# ~~ZOL – ZFS On Linux~~
# OpenZFS

## Il miglior Filesystem per Linux non è per Linux

Sossi Andrej – andrej.fil@gmail.com


Casa del Popolo,
Sabato 28 ottobre 2023



# *file-system*

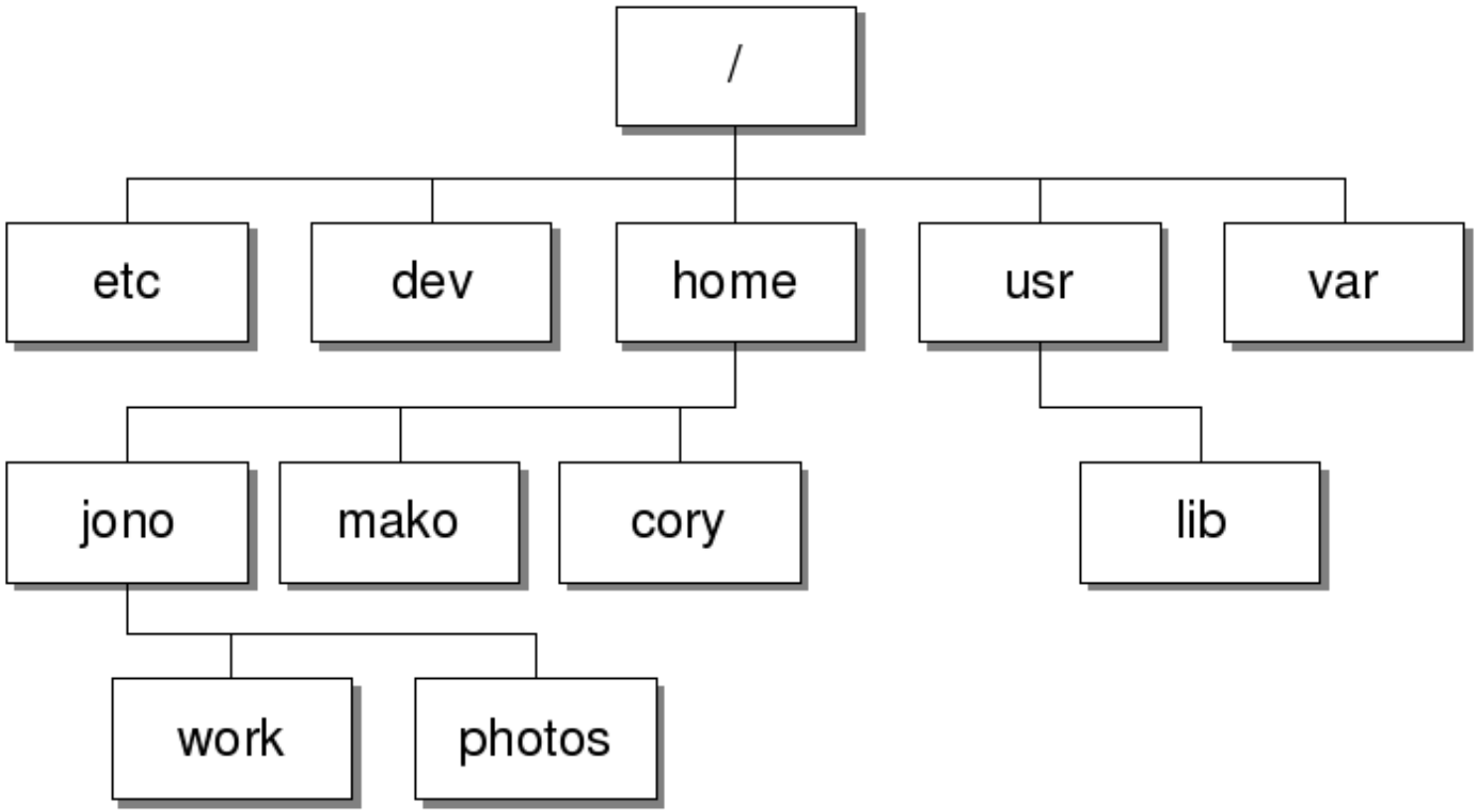
- Cos'è un file-system?
  (Dipende a cosa pensiamo)

  Molti utilizzano il termine file-system in maniera improrpia e alcuni senza sapere nemmeno le basi

# *file-system*

- Il file-system Linux (Unix) è una lista di file 'raggruppati' per cartelle.
- Le cartelle sono in relazione gerarchica senza possibilità di relazioni circolari
- Le cartelle e i file possono avere nomi arbitrari
- Alcuni nomi sono definiti dallo standard POSIX e/o System V

# file-system



# *file-system*

- Le cartelle sono cartelle tutte uguali
- I file sono di tipo diversi
  file regolari
  link
  device a blocchi
  device a caratteri
  fifo
  socket
  ...

# file-system

File Edit View Terminal Tabs Help

```
bash-4.4$ ls -al /dev/
total 4
drwxr-xr-x  19 root root         6060 Oct 16 18:48 .
drwxr-xr-x  25 root root         4096 Oct  4 00:00 ..
crw-------   1 root root      10, 175 Oct 16 15:24 agpgart
crw-------   1 root root      10, 235 Oct 16 17:24 autofs
drwxr-xr-x   2 root root          760 Oct 16 18:48 block
drwxr-xr-x   2 root root           80 Oct 16 18:48 bsg
crw-rw----   1 root disk      10, 234 Oct 16 15:24 btrfs-control
drwxr-xr-x   3 root root           60 Oct 16 17:24 bus
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdr -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdr0 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdr6 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdrom -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdrom0 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdrom6 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdrw -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdrw0 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdrw6 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdwriter -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdwriter0 -> sr0
lrwxrwxrwx   1 root root            3 Oct 16 15:24 cdwriter6 -> sr0
drwxr-xr-x   2 root root         5060 Oct 16 18:48 char
crw-------   1 root root       5,    1 Oct 16 15:25 console
```

# *file-system*

- Dove si trovano 'fisicamente' questi file? Dipende: dischi, servizi di rete (NFS, SAMBA, fusefs), RAM, chievette USB, CD, DVD, floppy o **da nessuna parte**.
- In Linux (Unix) il file-system gestito dal sistema operativo è uno.
- Quindi come rappresentare tutte queste 'sorgenti' di file in un unico filesystem?

# file-system

File Edit View Terminal Tabs Help

```
crw-rw----   1 root tty        7,   6 Oct 16 15:25 vcs6
crw-rw----   1 root tty        7,   7 Oct 16 15:25 vcs7
crw-rw----   1 root tty        7, 128 Oct 16 15:24 vcsa
crw-rw----   1 root tty        7, 129 Oct 16 15:24 vcsa1
crw-rw----   1 root tty        7, 130 Oct 16 15:25 vcsa2
crw-rw----   1 root tty        7, 131 Oct 16 15:25 vcsa3
crw-rw----   1 root tty        7, 132 Oct 16 15:25 vcsa4
crw-rw----   1 root tty        7, 133 Oct 16 15:25 vcsa5
crw-rw----   1 root tty        7, 134 Oct 16 15:25 vcsa6
crw-rw----   1 root tty        7, 135 Oct 16 15:25 vcsa7
drwxr-xr-x   2 root root            60 Oct 16 17:24 vfio
crw-------   1 root root       10,  63 Oct 16 15:24 vga_arbiter
crw-------   1 root root       10, 137 Oct 16 17:24 vhci
crw-------   1 root root       10, 238 Oct 16 17:24 vhost-net
crw-rw-rw-   1 root root        1,   5 Oct 16 15:24 zero
bash-4.4$
bash-4.4$ mount
/dev/mapper/cryptvg-root on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda6 on /boot type ext4 (rw)
gvfsd-fuse on /home/fil/.gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,user=fil)
bash-4.4$
```

# *file-system*

- Le 'sorgenti dei file' vengono montati sul file-system
- Es.:
  la / è montata da un disco tipo **ext4** che contiene (tra le altre) la cartella /proc
  la /proc è montata da un tipo **proc**
- I file montati da **proc** sono creati direttamente dal kernel (Linux) è rappresentano i processi che girano sul system

# *file-system*

- Ma i 'nostri' file dove e come vengono salvati?
- I 'nostri' file vengono salvati sul disco.
- Poi il disco viene montato su / o su /home o su /opt o … dove volete
- Per scrivere i bit dei file su disco esistono più strategie diverse.
- Queste 'strategie' vengono chiamate tipi di file-system.

# *file-system*

- Quanti filesystem conosciamo?
- (UFS), ext2, ext3, ext4, XFS, JFS, ReiserFS, btrfs, (ZFS, HFS, FAT16, FAT32, exFat, NTFS, ReFS, HPFS…), fuse, etc.
- Caratteristiche molto diverse.

# UFS

- UFS – Unix File System
- È un derivaro dal Berkley File System
- Attualmente è stato migliorato aggiungendo funzionalità (come il journaling etc.)
- Usato in tutti i derivati BSD (FreeBSD, Net-BSD, OpenBSD, Solaris,…)
- Supportato anche da Linux e MacOS

# *ext**

- ext – extended file system
- È il primo Filesystem creato per Linux (1992) per superare i limiti del filesystem di Minix (grandezza delle partizioni, lunghezza dei nomi dei file,...)
- La struttura di ext è basato sulla strutura di UFS
- ext2 – nasce come successore di ext per superare i suoi difetti.
  ext2 subi vari aggiornamenti (ACL, etc.)

# *ext**

- ext3 – è un ulteriore aggiornamente di ext2.
- Aumentate le dimensioni massime, aggiunge in Journaling, possibilità di aumentare la dimensione della partizione a caldo, migliora l'indicizzazione delle cartelle di grandi dimensioni.
- Si può convertire un ext2 in ext3 senza perdita di dati.

# ext*

- ext4 – nasce (2006-2008) come successore di ext3
- Continua ad essere possibile convertire un etx3 in ext4
- Aggiunge precchie nuove funzionalità (troppe e troppo complesse da spiegarle… Guardate in wikipedia)

# Altri file-system per Linux

- In parallelo alla serie di ext* vengono sviluppati altri file system per il kernel Linux
- xiafs – nato in parallelo con ext, oggi considerato obsoleto e non più mantenuto (cancellato dal kernel 2.1.21)
- RaiserFS – Ottimi filesystem, ne esistono 4 versioni (ReiserFS4 non è mai entrato nel kernel ufficiale)
- BtrFS – viene creato per avere in Linux le funzionalità fornite da ZFS (segue...)






# *Altri file-system*

- Oltre i file system sviluppati specificata-mente, Linux supporta molti altri tipi di file system
- Spesso il supporto viene creato semplice-mente per compatibilità
- In altri casi il tipo di file system diventa un opzione da considerare per un utilizzo gior-naliero

# Altri file-system

- *FAT* - Viene sviluppato per MS-DOS poi aggiornato nel tempo. L'utilizzo necessario per compatibilità con Windows e altri apparati (macchine fotografiche etc.)
- NTFS – nato per Windows in linux c'è un supporto parziale
- HFS HFS+ – nato per MacOS X in Linux c'è un supporto parziale
- XFS – nato per IRIX da SGI. Funziona otti-manete in Linux
- Altri...

# Btrfs

- Oracle corp. annunciz Btrfs nel 2007
- Il nome completo è: B-tree FS o "Butter FS" o "Better FS"
- Prima apparizione in Linux 2.6.29-rc1 2008 (v.0.20)
- Usato in prodizione Linux 3.10 2012 (v.1.0)
- Nasce per essere un alternativa a ZFS
- Dimensione massima: 16 EB
- Numero massimo di file $2^{64}$

# Btrfs

- Massimo nome di un file: 255 caratteri
- Supporta tutti i caratteri (eccetto / e NULL)
- Algoritmi di compressione: zllib, LZO, Snappy
- Tecnica di scrittira: COW (copy on write)
- Supporto snapshot
- Estendibile su più dischi
- Defrag automatico
- RAID0, RAID1 e RAID10
- Checksum integrato

# ZFS

- ZFS – Zettabyte File System (anche se oggi non è più da considerare un acronimo)
- Nasce all'interno della Sun Microsystems
- L'idea di fondo: creare un FileSystem che potesse essere utilizzato come le altre risorse del PC. Serve più RAM, aggiungo più RAM, perché se aggiungo più disco non posso usarlo, ma devo eseguire operazioni per poter utilizzare l'ulteriore spazio?

# ZFS

- Con ZFS basta un comando per estendere il file system su nuovi device e lo spazio e subito utilizzabile, senza dover “formttare” o “modificare” in qualche modo il file system originale e principalmente si esegue con il file system “on-line”, cioè senza smontarlo.

# ZFS

- Progetto della Sun Microsystem del 2001
- Dopo parecchi test viene annunciato ufficialmente il 14 settembre 2004
- Fine 2005 il codice sorgente viene inserito nella main trunk di OpenSolaris
- Metà del 2006 viene reso disponibile in Solaris 10
- Per 5 anni vengono aggiunte nuove funzionalità
- Licenza CCDL (Common Development and Distribution License)






# ZFS

- 2010 anno del “cambiamento”
- Sun fallisce e viene comprata da Oracle
- Prima di fallire Sun pubbica tutto il codice di Solaris ripulito da licenze proprietarie
- Illumos iventa il progetto della Community per il mantenimento di OpenSolaris
- Alcune tecnologie vengono prese da Free-BSD e rese note a un pubblico più ampio
- OpenZFS viene portato su FreeBSD

# ZFS

- Un primo tentativo di supportare ZFS in Linux mediante FUSE
- Poi inizia il progetto ZOL (ZFS On Linux) per avere un supporto nel kernel (modulo)
- La ricetta del disastro è pronta… quattro progetti: Oracle ZFS, OpenZFS mantenuto da Oracle, OpenZFS mantenuto dalla community FreeBSD, ZOL mantenuto dalla Community Linux, altri...

# ZFS

- Ognuno manteneva ZFS per conto proprio
- Ognuno aggiungeva nuove funzionalità
- I vari tentativi di allineamento tra i progetti risultavano parecchio faticosi e poco efficaci
- Fortunatamente le varie comunità decidono di cominciare a parlare per evitare sprechi e sovrapposizioni

# ZFS

- Si decide che ZOL sarà a capo della gestione della repository preincipale dei OpenZFS
- Ognuno dovrà proporre e pubblicare le modifiche nella repository centrale
- Ognuno sarà libero di gestire e mantenere in autonomia le “aggiunte” allineate alla repository centrale
- Oracle prosegue sulla propria strada

# ZFS

- Per evitare problemi di compatibilità tra la versione dell community e quella di Oracle la nuova versione comunitaria parte da 5000
- Distro evitato. Lo sviluppo adesso prosegue in maniera coordinata e omogenea tra le varie comunità (Linux, FreeBSD, Illumos,...)

# ZFS

- OpenSolaris e FreeBSD sono basati su UNIX SIXTH EDITION V6 (1976) e del successivo 4.1BSD (1981)
- Linux si basa(va) invece su UNIX SYSTEM V (1983+)
- Le due strade si sono divise nel lontano 1976
- Ci sono molte cose in comune, ma anche molte grosse differenze

# ZFS

- Non è quindi pensabile di prendere ZFS e inserirlo derettamente nel kernel Linux a causa delle differenze architetturali
- Nasce così ZOL che è di fatto uno strato software che espone al sisitema operativo delle API (funzioni) che Linux si attende per un filesysem e l'implementazione mappa sulle chiamate di ZFS

# ZFS

- Oggi ZOL non esiste più, nel senso stretto, visto il lavoro per portare tutto lo sviluppo a fattor comune sotto il nome di OpenZFS

# ZFS

- Oltre ai problemi tecnici c’è anche uno legale
- FreeBSD viene distribuito con la licenza BSD
- Linux viene distribuito con licenza GPL
- Il sorgente di ZFS è licenziato con la CDDL
- La CDDL non è distribuibile ne con la BSD ne con la GPL
- ZFS quindi rimane esterno al kernel, come modulo separato

# ZFS

- Essendo all’esterno del kernel c’è la necessità di compilarlo a parte
- Quindi a ogni aggiornamento del kernel bisogna aggiornare o ricompilare il modulo, (così come succede con i driver proprietari Nvidia)

# ZFS

- Caratteristiche:
- è in filesystem pensato per il futuro
- Basato su 128 bit ha capacità nettamente maggiori dei filesystem più moderni basati su 64 bit.

# ZFS

- Caratteristiche (alcuni numeri):
- Dimensione massima
  - file: 16 exabyte
  - pool: $3x10^{23}$ petabyte
  - nr. file (per directory): $2^{56}$ ($2^{48}$)
  - nr. Dischi fisici nel pool: $2^{64}$

# ZFS

- Cratteristiche (tratto da wikipedia):
- Un utente che volesse creare mille file al secondo, impiegherebbe 9.000 anni a raggiungere il limite.
- [...]la meccanica quantistica impone alcuni limiti fondamentali[...]Un pool di storage a 128 bit completamente riempito[...]richiederebbe 136 miliardi di kg.

# ZFS

- Cratteristiche (tratto da wikipedia):
- Partendo dal dato di 136 miliardi di kg di storage completamente riempito si può calcolare l’energia necessaria per scrivere i dati. “Quindi, riempire uno storage a 128 bit dovrebbe richiedere più energia che bollire gli oceani.”

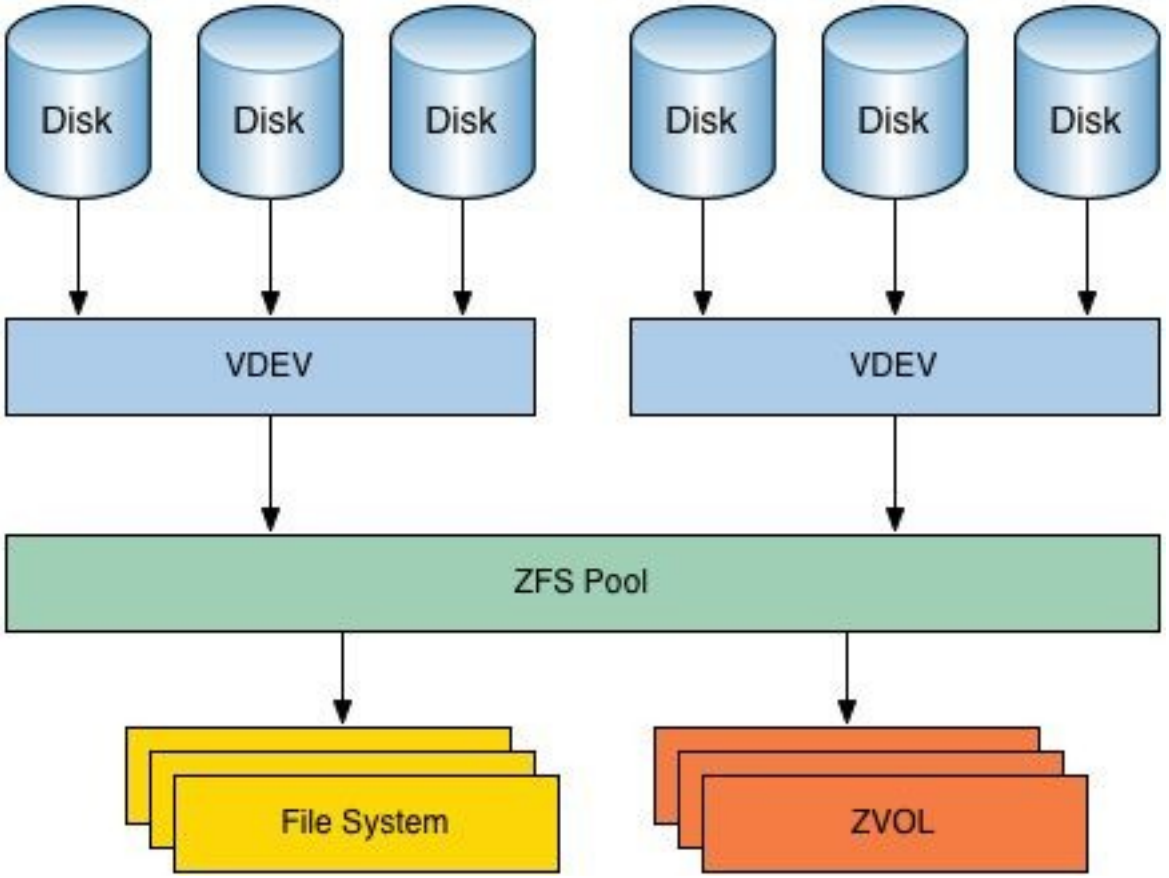
# ZFS

- ZFS come BtrFS non sono “solamente” dei file system, ma sono dei gestori lgici dei dischi, in altre parole posono “raggrup-pare” più dischi e mostrarli come singolo volume.
- Supporta vari tipi di RAID (vediamo dopo)
- Checksum sui dati e continua verifica in backgroud
- Compressione dei blocchi (più algoritmi)
- Deduplica dei blocchi
- Cifratura

# ZFS

- Copy on write (COW) transazionale
- Snapshot
- Espandibilità on line (non è supportata il rimpicciolimento)
- Estrema solidità in caso di situazioni poco carine (calo di corrente, crash del kernel, etc.)
- Estrema configurabilità per partizione

# ZFS








Lug Trieste

# RAID-Z

- RAID… ZFS ha un approcio un po’ diverso dal solito...
- RAID-Z ha 5 modalità
- RAID-Z in generale NON richiede dischi di misure uguali

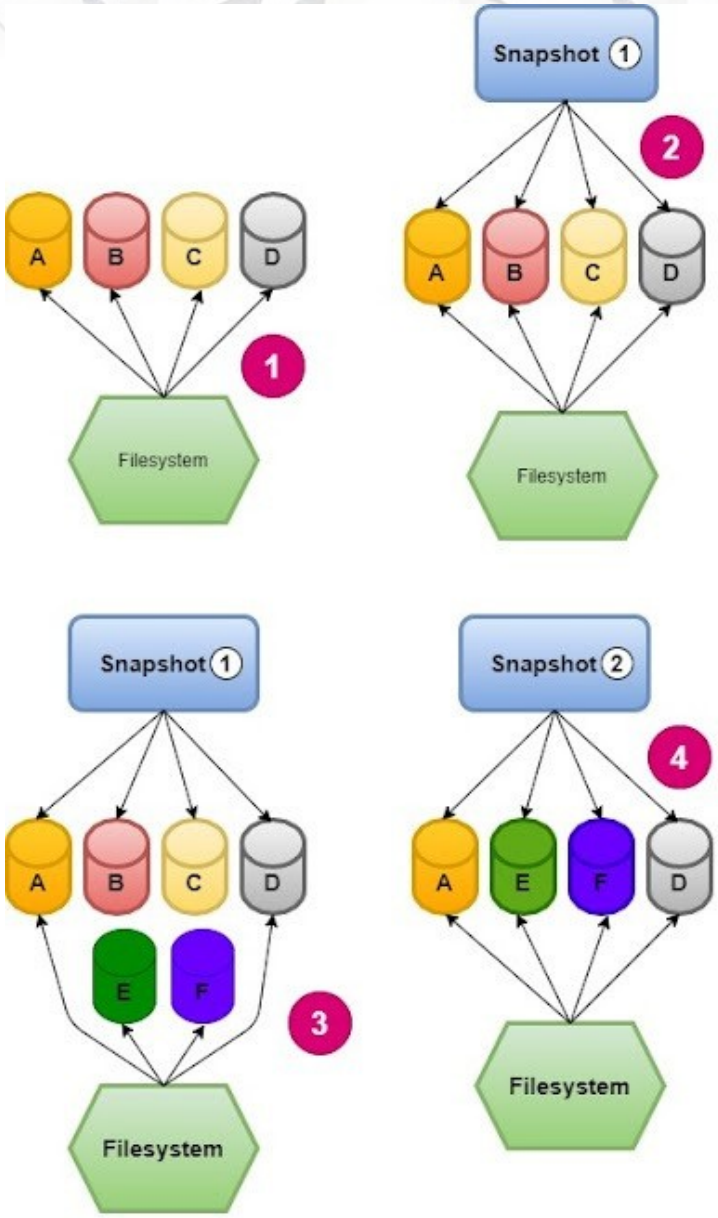
# RAID-Z

- Di default i dischi vengono concatenati, simile a RAID-0 “striping”
- RAID-Z1 simile a RAID-5 garantisce la possibilità di perdere uno dei dischi del pool senza compromettere i dati
- RAID-Z2 simile a RAID-6 garantisce la possibilità di perdere due dei dischi del pool senza compromettere i dati

# RAID-Z

- RAID-Z3 simile a RAID-6 garantisce la possibilità di perdere due dei dischi del pool senza compromettere i dati
- Il mirroring (come il RAID-1) – non richiede dischi uguali, ma si allinea a quello più piccolo

# SNAPSHOT








Lug Trieste

# SNAPSHOT

- Utili per fare esperimenti
- Utili per fare backup (le snapshot possono essere inviate ad un altro PC; vedi “zfs send” e “zfs receive”)
- Utili per clonare i dati senza occupare spazio...

# Benchmarks

- Riporto i dati che ho trovato documentati da “morrolinux”
- Non ho rieseguito i test su una mia macchina, ma i risultati combaciano con l’esperienza durante l’uso “giornaliero” sui server
- Il conftonto è stato fatto tra ext4, xfs, brtfs e zfs
- Il test sono stati svolti con Phoronix Test Suite

# Benchmarks

- Primo test: lettura randomica; block size di 4KB
- Ext4: 134 MB/s
- XFS: 131 MB/s
- Btrfs: 133 MB/s
- ZFS: 208 MB/s

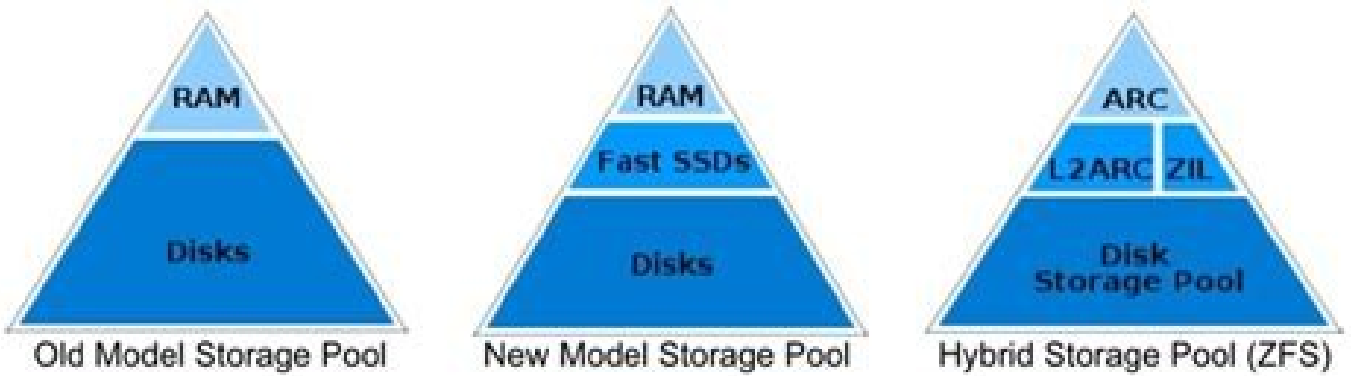
# Benchmarks

- Secondo test: lettura randomica; block size di 2MB
- Ext4: 514 MB/s
- XFS: 515 MB/s
- Btrfs: 508 MB/s
- ZFS: 3683 MB/s

- Cosa?!? Ragioniamo… Com fa ZFS ad avere numeri così buoni

# ZFS

- Semplice: cache, cache, cache




- L'SO utilizza normalmente la page cache per tutti i file system, ZFS ne utilizza una sua
- ARC – Adaptive Replacement Cache
- ZIL – ZFS Intent Log (noto anche come SLOG (Secondari Log device))

# Benchmarks

- Terzo test: scrittura randomica; block size di 4KB
- Ext4: 181 MB/s
- XFS: 180 MB/s
- Btrfs: 59,50 MB/s
- ZFS: 203,40 MB/s

# Benchmarks

- Quarto test: scrittura randomica; block size di 2MB
- Ext4: 418 MB/s
- XFS: 418 MB/s
- Btrfs: 92.8 MB/s
- ZFS: 5294 MB/s

- Cosa?!? Di nuovo?!?

# ZFS

- La “cache” funziona anche in scrittura
- Tutti i file system “accumulano” i dati da scrivere nella RAM e poi inviano al disco un blocco di dati cercando di ottimizzare i le operazioni dei dischi
- ZFS si permette di accumulare in RAM molto di più e ha strategie molto più aggressive e crea transazioni di scrittura più grandi

# ZFS

- Ma se ZFS mette in RAM, sono proprio sicuro di non perdere il dato?
- Tutti i test sono stati fatti con scritture asincrone, cioè il software dice al SO di scrivere.
- Alcuni software (basi di dati in primis) chiedono esplicitamente al sistema operativo che il dato sia scritto fisicamente su disco prima di confermare (scritture sincrone). E come si comporta ZFS?

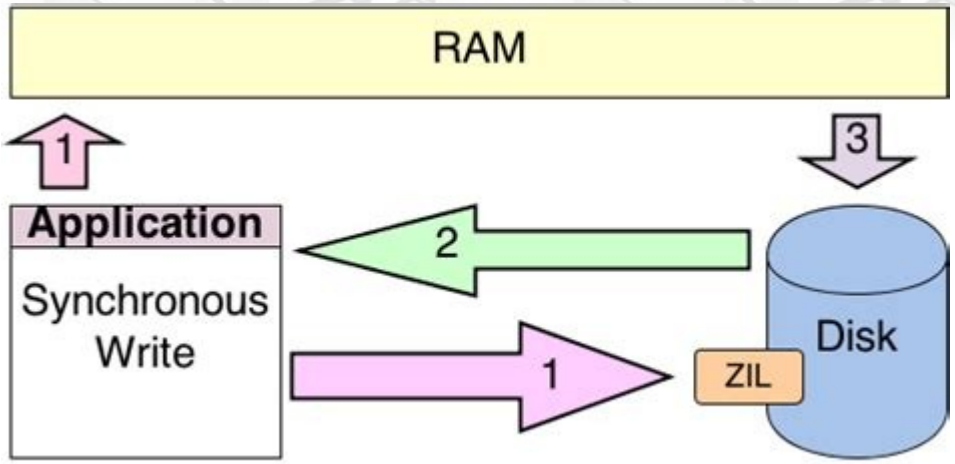





# Benchmarks

- Quarto test: scrittura randomica; block size di 2MB
- Ext4: 418 MB/s
- XFS: 418 MB/s
- Btrfs: 92.8 MB/s
- ZFS: 5294 MB/s
- ZFS-sync: 91 MB/s

# ZFS




- 1) Il dato viene scritto su ZIL
- 1) Il dato viene scritto in RAM
- 2) Viene data conferma all'applicativo
- 3) Raggruppate le scritture viene scritto il tutto sul disco (come di solito)
- 4) Viene liberata la ZIL





Lug Trieste

# ZFS

- La scritture sincrone in realtà si trasformano in più scritture.
- Se mettete la ZIL su un altro disco apposito (SSD) le performance migliorano notevol-mente

# Benchmarks

- Terzo test: scrittura randomica; block size di 4KB
- Ext4: 181 MB/s
- XFS: 180 MB/s
- Btrfs: 59,50 MB/s
- ZFS: 203,40 MB/s
- ZFS-sync: 1.994 MB/s

# Benchmarks

- Da precisare che non si è fatto delle scritture sincrone sugli altr file system, quindi è una comparazione non prettamente corretta
- ZFS ragiona internamente con blocchi di 128KB. Se forzo scrittue a 4KB ZFS continua a scrivere 128KB alla volta, quindi lo forzo a un sovralavoro
- È possibile configurare una partizione ZFS che ragioni a 4KB e le prestazioni migliorano, ma restani inferiori agli altiri

(((ho sentito PostgreSQL???)))

# Benchmarks

- Quinto test: lettura sequenziale; block size di 4KB
- Ext4: 112 MB/s
- XFS: 97,8 MB/s
- Btrfs: 139 MB/s
- ZFS: 1499 MB/s
- ZFS-sync: 1299 MB/s

# Benchmarks

- Quinto test: lettura sequenziale; block size di 4KB
- Ext4: 112 MB/s
- XFS: 97,8 MB/s
- Btrfs: 139 MB/s
- ZFS: 1499 MB/s
- ZFS-sync: 1299 MB/s

# Benchmarks

- Sesto test: lettura sequenziale; block size di 2MB
- Ext4: 138 MB/s
- XFS: 138 MB/s
- Btrfs: 136 MB/s
- ZFS: 3371 MB/s
- ZFS-sync: 3711 MB/s

# Benchmarks

- Settimo test: scrittura sequenziale; block size di 4KB
- Ext4: 256 MB/s
- XFS: 252 MB/s
- Btrfs: 90.4 MB/s
- ZFS: 168.6 MB/s
- ZFS-sync: 4 MB/s

# Benchmarks

- Settimo test: scrittura sequenziale; block size di 2MB
- Ext4: 418 MB/s
- XFS: 418 MB/s
- Btrfs: 168 MB/s
- ZFS: 5280 MB/s
- ZFS-sync: 101.4 MB/s

# ZFS

- ZFS è migliore nella maggior parte dei casi.
- Ci sono dei casi d’uso non ideali
- Se ve la andate a cercare (((ho sentito di nuovo PostgreSQL))) c’è la possibilità di migliorare la situazione on un tunning
- Modificare i parametri di ZFS non è difficile, ma farlo bene non è banale; esistono delle guide e dei casi d’uso specifici ben documentati

# ZFS

- Oltre le performance bisogan tenere conto delle funzionalità
- Avere delle funzionalità in più per dei rallentamenti in alcuni casi merita (parere personale)
- Ext-4 è sicuramente la scelta perfett se non volete pensare a nulla
- ZFS è ottimo, ma vi richiederà un attimo di attenzione su alcune configurazioni

# ZFS-limiti

- RAM – l'uso normale del file system consuma molta RAM, a causa della cache (si calcola 1GB per 1TB di storage)
- Write amplification – sopratuto su dischi SSD potrebbe essere un problema serio
- Attenti a non superare 80% della capienza (oggi si dice 94%) sopratutto se c'è tanta frammentazione

# ZFS-limiti

- La licenza non GPL-compatibile vi richiederà dei lavori aggiuntivi durante gli aggiornamenti o dovere usare una distribuzione che dichiara appositamente di supportare ZFS
- Torvald Linus sconsiglia vivamente l'utilizzo di OpenZFS a causa della licenza

# ZFS-limiti

- Se volete sfruttare al meglio le funzionalità dovete ragionare bene sul Hardware: dischi per lo storage diversi da quelli per lo SLOG (ZIL) – Tipicamente un SSD molto veloce per lo ZIL da rimpiazzare in un paio di anni (non dimenticatevi della RAM).

# Conclusioni

- ZFS è letteralmente una rioluzione, ma come per tutte le rivoluzioni necessita di adatta-mento e consapevolezza
- Bisogna studiare e provare per poter sfuttare al massimo nuove opportunità fornite dal nuovo approccio

# *Licenza d'uso di questo documento*








Lug Trieste