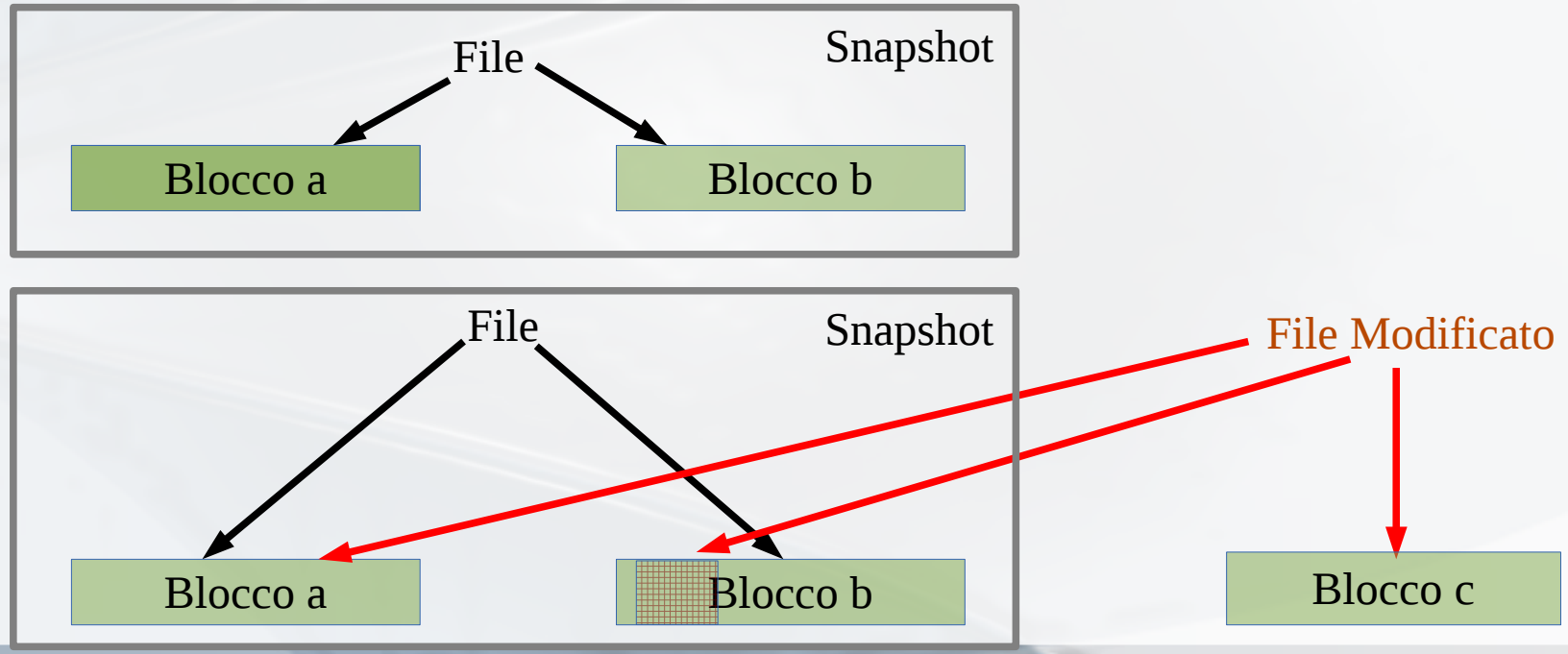
- Uno snapshot è una copia istantanea di tutto il filesystem. (operazione atomica).
- Permette di ripristinare/consultare velocemente il sistema all'istante salvato.
- Facilita le operazioni di backup on-line garantendone la coerenza. Ma non è il Backup !!



- i dati non sono mai modificati sul posto, quindi per creare uno snapshot basta modificare i metadati. Avrò una serie di blocchi su disco che verranno marcati come appartenenti allo snapshot.

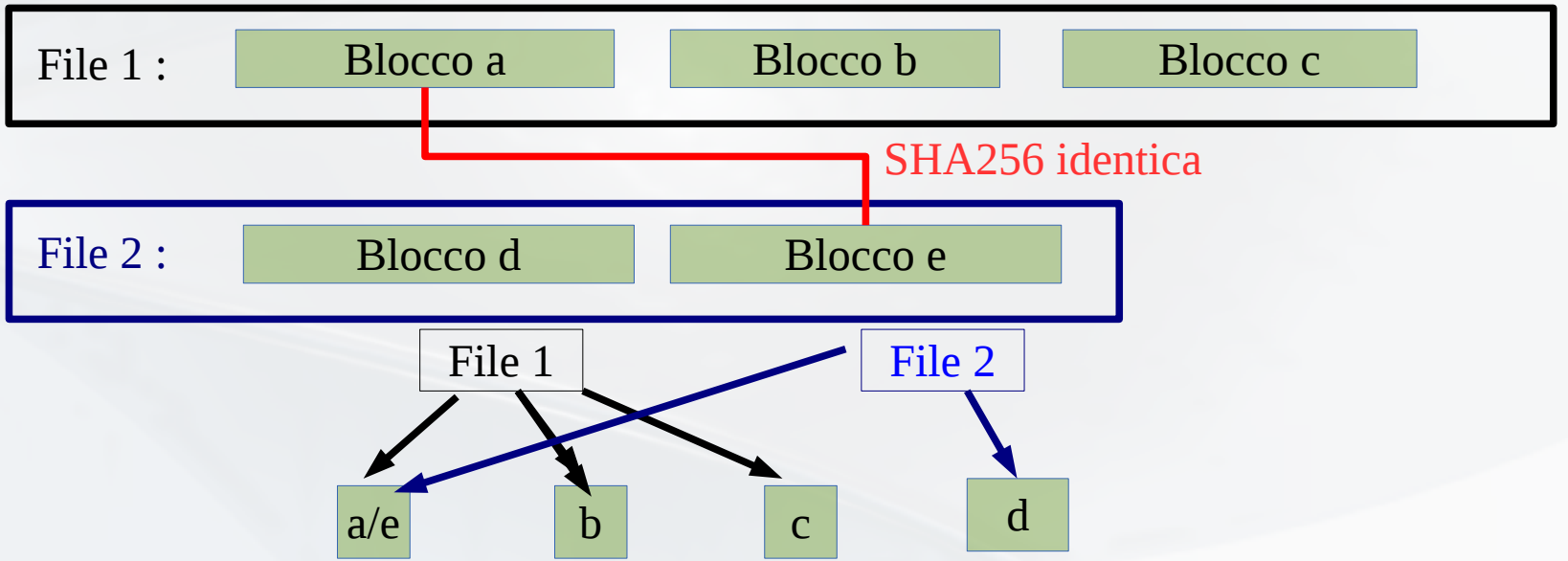
Vantaggi del COW: Riutilizzo

- Se modifico dei dati appartenenti ad un snapshot basta che salvi le differenze. Riutilizzando i blocchi presenti in totale o con offset.



Vantaggi del COW: Deduplica

- La condivisione di risorse è applicabile anche su file “scorrelati” fra loro ma che “casualmente” hanno dei blocchi uguali. confrontando le hash dei blocchi presenti su disco con quelle dei blocchi che devo scrivere posso sapere se posso riutilizzare un blocco già presente.



Vantaggi del COW: Wear leveling

- I dispositivi allo stato solido (SSD) supportano un numero limitato di scritture, quindi è necessario distribuirle equamente su tutta la superficie del disco al fine di garantire una vita più lunga al dispositivo. Nei file system di tipo COW questo è di più facile implementazione .(anche se molto dipende dal firmware del SSD)

Svantaggi del COW: Deframentazione

- I file system di tipo COW soffrono molto della frammentazione. In generale tendono avere accessi molto casuali in lettura e scritture sequenziali.
- Implementare la deframmentazione risulta più ostico rispetto ai file system basati su sovrascrittura, perché più metadati referenziano lo stesso blocco. Quindi sussistono esigenze diverse : rendo sequenziale la lettura per quale “volume - snapshot o clone“ ?

COW: Puliamo la stalla

Per poter liberare le risorse non più utilizzate è necessario contare quante volte un blocco è referenziato. I blocchi che hanno valore 0 nel loro "reference count" sono marcati come liberi per poter essere riutilizzati. Contare i riferimenti funziona perché i metadati sono rappresentati tramite **grafi aciclici diretti** quindi non sorgono problemi indotti dalla ricorsività. (Semplificando uno snapshot può contennere un file ma un file non può contenere uno snapshot).

ZFS & BTRFS: Funzionalità

- Sostituzione di LVM. Con snapshot più efficienti.
- Sostituzione di Raid Software (mdadm) .
- Integrità:checksums di dati e metadati.
- Continuità del servizio: deframmentazione, riparazione, bilanciamento, ridimensionamento on-line.
- Compressione.
- Deduplica (btrfs per ora solo batch).
- ZFS: Integrazione con condivisomi di rete (NFS,SAMBA).

ZFS :Zettabyte filesystem

- Dimensione massima (device virtuale) 2^78 byte circa 300 miliardi di TB. (per un volume 2^64 byte)
- Non usa extend ma blocchi di dimensione variabile.(struttura a blocchi indiretti come ext2) .
- ZFS integrato in Open Solaris nel 2005 (inizio sviluppo 2001)
- Licenza : Common Development and Distribution License (CDDL) incompatibile con GPL.
- ZFS on Linux http://zfsonlinux.org/ si basa sul wrapper Solaris Porting Layer (SPL) per implementare le Solaris kernel APIs in Linux.

ZFS :device virtuale

- zpool :device virtuale costituito da uno o più device fisici. Può garantire la sicurezza dei dati mediante “ridondanza”.

Modalità	Tolleranza	Min./raccomandato
“unione”	**A livello raid nessuna…**	1
mirror	**1 disco**	2
raidz(1)	**1 disco**	2/3
raidz2	**2 dischi**	3/4
raidz3	**3 dischi**	4/5

- Altra ridondanza non basata su parità/raid attivabile tramite propietà zfs copies =1 | 2 | 3

ZFS : Comandi

Ci sono solo due comandi per gestire un file system zfs
…ma con tanti “*sotto-comandi*” ;-)

- **zpool** : gestisce il device virtuale. Esempio creazione di un pool:
  `zpool create “dati_zfs” raidz /dev/sda /dev/sdb /dev/sdc`
- **zfs** : gestisce i volumi (file sytem)/snapshot
  con il comando zfs è possibile creare più volumi e attribuirgli proprietà diverse a secondo dell'utilizzo o del contenuto previsto .
  - `zfs create` crea il volume.
  - `zfs set|get` legge e assegna le proprietà al volume.

NB: uso termine volume non nel senso inteso normalmente con ZFS.

ZFS: Condivisone dello spazio

Lo spazio in un pool zfs è condiviso fra tutti i volumi presenti.

```
# zpool create zfsPoolName /dev/loop0
# zpool list
NAME          SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH
zfsPoolName  1016M   126K  1016M     0%  1.00x  ONLINE
#zfs create zfsPoolName/vol1
#zfs create zfsPoolName/vol2
#df -h
File system       Dim. Usati Dispon. Uso% Montato su
zfsPoolName       984M     0    984M   0% /zfsPoolName
zfsPoolName/vol1  984M     0    984M   0% /zfsPoolName/vol1
zfsPoolName/vol2  984M     0    984M   0% /zfsPoolName/vol2
```

ZFS: operazioni

- zfs create
- zfs snapshot “nome_volume”@”nome-snapshot”
  (visto come cartella sotto “path_volume”/.zfs/snapshot)
- zfs destroy (snapshot /volume)
- zfs clone: clona uno snapshot trasformandolo in un volume scrivibile.
- zfs rollback : ritorna allo snapshot più recente ,-r per uno snapshot più vecchio elimina tutti gli snapshot intermedi, -R elimina anche i cloni.
- zfs promote : Inverte le dipendenze fra il clone è lo snapshot originario.

ZFS: Se le cose vanno male !

- **`zpool status`** : mostra la “salute” del nostro device zfs. Inoltre mostra anche se ci sono operazione di riparazione attive.

  zpool scrub :controlla il file system e se necessario ricostruisce l'array di dischi. (on-line).

- Sostituzione di un disco in fault :
  `zpool replace “nomezpool” “old_dev” “new_dev”`
- E' possibili assegnare dei dischi di spare.
  `zpool  add “nomezpool” spare  “nome device”`

ZFS: Proprietà del “volume“

```
PROPERTY              VALUE                    SOURCE
 type                  filesystem               -
 creation              Thu May 16 15:29 2013    -
 used                  3.43G                    -
 available             12.4G                    -
 referenced            615M                     -
 compressratio         1.47x                    -
 logicalused           4.87G                    -
 logicalreferenced     1.15G                    -
 mountpoint            /samba                   local
 sharenfs              off                      default
 checksum              on                       default
 compression           gzip                     local
 atime                 off                      local
 dedup                 on                       local
 quota                 none                     default
 compression           gzip                     local
 snapdir               visible                  local
```

ZFS: Ereditarietà

Se creo un “sotto volume”, di default vengono ereditate le proprietà dal “data-set” padre .

```
pool/share  recordsize          128K                default
pool/share  mountpoint          /samba/share        inherited from pool
pool/share  sharenfs            off                 default
pool/share  checksum            on                  inherited from pool
pool/share  compression         lzjb                local
pool/share  atime               off                 inherited from pool
```

ZFS: Proprietà in sola lettura

- used : lo spazio utilizzati da questo data-set e da tutti i suoi figli.(snapshot/e “sotto-volumi”) . Uno snapshot inizialmente a used=0.
- referenced : somma di tutti blocchi referenziati da questo data-set. (per un clone inizialmente uguale al valore dello snapshot padre).
- compressratio : rapporto di compressione .
- dedupratio: rapporto di deduplica (zpool get all)
- logicalused ,logicalreferenced : spazio visto a livello applicativo. Quello che sarebbe stato occupato senza compressione e deduplica.

ZFS: proprietà modificabili

```
zfs set “proprietà”=”valore” “nome-volume”
```

- checksum=on | off | fletcher2 | fletcher4 | sha256. Con on prende il default (fletcher4 su FreeBSD). Verifica integrità dati e metadati.
- compression=on | off | lzjb | gzip | gzip-N | zle | lz4. Algoritmo di compressione.Default lzjb .zle (zero-length encoding) elimina gli zeri. In FreeBSD 9.2 aggiunto lz4.
- dedup=on | off | verify | sha256. Attiva la decuplica può imporre la verifica bit a bit.
- snapdir=visible | hidden. Visualizza la cartella .zfs nella quale è possibile trovare tutti gli snapshot come cartelle in sola lettura.

ZFS: molte quote

- Quota a limite a livello di volume .
  `zfs set quota='10G' 'nomevol'`
- Reservation : limita gli altri dataset al fine di garantire sufficiente spazio a questo volume .
  zfs set reservation='500M' 'nomevol'
- Quote per gruppi e utenti POSIX:

```
zfs set "groupquota@${NOME_GROUPPO}=${VAL_NUOVA_QUOTA}" "${ZFS_VOL_NAME}"

zfs groupspace "${ZFS_VOL_NAME}"

zfs userspace "${ZFS_VOL_NAME}"
```

ZFS: ZIL

- ZFS Intent Log è un log utilizzato per le scritture sincrone. ZFS usa una cache in memoria per le scritture . Normalmente le scritture sono raggruppate in gruppi e rese persistenti ogni 30 secondi. Le scritture sincrone (Database,NFS;Samba) devono però essere subito scritte su disco. Le scritture di tipo sincrono sono prima scritte nello ZIL al fine di :
  - garantire la coerenza dei nuovi dati in caso di fault (scrittura parziale).
  - per ottimizzare le scritture dei blocchi su disco (minimizzare la “rottura” dei blocchi già persistenti tramite offset).
  - tornare il più rapidamente possibile al chiamante.
- E' possibile disabilitare lo ZIL (sconsigliato!!) o trattare tutte le scritture come sincrone (sicurezza a scapito della velocità).

```
zfs set  sync=standard | always | disabled
```

ZFS: ZIL-CACHE

Con il comando zpool si può dedicare un device allo ZIL . Altamente consigliabile aggiungere due device in mirror. (magari SSD).

```
zpool create poolzfs raidz /dev/sda /dev/sdb /dev/sdc log mirror /dev/sdd
dev/sde
```

ZFS:cache L2

- E' possibile assegnare uno o più dischi veloci,(SSD) come cache di secondo livello. (Non serve che siano ridondati)

  zpool add poolzfsname cache /dev/sdh /dev/sdg

- Si può configurare cosa mettere in cache per volume

```
zfs set  secondarycache=all | none | metadata  nomevol
```

- OT: bcache: cache ssd tramite device-mapper quindi indipendente dal filesystem. (In mainline dal 3.10)
  http://bcache.evilpiepirate.org/

ZFS: NFS-SAMBA

Si possono configurare le condivisioni nfs e samba direttamente da zfs. Per esempio su FreeBSD si può usare la seguente sintassi.

```
zfs set sharenfs='-ro -network 192.168.0/24'
```

I servizi sono erogati dai consueti demoni, quindi è soltanto un frontend di gestione.

ZFS: SAMBA

shadow_copy2 :Modulo samba che permette il ripristino di un file salvato in uno snapshot direttamente dal client windows.

Configurazione di smb.conf:

```
vfs objects = shadow_copy2
shadow:format = %Y-%m-%d_%H.%M.%S--8d
shadow:sort = desc
shadow:snapdir = /samba/samba_t/.zfs/snapshot
shadow:basedir = /samba/samba_t
shadow:localtime = yes
```

ZFS :send recive

Con i comandi zfs send /zfs recive è possibile replicare un volume zfs ,realizzando cosi un sistema master slave con un semplice script nel cron.

```
/sbin/zfs send -i  "${LAST_SNAPSHOT_ON_SLAVE}"-R -D "${LAST_SNAPSHOT_ON_MASTER}"\
       | /usr/bin/ssh -c "${SSH_CRIPTO_ALG}"  "${HOST_ZFS_SLAVE}" /sbin/zfs receive -v -F \
 "${ZFS_VOL_NAME}" &>"${OUTPUTDIR}/zfs_send_recive.log"
CODICE_ERRORE_SEND="$?"
/usr/bin/logger -t "${LOG_TAG}" -f "${OUTPUTDIR}/zfs_send_recive.log"
if [[ $CODICE_ERRORE_SEND -ne 0 ]];
then
  gestisciErrore  "Errore  IN ZFS SEND RECIVE CODICE ERRORE  ${CODICE_ERRORE_SEND}  " 60
  exit 60
fi
```

Btrfs: B-Tree filesystem

- Dimensione massima : 2^64 byte circa 18 milioni TB .
- Grub2 supporta btrfs .
- Supporta `RAID0, RAID1, RAID5, RAID6  RAID10`
- Usa una foresta di b-tree:
  - Fstree - :inode indicizzati sia per path sia per ordine di creazione .Esiste un filesysytem tree per ogni volume)
  - Checksum tree : chechsum per ogni extend
  - Extend tree : item con cotattore di reference e offset .
  - Device tree : mappatura fisica di un extend attribuendolo ad un chunk. (singleton)
  - Chunk tree :distribuzione dei Chunk sui vari dischi. (singleton)
  - Root tree: contiene gli altri alberi.
  - Log tree : scritture sincrone.
  - Relocation trees :usato nella deframentazione e ricostruzione dell'array .(singleton)

Btrfs :benchmark

- Nonostante sia un filesystem “in erba” in molti benchmark ha risultati paragonabili ad ext4 .

  Riferimenti:

  https://oss.oracle.com/projects/btrfs/dist/documentation/benchmark.html

  http://www.phoronix.com/scan.php?page=article&item=linux_310fs_fourway

Btrfs :creare il filesystem

- E' possibile impostare una politica diversa per dati e metadati.

```
mkfs.btrfs  -m (politica-metadati) -d (politica-dati) /dev/sda /dev/sdb …
```

Configurazioni possibili	
Numero dischi	politiche
1	dup, single
2	raid0, raid1, raid5
3	raid0, raid1, raid5, raid6
4	raid0, raid1, raid5, raid10

- Visualizzazione configurazione dell'array di dischi:

```
btrfs filesystem show /dev/sda
btrfs device stats /dev/sda
btrfs filesystem  df “path/mountpoint”
```

Btrfs: solid state drive (SSD)

- Ottimizzazioni per SSD attivabili tramite opzione di mount **-o ssd** per abilitare il TRIM aggiungere `-o discard` (disabilitato di default per problemi performance con alcuni device) .
- Con un solo disco di default vengono duplicati i metadati su blocchi distanti fra loro (-m dup). La duplicazione è disattivata nel caso di dischi SSD. Questo perché gestiscano internamente la reale disposizione dei blocchi al fine di distribuire equamente le scritture su tutta la superficie (wear leveling) .

  (Per identificare ssd su linux cat /sys/block/sdX/queue/rotational)

Btrfs: volume di default

- Ad ogni volume creato viene attribuito un numero identificativo. Tramite questo id è possibile scegliere il volume montato di default.
  - **btrfs subvolume create /mnt/data/sub1** (crea volume con nome sub1 sotto /mnt/data)
  - **`btrfs subvolume list /mnt/data`** (mostra lista volumi con id)
  - **btrfs subvolume set-default id-num /mnt/data** (setta volume di default)
  - Quale volume montare può essere anche indicato al mount con **-o subvolid=**
  - E' possibile visualizzare altri id/proprietà del volume tramite **`btrfs subvolume show /mnt/data/sub1`**

Btrfs: snapshot

- Crea di default snapshot scrivibili in un qualsiasi percorso del volume.
  ```
  btrfs subvolume snapshot “nomeVolume”  “path/nomesnapshot”
  ```
- con -r crea snapshost in sola lettura.
  ```
  volume snapshot -r “nomeVolume”  “path/nomesnapshot”
  ```
- Per eliminare uno snapshot
  ```
  btrfs  subvolume delete “path/nomesnapshot”
  ```

Btrfs: Upgrade tranquilli !!

- Interessante è l'integrazione con gestore pacchetti .Per esempio è possibile creare uno snapshot prima di aggiornare la nostra distribuzione. In caso di problemi sarà quindi possibile ripristinare la vecchia versione per esempio passando a GRUB l'id del vecchio volume.
- Esempio di integrazione : apt-btrfs-snapshot https://launchpad.net/apt-btrfs-snapshot

Btrfs : ridimensionamento

- E' possibile ridimensionare online

```
btrfs filesystem resize -100G /path
```

- Per assegnare tutto lo spazio disponibile.

```
btrfs filesystem resize max /path
```

Btrfs: sostituzione disco.

- Sostituzione online di un disco rotto. Il parametro -r indica di usare le informazioni di parità RAID.
  `btrfs replace start -r “/dev/old” “/dev/new“ “/path/mountpoint”`
- **`btrfs replace status “/path/mountpoint”`**
  visualizza i progressi dell'operazione.

Btrfs: deframentazione

- Deframentazione online di file e cartelle
- `filesystem defragment` *“nomefile”*
- `filesystem defragment` *“nomedir”*

con **`-c zlib`** oppure **`-c lzo`** comprime i file mentre vengono deframmentati .

Btrf :quota a livello di volume

- E' necessario creare un gruppo specifico (qgroup) con nome **0/*idvolume*** .
  - **`btrfs quota enable /path/mountpoint`**
  - **`btrfs quota rescan /path/mountpoint`** (crea i vari qgroup)
- Posso assegnare un limite ad ogni volume
  - **`btrfs qgroup limit 200G /path/subvol`**
  - **`btrfs qgroup show /path/mountpoint`**

Btrfs: dedump

- Implementata la deduplica batch migliorata nel 3.12 tramite l'aggiunta di una syscall specifica (sperimentale).
- Programma in Python: bedup https://github.com/g2p/bedup/
- Esiste patch per abilitare opzione di mount -o dedup ma non ancora in mainline.

Btrfs: compressione

- La compressione viene abilitata al mount. Si può scegliere fra ZLIB(più efficiente), LZO (più veloce). La compressione è un attributo slavato per extend.
  `mount  -o compress=lzo`
- Si può far comprimere determinati file o cartelle impostandogli l' attributo **`c`** (compressible) con il comando chattr.
  `chattr -RV +c “pathNomeDir”` *(-R ricor*sivo V prolissso)

  Gli attributi dei file sono visualizzabili tramite **`lsattr.`**

Btrfs: send/recive

E' possibile replicare in maniera incrementale un volume. -p indica lo snapshot di partenza da cui costruire l'incrementale.

```
btrfs send -p /home/BACKUP_OLD /home/BACKUP-new | btrfs receive /backup/home
```

Btrfs :seed device

Un disco usato come seed non viene mai modificato ,ma costituisce una base riutilizzabile per altri dischi.

```
mkfs.btrfs /dev/sda
mount  /dev/sda /mnt/seed
tar xf contenuto.tar -C /mnt/data
umount /mnt/data
btrfstune -S 1 /dev/sda (imposto di tipo seed)
mount /dev/sda /mnt/data (viene read only )
btrfs device add /dev/sdb /mnt/data
mount -o remount,rw /mnt/data
cp cambiamento /mnt/data (modifica apportata solo a /dev/sdb)
```

Btrfs: snapshot di file

- Tramite coreutils recenti è possibile effettuare uno snapshot a livello di file e cartelle.

  `cp --reflink file1 file2` (snapshot a livello di file)

- In Samba 4.1 :si può utilizzare il modulo il btrfs. tutte le copie server-side sono effettuate sfruttando le potenzialità del COW

  In smb.conf :

```
vfs objects = btrfs
```

Conclusioni

- Btrfs risulta già molto performante grazie all'utilizzo dei btree ma è ancora troppo immaturo per un utilizzo server-side.
- Da migliorare :
  - Deduplica
  - Supporto quote gruppi/utenti
  - Gestione delle proprietà dei volumi
- Btrfs è il futuro: le nuove funzionalità offerte dal COW sono molto utili e porteranno sicuramente ad una vittoria di btrfs sui filesystem basati su sovrascrittura.

Fonti

- man
- http://zfsonlinux.org/
- https://wiki.freebsd.org/ZFSTuningGuide
- https://wiki.freebsd.org/ZFSQuickStartGuide
- http://en.wikipedia.org/wiki/ZFS
- IBM Research Report BTRFS: The Linux B-tree Filesystem
- http://en.wikipedia.org/wiki/Btrfs
- https://btrfs.wiki.kernel.org/
- http://www.funtoo.org/wiki/BTRFS_Fun
- cplipart http://tux.crystalxp.net