Istituto Nazionale di Fisica Nucleare

Corso RedHat per sistemisti INFN

RHEL/SL/CentOS 7

Novita' storage RH7

- Nuovo ISCSI layer
- XFS come default file-system
- System Storage Manager
- VDO

iSCSI

- Protocollo IP che consente di accedere a storage condiviso via rete IP simulando un l'accesso SCSI livello di block device
- I messaggi definiti dallo standard SCSI (Command Descriptor Blocks) sono incapsulati in pacchetti TCP ed inviati sulla SAN
- Nella versione 7 dei sistemi basati su RedHat Enterprise, l'implementazione e' basata su due prodotti opensource:
  - **Linux-IO** (sviluppato da Datera) lato server
  - **Open-iSCSI** lato client

iSCSI

- L'invio dei dati avviene in chiaro, non c'e' garanzia di confidenzialita' e integrita'
- L'accesso agli storage server puo' essere protetto da username e password
  - CHAP: cifratura debole delle credenziali
- E' fortemente consigliato confinare il traffico in una o piu' VLAN
- L'accesso allo storage da parte del client puo' essere mutuato da hardware dedicato (HBA)
  - Viene meno la necessita' dell'implementazione software

Elementi di iSCSI

Initiator	L'iSCSI client, implementato spesso via software ma disponibile anche come device (**HBA**). Deve essere associato ad un **IQN**
Target	Espone piu' device fisici (backend) attraverso una o piu' interfacce di rete del server. Ad ogni backend e' associato un id numerico (**LUN**). Ogni target e' contrassegnato da un **IQN**. Un server puo' fornire piu' target.
ACL	Il LUN mapping e' eseguito attraverso regole che negano o consentono l'accesso di un **IQN initiator** ad un target
Discovery	Sessione iSCSI che consiste nella richiesta da parte dell'initiator delle risorse esposte da un server

Elementi di iSCSI

IQN	L'iSCSI Quilified Name e' I'identificativo del target o dell'initiator. Ha il seguente formato: `iqn.2018-04.com.example[:nome_opzionale]`
Login	L'initiator, per poter accedere ad un target, deve eseguire un login (che puo' essere anche privo di credenziali)
LUN	Un id numerico (Logical Unit Number) associato al backend che viene esportato dal target. Ad un target possono essere associate piu' LUN.

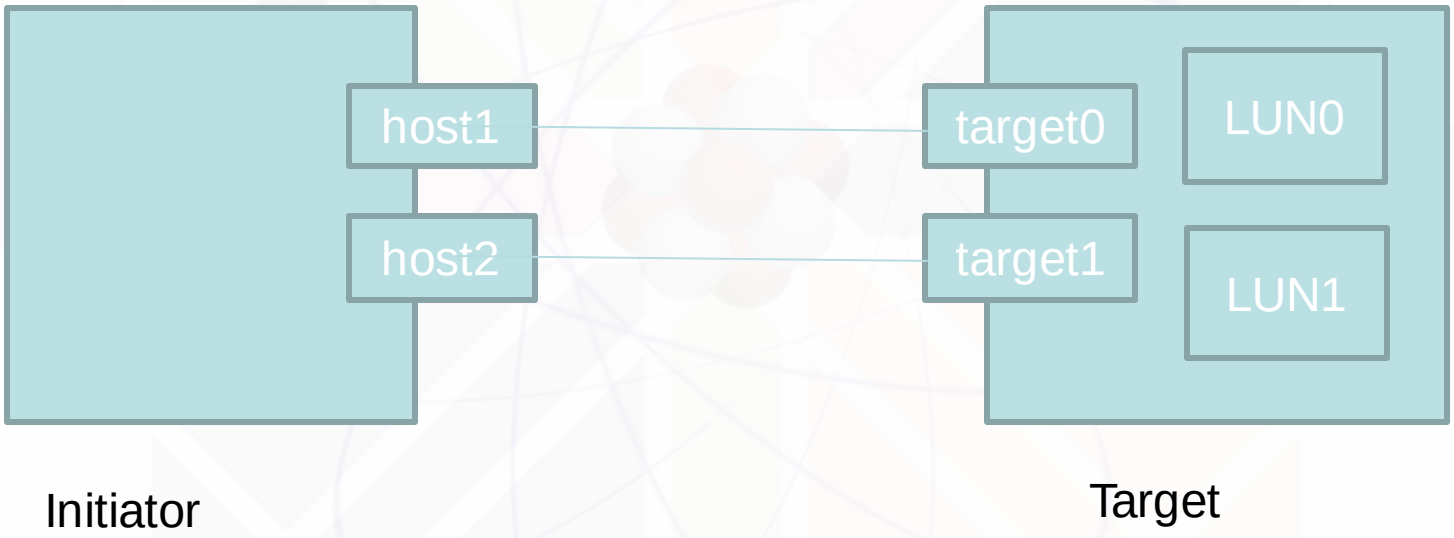
Elementi di iSCSI

Node	Un qualsiasi initiator o target identificato da un IQN
Portal	Una coppia **indirizzo IP/porta TCP** utilizzata sul server per la connessione dall'initiator al target. Un target puo' includere diversi portal
TPG	Target Portal Group. E' l'insieme dei portal associati ad un target, a cui sono associate un certo numero di ACL e di LUN
Session	Esistono due tipi di sessione iSCSI, la **operational session**, cioe' il normale scambio di CDB che viene sempre fatto partire dall'initiator ed inizia con il **login**, e la **discovery session**, cioe' la sessione di richiesta delle risorse messe a disposizione di un server

Target, LUN e indirizzi SCSI

- Nella terminologia SCSI un **target** corrisponde ad una singola unita' fisica che espone un certo numero di unita' logiche (**LUN**).
  - Ad esempio, un controller di uno storage enclosure e' un target.
- Un target puo' essere acceduto da interfacce differenti della macchina client (**host**).
- Nella macchina "client" (quella che implementa l'initiator), il singolo block device (es. /dev/sda) e' associato un **indirizzo SCSI**
  - vettore di 4 parametri: **host**, **bus**, **target** e **LUN**
- Esempio: un client esegue l'iSCSI login ad un server su cui sono definiti:
  - Un target
  - Un backstore (associato ad una LUN, ad esempio "1")
  - Due portal (ad esempio su due interfacce di rete differenti)
- Il client creera' due device (chiamiamoli /dev/sdb e /dev/sdc) associati agli indirizzi SCSI [1:0:0:1] e [2:0:0:1], che corrispondono allo stesso volume acceduto da due cammini differenti.

- 4 device scsi sull'initiator
  - [1:0:0:0] [1:0:0:1] [2:0:1:0] [2:0:1:1]



Il target iSCSI su CentOS 7

- Il pacchetto Linux-IO per CentOS7 e' `targetcli`
  - `yum install targetcli`
- Il servizio systemd e' target
  - `systemctl enable target`
  - `systemctl start target`
- Gli elementi da definire per creare un target iSCSI sono: un **IQN**, uno o piu' **backstore**, un **TPG**, uno o piu' **portal**, una o piu' **ACL**, una o piu' **LUN**
- L'interfaccia e' intuitiva e la configurazione si presenta come un directory tree. Molti comandi sono mutuati dalla shell di login (cd, pwd, ls, set)

I backstore

- E' l'unita' locale al server che fisicamente contiene i dati
- Il target puo' esportare diversi tipi di backstore
- **Fileio**: viene utilizzato un file sul filesystem della macchina. Il meccanismo e' simile a quello del loop device
- **Block**: un device di tipo **b** (block) definito sul server (un disco, una partizione, un volume LVM,….)
- **Pscsi**: passthrough verso un device SCSI. Comunica i CDB direttamente al device sottostante. Non e' generalmente utilizzato
- **Ramdisk**: Crea un ramdisk sulla memoria del server. Lo storage non e' persistente e viene cancellato al reboot

Esempio: come creare un backstore

```
cd /backstores/
/backstores> block/ create block1 /dev/vdc
Created block storage object block1 using /dev/vdc.
/backstores> fileio/ create file1 /root/disk01.img size=1G
Created fileio file1 with size 536870912
/backstores> ls
/backstores> ls
o- backstores ..................................................... [...]
  o- block ......................... [Storage Objects: 1]
  | o- block1 ..... [/dev/vdc (1.0GiB) write-thru deactivated]
  |   o- alua ........................... [ALUA Groups: 1]
  |     o- default_tg_pt_gp .... [ALUA state: Active/optimized]
  o- fileio ........................ [Storage Objects: 1]
  | o- file1  [/root/disk01.img (512.0MiB) write-back deactivated]
  |   o- alua ........................... [ALUA Groups: 1]
  |     o- default_tg_pt_gp .... [ALUA state: Active/optimized]
  o- pscsi ......................... [Storage Objects: 0]
  o- ramdisk ....................... [Storage Objects: 0]
```

Creare il target

- Al comando create e' possibile indicare un IQN secondo lo schema dell'RFC.
  - `iqn.2018-04.com.example:iscsidisk01`
  - Se non viene indicato una valore per la parte opzionale `targetcli` ne genera uno
- Un TPG viene automaticamente creato e associato al target. Nel TPG devono essere configurati gli ACL, le LUN ed i Portals

Esempio: creazione del target

```
/backstores> cd /iscsi/
/iscsi> create iqn.2018-04.com.example:iscsidisk01
Created target iqn.2018-04.com.example:iscsidisk01.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
/iscsi> ls
o- iscsi ..................................... [Targets: 1]
  o- iqn.2018-04.com.example:iscsidisk01 ..... [TPGs: 1]
    o- tpg1 ..................... [no-gen-acls, no-auth]
      o- acls ................................... [ACLs: 0]
      o- luns ................................... [LUNs: 0]
      o- portals ................................ [Portals: 1]
        o- 0.0.0.0:3260 ............................. [OK]
```

Configurare il TPG

- **ACL**: vanno create indicando quali LUN sono accessibili da quali initiator (indicando l'IQN). Generalmente questo processo e' indicato con il nome di **LUN mapping**
  - Il parametro `auto_add_mapped_luns` e' di default settato a True. Cio' significa una ACL consente ad un initiator l'accesso a tutte le LUN
- **LUN**: ad ogni backstore che si desidera rendere disponibile va associata una LUN
- **Portal**: creare un portal significa indicare quale indirizzo IP e porta TCP dovranno essere designati per esportare le LUN
  - Non specificare l'IP e la porta equivale a configurare 0.0.0.0:3260

Esempio: configurare un ACL

```
/iscsi/iqn.20...sidisk01/tpg1> acls/ create iqn.2018-04.com.example:iscsiclient01
Created Node ACL for iqn.2018-04.com.example:iscsiclient01
/iscsi/iqn.20...sidisk01/tpg1> ls
o- tpg1 ..................................... [no-gen-acls, no-auth]
  o- acls ..................................................... [ACLs: 1]
  | o- iqn.2018-04.com.example:iscsiclient01 ..... [Mapped LUNs: 0]
  o- luns ..................................................... [LUNs: 0]
  o- portals ................................................ [Portals: 1]
    o- 10.4.0.96:3260 ..................................... [OK]
```

Esempio: configurare un portal

```
/iscsi/iqn.20...sidisk01/tpg1> portals/ delete 0.0.0.0 3260
Deleted network portal 0.0.0.0:3260
/iscsi/iqn.20...sidisk01/tpg1> portals/ create 10.4.0.96
Using default IP port 3260
Created network portal 10.4.0.96:3260.
/iscsi/iqn.20...sidisk01/tpg1> ls
o- tpg1 ..................................... [no-gen-acls, no-auth]
  o- acls ....................................................... [ACLs: 0]
  o- luns ....................................................... [LUNs: 0]
  o- portals ................................................. [Portals: 1]
    o- 10.4.0.96:3260 ................................................ [OK]
/iscsi/iqn.20...sidisk01/tpg1>
```

Esempio: configurare le LUN

```
/iscsi/iqn.20...sidisk01/tpg1> luns/ create /backstores/block/block1
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.2018-04.com.example:iscsiclient01
/iscsi/iqn.20...sidisk01/tpg1> luns/ create /backstores/fileio/file1
Created LUN 1.
Created LUN 1->1 mapping in node ACL iqn.2018-04.com.example:iscsiclient01
/iscsi/iqn.20...sidisk01/tpg1> ls
o- tpg1 ..................................... [no-gen-acls, no-auth]
  o- acls ....................................................... [ACLs: 1]
  | o- iqn.2018-04.com.example:iscsiclient01 ..... [Mapped LUNs: 2]
  |   o- mapped_lun0 ..................... [lun0 block/block1 (rw)]
  |   o- mapped_lun1 ..................... [lun1 fileio/file1 (rw)]
  o- luns ....................................................... [LUNs: 2]
  | o- lun0 ........... [block/block1 (/dev/vdc) (default_tg_pt_gp)]
  | o- lun1 .... [fileio/file1 (/root/disk01.img) (default_tg_pt_gp)]
  o- portals ................................................. [Portals: 1]
    o- 10.4.0.96:3260 ..................................... [OK]
```

Accesso allo storage iSCSI

- In CentOS 7 la funzione dell'initiator e' svolta dal software Open-iSCSI. Pacchetto da installare e' `iscsi-initiator-utils`
- Il comando `iscsiadm` consente di eseguire *discovery* e *login* manualmente, e di gestire i database dei nodi e delle interfacce
- I servizi `iscsid` e `iscsi` rispettivamente gestiscono le connesioni verso i target ed il login automatico agli stessi.
- Il nome IQN dell'initiator deve essere inserito nel file `/etc/iscsi/initiatorname.iscsi` ed e' quello che viene utilizzato nella ACL del target.
- Le risosre disponibili dopo una sessione di *discovery* vengono inserite nel file `/var/lib/iscsi/nodes`, la lista delle interfacce utilizzabili in `/var/lib/iscsi/ifaces` e la lista dei possibili host verso cui eseguire una discovery in `/var/lib/iscsi/discoverydb`

Esempio: configurare un initiator

- Dopo aver installato il pacchetto `iscsi-initiator-utils` verificare che l'IQN sia quello configurato nelle ACL del target
- Avviare il servizio `iscsid`
- Eseguire un *discovery* verso il target

```
[root@h ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2018-04.com.example:iscsiclient01
[root@h ~]# systemctl start iscsid
[root@h ~]# iscsiadm -m discovery -t sendtargets -p
  10.4.0.96:3260
10.4.0.96:3260,1 iqn.2018-04.com.example:iscsidisk01
```

Esempio: configurare un initiator

- Verificare che il *discovery* sia andato a buon fine esaminando il database dei nodi ed il discoverydb
- Eseguire un *iscsi login* al nodo target
- Verificare la presenza dei device

```
[root@host101 ~]# systemctl status iscsi -l
[root@host101 ~]# iscsiadm -m node
10.4.0.96:3260,1 iqn.2018-04.com.example:iscsidisk01
[root@host101 ~]# iscsiadm -m discoverydb
10.4.0.96:3260 via sendtargets
```

Esempio: configurare un initiator

```
[root@localhost ~]# iscsiadm -m node -n iqn.2018-04.com.example:iscsitarget01 -Lall
Logging in to [iface: default, target: iqn.2018-04.com.example:iscsitarget01, portal: 10.4.0.96,3260] (multiple)
Login to [iface: default, target: iqn.2018-04.com.example:iscsitarget01, portal: 10.4.0.96,3260] successful.
[root@host101 ~]# lsscsi
[1:0:0:0] cd/dvd QEMU QEMU DVD-ROM    0.12  /dev/sr0
[2:0:0:0] disk   LIO-ORG   block1     4.0   /dev/sda
[2:0:0:1] disk   LIO-ORG   file1      4.0   /dev/sdb
```

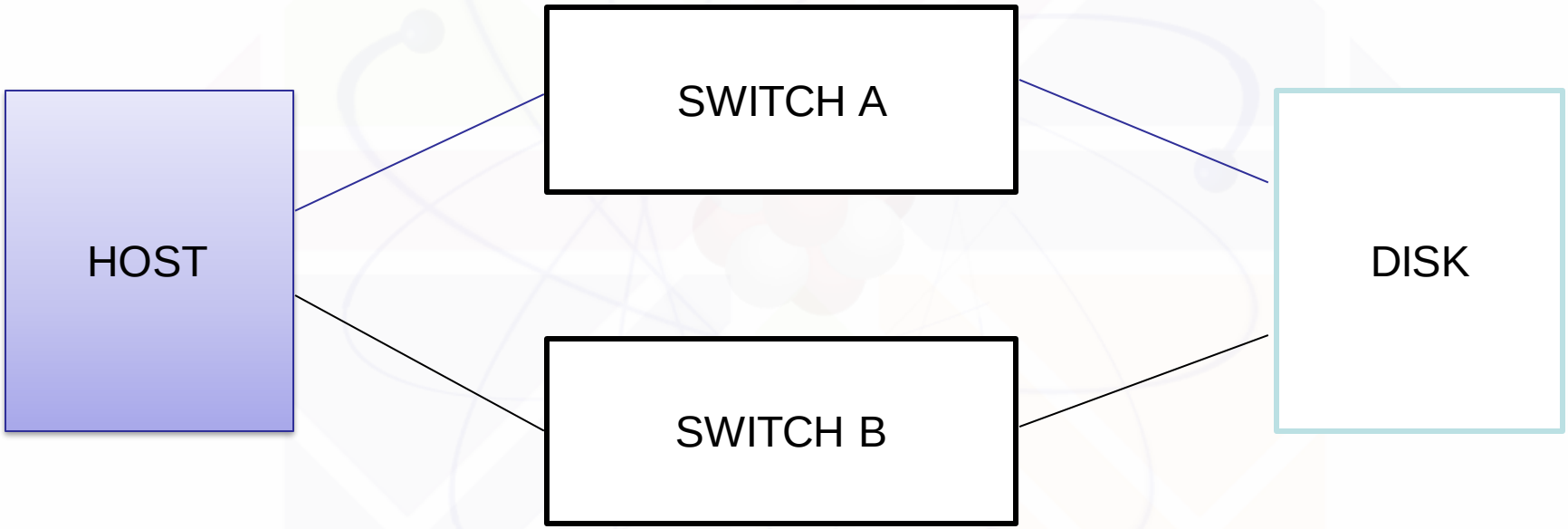
- In questo caso la LUN 0 (il device a blocchi) e' stata associata al device /dev/sda e la LUN 1 (ramdisk) al device /dev/sdb

Istituto Nazionale di Fisica Nucleare

Un accenno al multipath

Multipath

- Modulo del device mapper che consente di gestire storage device raggiungibili via SAN attraverso diversi percorsi (path)



Elementi di multipath

- Comando `multipath:` per la configurazione ed il monitoring del sistema
- Servizio `multipathd:` gestisce la modalita' di utilizzo dei percorsi di accesso
- File `/etc/multipath.conf:`
  - configurazioni di default di `multipathd`
  - lista dei device che non devono essere gestiti (black list)
  - configurazioni specifiche di singoli device

Multipath: getting started

- Installazione:

  `yum install device-mapper-multipath`

- Template di configurazione:

  `/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf`

- Start del servizio:

  `systemctl start multipathd`

- Lista dei device e dei percorsi:
  - `multipath -ll`

Esempio di multipath

```
[root@localhost ~]# multipath -ll
iscsi_block (36001405edba3eb54d3449369d4805a6f) dm-0
LIO-ORG ,block1
size=1.0G features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=50 status=active
  |- 3:0:0:0 sdb 8:16 active ready running
  `- 4:0:0:0 sdd 8:48 active ready running
iscsi_file (3600140517c73e880e8a4d408be88cd42) dm-1
LIO-ORG ,file1
size=1.0G features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=50 status=active
  |- 3:0:0:1 sdc 8:32 active ready running
  `- 4:0:0:1 sde 8:64 active ready running
```

Istituto Nazionale di Fisica Nucleare

Filesystem locali

Filesystems locali

- Supportati: XFS, EXT4, BTRFS (in technology preview ma deprecato dalla CentOS 7.4)
- XFS si puo' solo estendere mentre EXT4 anche restringere
- Sia XFS che EXT4 possono essere sospesi, rispettivamente con `xfs_freeze` e `freezefs` per poter eseguire snapshot coerenti
  - La snapshot LVM sospende automaticamente il filesystem (EXT4 o XFS) creato sul volume
- XFS supporta la deframmentazione nativamente
- Dalla CentOS 7, EXT4 puo' essere deframmentato con il comando `e4defrag`

XFS vs. EXT4

	Ext 4	XFS
Create a file system	mkfs.ext4 or mkfs.ext3	mkfs.xfs
File system check	e2fsck	xfs_repair
Resizing a file system	resize2fs	xfs_growfs
Save an image	e2image	xfs_metadump and xfs_mdrestore
Label or tune	tune2fs	xfs_admin
Backup a file system	dump and restore	xfsdump and xfsrestore

Istituto Nazionale di Fisica Nucleare

System-storage-manager

SSM

- E' una utility a linea di comando che unifica e semplifica la creazione di volumi, filesystem e snapshot
- Si basa su di un layer di astrazione (`ssmlib/main.py`) che definisce le operazioni di base (create, remove, snapshot, ...)
- Sono implementati, per ora, tre backend: LVM, Crypt, MD e Btrfs (che probabilmente sparira' a breve)
- Ci concentreremo sul backend LVM, il piu' utile ed interessante

SSM LVM backend

- Per installare il backend LVM di SSM, oltre al pacchetto `system-storage-manager` e' necessario installare `lvm2`
- Nota: per SSM il termine **pool** equivale al **volume group** LVM
- I comandi supportati da SSM sono `check, resize, create, list, add, remove, snapshot, mount`

Esempi SSM per LVM

- Creare un volume LVM formattato XFS:

```
~]# ssm create --fs xfs -s 1G /dev/vdb /dev/vdc
Physical volume "/dev/vdb" successfully created
Physical volume "/dev/vdc" successfully created
Volume group "lvm_pool" successfully created
Logical volume "lvol001" created
```

- Incrementare un voume LVM ed il relativo filesystem aggiungendo un nuovo volume fisico:

```
~]# ssm resize -s +500M /dev/lvm_pool/lvol001 /dev/vdc
Physical volume "/dev/vdc" successfully created
Volume group "lvm_pool" successfully extended
data blocks changed from 230400 to 358400
```

Esempi SSM per LVM

- Creare la snapshot di un volume

```
~]# ssm snapshot /dev/lvm_pool/lvol001
Logical volume "snap20150519T130900" created
```

- Rimuovere un pool/volume/snapshot

```
~]# ssm remove lvm_pool
Do you really want to remove volume group "lvm_pool" containing 2
   logical
volumes? [y/n]: y
Do you really want to remove active logical volume
   snap20150519T130900?
[y/n]: y
Logical volume "snap20150519T130900" successfully removed
Do you really want to remove active logical volume lvol001?
   [y/n]: y
Logical volume "lvol001" successfully removed
Volume group "lvm_pool" successfully removed
```

Istituto Nazionale di Fisica Nucleare

E con l'arrivo della 7.5.....

Un po' di storia...

- Red Hat non includeva nel suo OS una soluzione di storage che disponesse di deduplica e compressione on the fly.
  - ZFS e' sotto il controllo di Oracle
  - BTRFS non implementa la deduplica (futuri sviluppi?)
- Red Hat si sfila dallo sviluppo di BTRFS (che rimane il default di SUSE) e, nel 2017, acquisisce Permabit
- Pubblica i sorgenti del software di gestione dello storage di Permabit e li include nella RHEL 7.5, come Virtual Ddata Optimizer
- Nella prossima versione di RHEL, la 8, gia' in versione beta, un nuovo filesystem **Stratis** includera' le funzioni del VDO

Virtual Data Optimizer

- Il VDO e' un layer di virtualizzazione del block device che implementa un meccanismo di deduplica e compressione dei dati "on line"
- **Kvdo**: e' il modulo del kernel che integra il device-mapper per eseguire le operazioni necessarie alle funzioni del VDO
- **Uds**: e' il modulo del kernel che comunica con lo Universal Deduplication Service Index per analizzare i dati da deduplicare

Virtual Data Optimizer

- Per ogni blocco di dati da scrivere sul volume, l'**UDS** determina se ne sia stato gia' scritto uno identico sul block device di backend.
- Il controllo viene eseguito prima attraverso un hash e successivamente (se se ne trova una uguale) byte-by-byte.
- Se il blocco gia' esiste se ne crea solo un riferimento, attraverso il **kvdo**, che sfrutta I meccanismi del device-mapper.

Modalita' di scrittura

- Modalita' **sincrona** del kvdo: Il blocco viene subito scritto sul device fisico, quindi dato l'acknowledgment e successivamente controllato per la deduplica
- Modalita' **asincrona** del kvdo: l'acknowledgment viene dato immediatamente ed il dato scritto sul device solo se non duplicato
- Configurare la prima per storage backend senza cache volatile (o la usa in write-through).
- Importante per evitare perdita di dati in seguito a spegnimenti repentini

Struttura del VDO

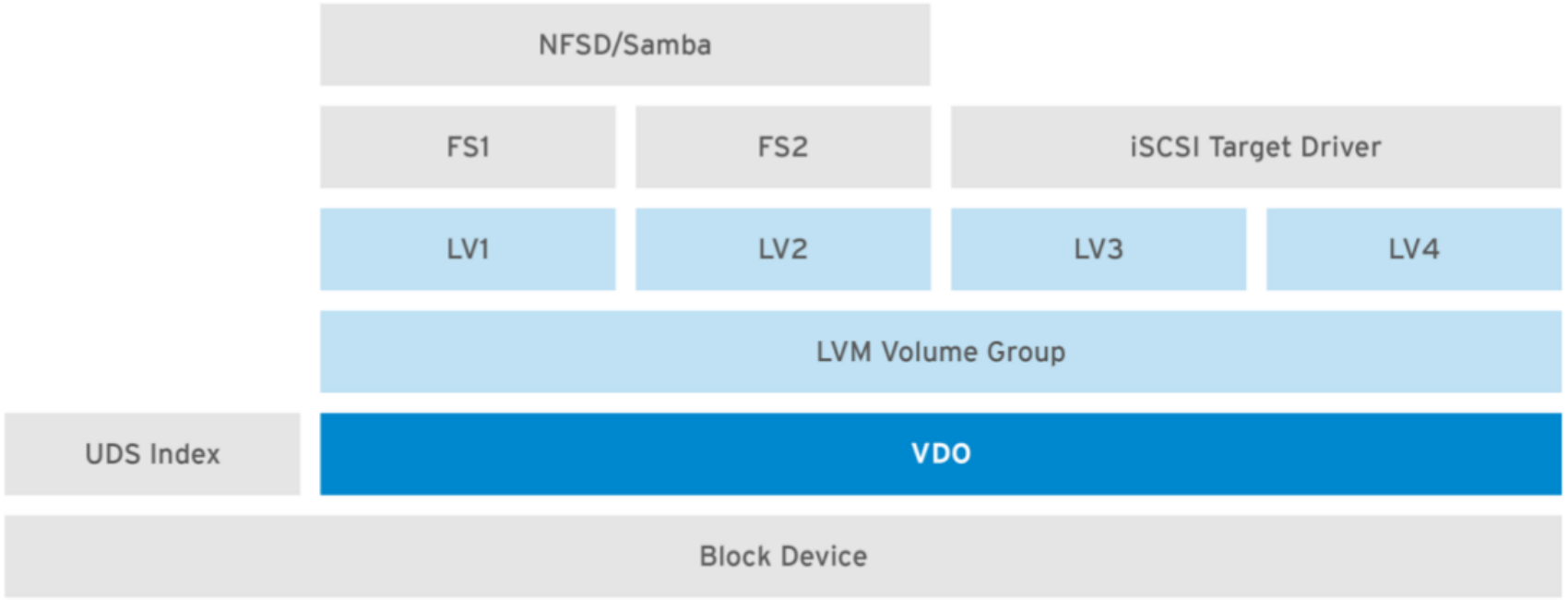
Physical Volume Size	10–99 GB	100 GB – 1 TB	2–10 TB	11–50 TB	51–100 TB	101–256 TB
Slab Size	1 GB	2 GB	32 GB	32 GB	32 GB	32 GB

- Il block device viene diviso in slab
  - Minima dimensione della slab: 1GB
  - Almeno un slab per I metadati
- Massima dimensione di un block device fisico: 256TB
- E' possibile indicare una dimensione logica maggiore di quella fisica

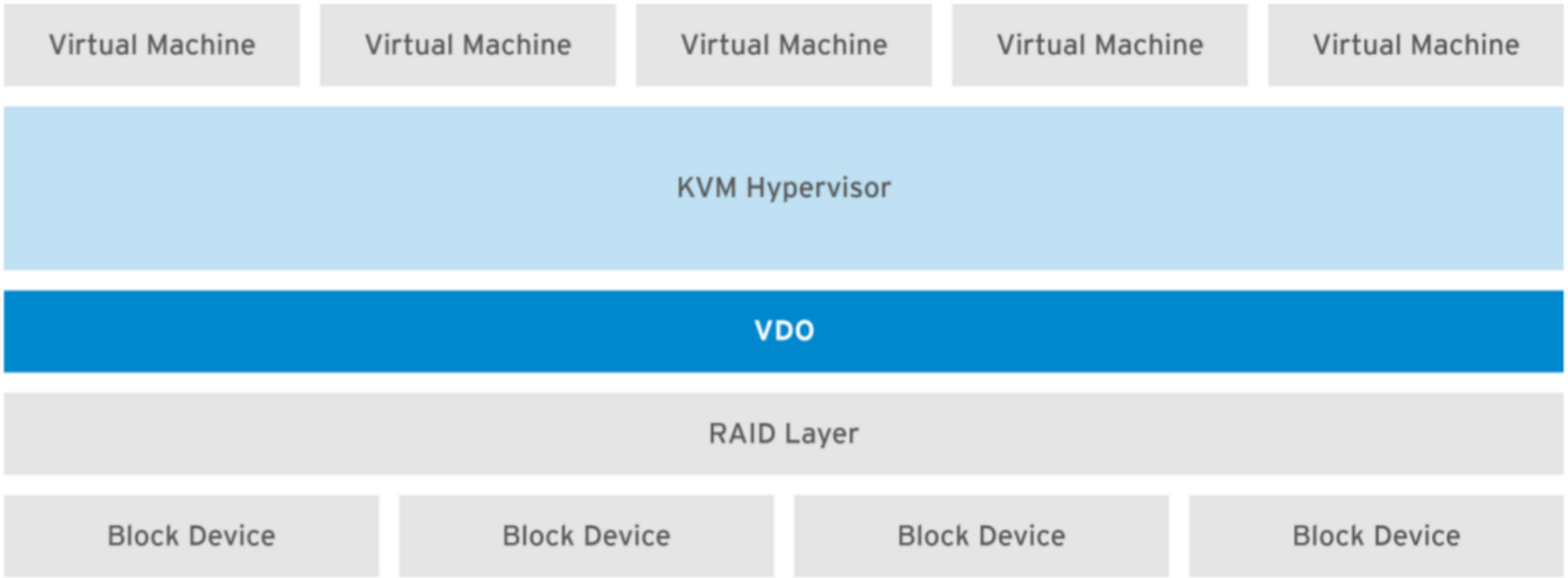
Scenari di deployment

- Possibili layer sottostanti : DM-Multipath, DM-Crypt, and software RAID (LVM or mdraid).
- Possibili layer sovrastanti: LVM cache, LVM Logical Volumes, LVM snapshots, and LVM Thin Provisioning.
- Presentazioni e post ufficiali del team di GlusterFS consigliano l'utilizzo di VDO come beckend per Gluster:
  - https://planet.gluster.org/
  - https://www.slideshare.net/GlusterCommunity/data-reduction-for-gluster-with-vdo

Esempi


NFSD/Samba
FS1
FS2
iSCSI Target Driver
LV1
LV2
LV3
LV4
LVM Volume Group
UDS Index
VDO
Block Device

Esempi



Un cluster di hypervisor (ad es. OVirt) necessita un ulteriore layer di condivisione dello storage tra VDO e KVM, che puo' essere implementato, ad esempio, via GlusterFS

Consumo di risorse

- L'UDS index, per mantenere gli hash dei blocchi da deduplicare, occupa porzioni di RAM e disco
- Deduplication window: la dimensione dello storico mantenuto nell'indice
- **Sparse Index**: modalita' di consumo contenuto ma minore percentuale di deduplica
  - 10 Tera di deduplication window per 1GB di RAM
- **Deep Index**: maggiore percentuale di deduplica e maggiore consumo di risorse
  - 1 Tera di deduplication windows per 1GB di RAM

Consumo di risorse: due casi comuni

- Storage primario

Physical Volume Size	10 GB – 1–TB	2–10 TB	11–50 TB	51–100 TB	101–256 TB
RAM Usage	250 MB	Dense: 1 GB Sparse: 250 MB	2 GB	3 GB	12 GB
Disk Usage	2.5 GB	Dense: 10 GB Sparse: 22 GB	170 GB	255 GB	1020 GB
Index Type	Dense	Dense or Sparse	Sparse	Sparse	Sparse

Consumo di risorse: due casi comuni

- Storage di backup

Physical Volume Size	10 GB – 1 TB	2–10 TB	11–50 TB	51–100 TB	101–256 TB
RAM Usage	250 MB	2 GB	10 GB	20 GB	26 GB
Disk Usage	2.5 GB	170 GB	850 GB	1700 GB	3400 GB
Index Type	Dense	Sparse	Sparse	Sparse	Sparse

Compressione vs. deduplica

- VDO supporta la compressione a livello di blocco tramite HIOPS (algoritmo sviluppato di Permabit)
  - E' attivata di default alla creazione di un volume
- Casi d'uso:
  - Compressione: log files, databse
  - Deduplica: backup, virtual machines repository, object storage repository
- La quantita' di storage risparmiato dipende dal tipo di contenuto. RedHat consiglia:
  - VM repository: proporzione logical/physical -> 10:1
  - Object storage: proporzione logical/physical -> 3:1

Comandi VDO

- Installazione:

```
yum install vdo kmod-vdo
```

- Start:

```
vdo start
systemctl start vdo
```

- Creazione di un volume:

```
vdo --create --name=vdoname --device=/dev/devname  --vdoLogicalSize=<size><G|T>
```

- Proprieta' del volume:

```
vdostats -- human-readable
vdo status
```