

Tecniche per la salvaguardia della disponibilità ed integrità dei sistemi di elaborazione e delle informazioni

1. Introduzione all'affidabilità; RAID

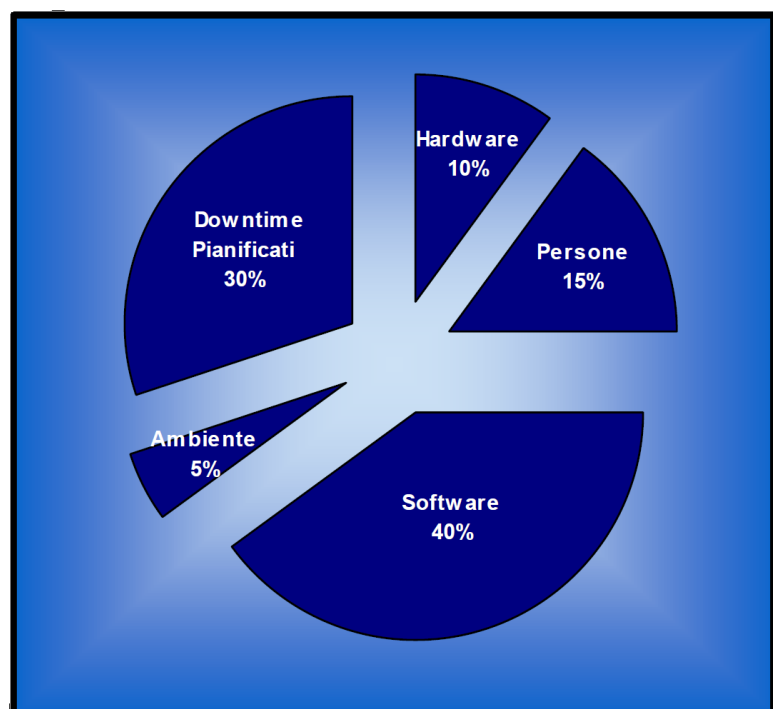
Marco Prandini
Università di Bologna

Disponibilità / Downtime

- La **disponibilità** o **availability** di un sistema è il rapporto tra il tempo per cui eroga correttamente i servizi (**uptime**) rispetto al tempo per cui ci si attende che lo faccia (tempo di osservazione)

$$A=U/O$$

- Il tempo durante il quale il servizio non è erogabile viene chiamato **downtime**



Cause di downtime per incidenza sul totale

Livelli di disponibilità

- Comunemente la disponibilità viene indicata in modo sintetico col “numero di 9” nella percentuale di uptime
 - aggiungere un 9 significa dividere per 10 il downtime

disponibilità %	downtime per anno	downtime per mese	downtime per settimana
98%	7,3 giorni	14,4 ore	3,36 ore
99%	3,65 giorni	7,20 ore	1,68 ore
99,5%	1,83 giorni	3,60 ore	50,4 minuti
99,9%	8,76 ore	43,2 minuti	10,1 minuti
99,99%	52,6 minuti	4,32 minuti	1,01 minuti
99,999%	5,26 minuti	25,9 secondi	6,05 secondi
99,9999%	31,5 secondi	2,59 secondi	0,605 secondi

3

Service Level Agreement

- La disponibilità è un indicatore sintetico ma non esaustivo
- Un contratto sul livello di servizio (*Service Level Agreement*) può prevedere vincoli più stringenti
 - Sulla distribuzione del downtime (es.: “four nines” nell’arco dell’anno ma in frazioni non superiori a n minuti consecutivi)
 - Su parametri di servizio non collegati al downtime
 - Prestazioni dei servizi
 - Tipologie di assistenza previste e loro caratteristiche
 - ...
- Oggetto del contratto tipicamente è anche il contratto stesso
 - Modalità di aggiornamento
 - Reportistica periodica sulle variabili monitorate

4

Ambiente

Prima di affrontare gli aspetti più propriamente sistemistici, uno sguardo a quel che c'è intorno

- **Per erogare con continuità un servizio, il sistema deve essere acceso e connesso!**
 - Le infrastrutture che garantiscono un ambiente affidabile sono complesse e molto costose → indispensabile dividerle
- **Data center o server farm sono i luoghi in cui vengono ospitati in grande quantità i sistemi di calcolo**
 - Housing o Co-location: fornitura di spazio e connettività per sistemi acquistati e gestiti dal cliente
 - Managed housing: fornitura dei sistemi in housing (su hardware comunque dedicato al cliente) e loro gestione sistemistica
 - Hosting: fornitura di uno o più servizi specifici (storage, web, posta, ...) su hardware condiviso tra più clienti
 - Il modello cloud è un caso speciale dell'hosting tradizionale

5

Problematiche principali di un data center

- **Resistenza della struttura**
 - cause naturali: terremoti, inondazioni, ...
 - cause artificiali: incidenti aerei e ferroviari, terrorismo, ...
 - cause interne: incendi, da controllare con sistemi che consentano l'intervento anche quando l'incendio stesso li danneggia parzialmente
- **Sicurezza e controllo degli accessi**
- **Condizionamento dell'aria**
 - gestione di temperatura e umidità con sistemi tolleranti ai guasti e alle interruzioni di erogazione dell'energia elettrica
- **Condizionamento dell'alimentazione elettrica**
 - erogazione su almeno due linee indipendenti per ogni apparato
 - pulizia della sinusoide per allungare la vita degli apparati
 - sistemi di continuità ad intervento istantaneo e di durata prolungata
 - combinano motogeneratori che possono sopprimere a giorni di mancata erogazione, ma richiedono 15-20 minuti per l'avviamento, a sistemi a batteria ad intervento istantaneo ma di bassa capacità
- **Connettività di rete**
 - connessione tramite provider indipendenti
 - collocazione fisica dei cavi su percorsi indipendenti

Per avere un'idea: <http://fibertown.com/houston-colocation-data-center/>

6

Disponibilità – scopo ed estensione

- Disponibilità di un sistema o di un'informazione è la possibilità di accedervi quando necessario → il riferimento temporale è importante nelle scelte progettuali
 - disponibilità istantanea o continua
 - disponibilità a lungo termine
- Per garantire la disponibilità continua di un sistema si può agire
 - sulla costruzione dei componenti
 - es. tecnologia meccanica, grading dei componenti elettronici
 - riduce il Mean Time To Failure (MTTF) o Mean Time Between Failures (MTBF)
 - sull'architettura
 - eliminazione dei **single point of failure** per mezzo della **replicazione**
 - introduce complessità, oltre certi limiti richiede una riprogettazione globale
 - ove possibile, si agisce sull'architettura, ma in modo da produrre componenti autocontenuti che non differiscano "ai morsetti" da quelli tradizionali
 - es. memorie ECC, wear-leveling SSD

7

Disponibilità continua dei dati

- Cosa vedremo in questa parte di presentazione
 - 1) Un rapido sommario delle tecniche di replicazione ed accesso ai dati
 - RAID
 - Multipath
 - Filesystem distribuiti
 - 2) Un'illustrazione dei principali strumenti di Linux per implementarle
 - md/dm
 - DRBD
 - NFS
 - GFS

8

RAID

- Il componente più soggetto a guasti è quello sollecitato meccanicamente: l'hard disk
- La tecnologia più diffusa per proteggere un sistema dal guasto di un disco è RAID (Redundant Array of Independent Disks)
 - il costo di un disco cresce più rapidamente del MTTF
 - per quanto sofisticato, un disco cede spesso senza preavviso
 - anzichè investire su di un disco più robusto, si combinano N dischi economici in modo che il fallimento di uno di essi non comprometta il funzionamento dell'unità logica risultante
 - complessità aggiuntiva: gestione dei metadati di allocazione dati
- Implementazioni
 - HW: il controller sgrava la CPU dei calcoli necessari e nasconde al sistema operativo i dettagli, mostrando l'analogo di un disco; può assistere nel mantenere la coerenza dei dati in caso di crash
 - SW: economico, sfrutta i cicli idle della CPU

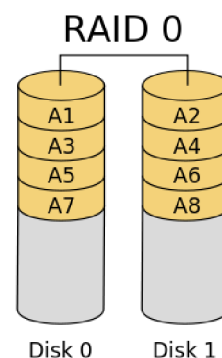
9

RAID levels

Tutte le immagini sono tratte da Wikipedia (<http://en.wikipedia.org/wiki/RAID>), realizzate da Colin M.L. Burnett (<http://en.wikipedia.org/wiki/User:Cburnett>), e pubblicate in accordo alla licenza CC Attribution-Share Alike

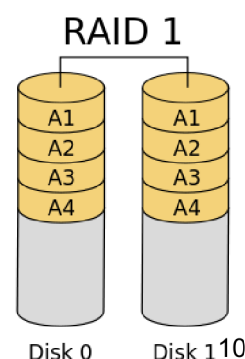
■ RAID0 – STRIPING

- distribuisce, non replica, i dati tra tutti i dischi dell'array
 - un guasto corrompe l'intero array
→ **non esattamente un RAID**
- usato per incrementare le prestazioni
 - se il collo di bottiglia è il throughput del singolo disco, B bytes scritti dal sistema incidono per B/N bytes su ciascuno degli N dischi



■ RAID1 – MIRRORING

- replica ogni dato identico su **tutti** i dischi dell'array
 - basta che un solo disco sopravviva per mantenere i dati
 - fornisce un incremento di velocità in lettura perchè dati diversi possono essere letti contemporaneamente da dischi diversi
 - costoso in termini di capacità



RAID levels

■ Striping con ridondanza

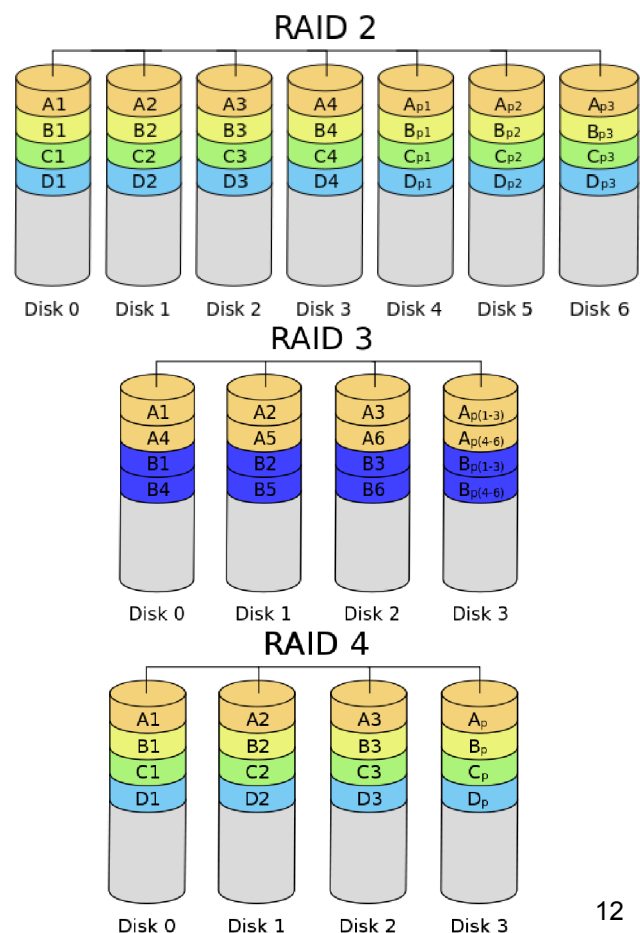
- mutuato dal RAID0
 - un dato viene suddiviso in “strisciole” destinate a diversi dischi
- a partire dalle diverse **stripe** (o segment), un algoritmo genera uno o più dati aggiuntivi, che vengono scritti su altrettanti dischi aggiuntivi, e permettono di ricalcolare una o più stripe originarie in caso di perdita
- per le prestazioni è cruciale la relazione tra dimensioni delle stripe, dimensioni dei blocchi gestiti su disco, ed algoritmo di calcolo

11

RAID levels

■ RAID2/3/4 – diversi tipi di sistemi a correzione d'errore

- 2: Hamming code per correggere anche errori multipli
 - 3: bitwise striping con parità su disco dedicato
 - 4: blockwise striping con parità su disco dedicato
 - sostanzialmente in disuso
 - caratterizzati dalla presenza di dischi dedicati ai dati e dischi dedicati alle informazioni per la correzione degli errori
- **collo di bottiglia**



12

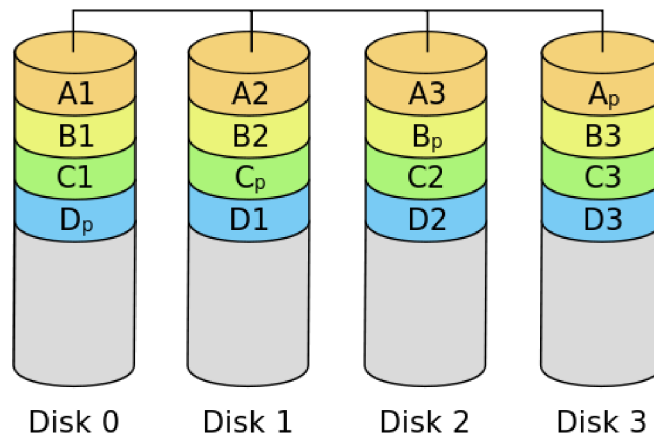
RAID levels

■ RAID5 – DISTRIBUTED PARITY

- ogni blocco di dati viene elaborato per produrre 1 blocco di parità
- con N-1 frammenti qualunque, si può ricostruire il blocco originario
 - prestazioni simili allo striping, ma con tolleranza al guasto di **un** disco

■ Read-in

- Per aggiornare la parità "**p**" quando cambia un dato "**d**" si esegue:
$$\text{new_p} = \text{old_p} \oplus \text{old_d} \oplus \text{new_d}$$
- **due letture** e **due scritture** di intere stripe
- molto penalizzante per scritture piccole (dim. blocco $\ll$ dim. stripe)



13

Recovery & Reconstruction

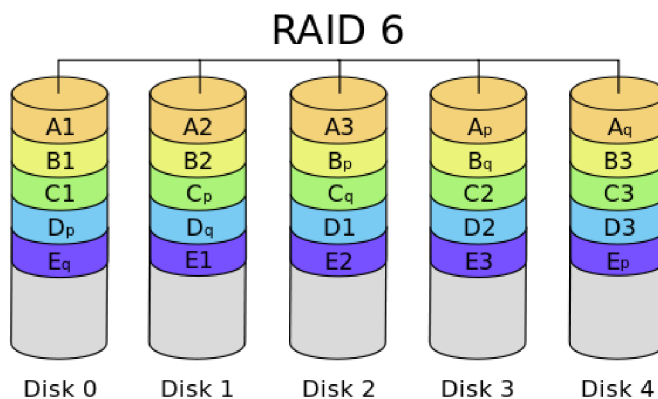
- Quando un disco si guasta, l'array funziona in **degraded mode**
- Ogni lettura deve coinvolgere **tutti dischi** e deve essere ricalcolato il dato perso applicando l'algoritmo di correzione d'errore
 - Fa eccezione il RAID1 in cui ogni disco ha una copia di tutti i dati
- Se si tratta di una disconnessione temporanea, alla riconnessione si avvia la procedura di resync
 - se i metadati sono coerenti e aggiornati, si identificano solo i blocchi da aggiornare ricalcolandoli dagli altri dischi
- Quando il disco deve essere sostituito, la procedura di **recovery** viene applicata all'intero contenuto dell'array per ripopolare il nuovo disco: **reconstruction** o **rebuild**
 - con le dimensioni dei dischi attuali può richiedere **decine di ore**
 - carico elevato, su dischi tipicamente identici per marca, modello ed età
 - rischio di ulteriore guasto, **irrecuperabile in RAID5**
 - basta anche UN settore illeggibile, che è quasi una certezza nei sistemi low-end caratterizzati da grandi dischi poco costosi (i sistemi di fascia media e alta tipicamente usano dischi più performanti ma per questo più piccoli, ma l'avanzamento tecnologico sta chiudendo il gap)
 - URE (unrecoverable read error rate) tipico dischi SATA = **1 ogni $2 \cdot 10^8$** settori
 - in un array da 6 dischi x 2TB ci sono circa **$2 \cdot 10^{10}$** settori...
 - soluzione: controllo periodico dei settori danneggiati (pesante ma salvifico)

14

RAID levels

■ RAID6 – DISTRIBUTED DOUBLE PARITY

- ogni blocco di dati viene elaborato per produrre 2 *sindromi*
- con N-2 frammenti qualunque si può ricostruire il blocco originario
→ tolleranza al guasto contemporaneo di due dischi



15

RAID – tabella riassuntiva

LIVELLO	CAPACITA'	GUASTI TOLLERATI	PRESTAZIONI TEORICHE	TASSO DI GUASTO	Esempio per $r=1\%$ e $N=6$
0	N	0	$N \times (R^*, W)$	$1 - (1-r)^N$	5.8%
1	1	N-1	$N \times (R)$	r^N	10^{-12}
5	N-1	1	$(N-1-oh) \times (R^*, W^{\wedge})$	$(1/2)N(N-1)r^2$	0.15%
6	N-2	2	$(N-2-oh) \times (R^*, W^{\wedge})$	$(1/6)N(N-1)(N-2)r^3$	0.002%

N=numero dischi

R=velocità base in lettura

W=velocità base in scrittura

oh=overhead computazionale

r=tasso di guasto del singolo disco

* ma attenzione al seek time
[^] a volte molto peggiori
 se necessario read-in,
 specialmente per RAID6

16

RAID – livelli composti

- Con un elevato numero di dischi non è sensato comporre un unico array
 - con RAID0 intollerabile rischio di guasto
 - con RAID1 intollerabile spreco di capacità
 - con RAID5 discreto rischio di guasto doppio e problemi di calcolo
- Un RAID viene visto come un block device → può essere trattato come un disco con cui costruire un altro RAID
- Tipicamente
 - un livello è RAID0
 - aggrega i dischi senza spreco di capacità
 - moltiplica il transfer rate
 - un livello sopperisce alla diminuita robustezza
 - RAID1 (→ RAID 10) o RAID5 (→ RAID 50)

17

RAID – spare disks

- Negli aggregati composti da un gran numero di dischi, le probabilità di fallimenti multipli contemporanei sono significative
 - casi particolarmente gravi: design difettoso su banchi di dischi tutti dello stesso modello
- Problema: intervenire in tempi rapidi (→ automaticamente) alla segnalazione di un fail
 - RAID6
 - penalità di prestazioni rispetto al RAID5
 - non risolve il problema oltre un certo numero di dischi
 - HOT SPARES: dischi presenti nell'array ma non normalmente usati, che sostituiscono automaticamente un disco failed
 - richiesto tempo di rebuild
 - nessun impatto durante il funzionamento normale
 - condivisibili tra molti array per ottimizzare i costi

18

RAID – quanti dischi e quale layout?

- RAID5 è il miglior compromesso prezzo-prestazioni ma con le dimensioni attuali può essere molto rischioso
- RAID6 è meno efficiente e comunque è consigliabile uno spare disk per sostituire immediatamente un failed disk
- Tipica enclosure da 12 slot:



- 2 array RAID6 + spare: 6 dischi utili + 5 o 6 di sicurezza



- Alternativa RAID10 (striping su 5 mirror di 2 dischi + spare)

- Capacità -16%
- Tasso di guasto molto simile
- Minori problemi di rebuild
- Prestazioni più omogenee per pattern di uso variegati

19

Multipath

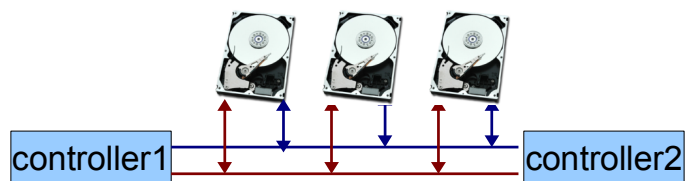
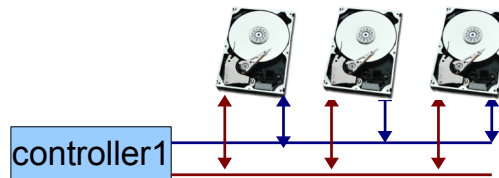
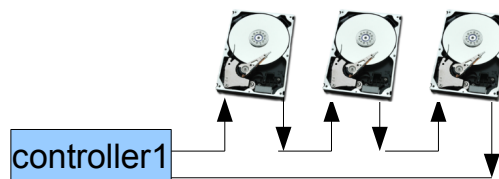
- Oltre alla ridondanza dei dischi, può essere necessario prevedere tolleranza ai guasti
 - dei componenti di controllo
 - dei percorsi di connessione
- Necessario:
 - hardware che permette di creare più percorsi di accesso allo stesso disco
 - supporto O.S. per gestire correttamente la molteplicità di percorsi
 - in Linux sia md che dm (oltre a driver proprietari)

20

Multipath – hardware necessario

■ Percorso disco-controller

- Dischi FC-AL: due connettori in/out per predisporre un loop
- Dischi SAS *dual port*: due porte da innestare su due backplane
- Porte diverse possono andare a porte multiple del controller o a controller separati



21

Multipath – hardware necessario

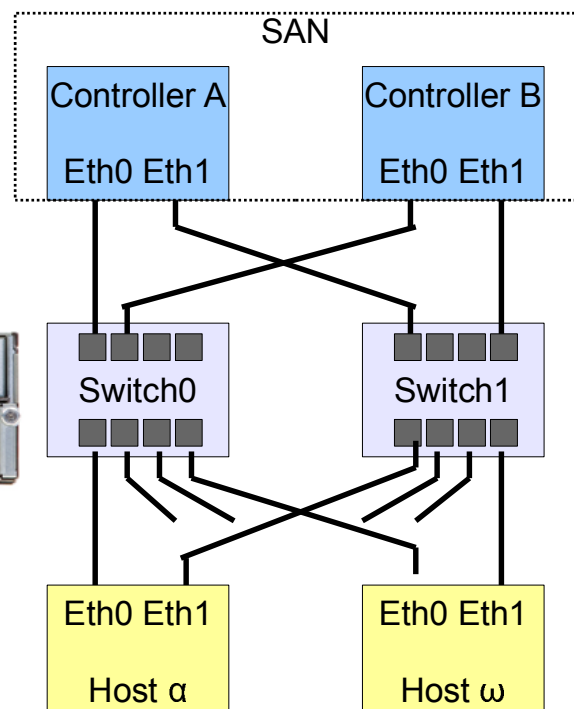
■ Controller

- Può essere installato in locale sull'host o essere collegabile in rete
 - Può disporre di una molteplicità di porte di rete



■ Percorso controller-host via rete

- Raramente diretto, solitamente via switch
- Se sono presenti più porte di rete per controller e per host, possibilità di duplicare gli switch



22

Distributed Filesystems

- Un componente essenziale per la realizzazione di sistemi scalabili ed affidabili è un sistema di storage condiviso
- Esistono due grandi approcci: NAS e SAN
- NAS
 - i dati sono su di un sistema che li conserva in un filesystem locale
 - i dettagli del filesystem sono pressochè trasparenti ai client
 - esiste un protocollo client/server per l'erogazione dei file
 - i client operano senza cooperare tra loro
 - punto di forza: la cooperazione può essere complessa e dare problemi di consistenza in caso di guasto
 - L'implementazione classica di NAS cambia a seconda del S.O.
 - Nel mondo UNIX è fatta tipicamente tramite il protocollo NFS
 - Nel mondo Microsoft tramite il protocollo CIFS

23

Distributed Filesystems

- SAN
 - i dati sono su di un sistema che li conserva allo stato grezzo
 - esiste un protocollo client/server per l'erogazione dei blocchi
 - punto di forza: il protocollo è più leggero, ed esistono implementazioni per garantire alta efficienza, scalabilità e robustezza in HW (trasparente al S.O.)
 - il filesystem è quindi definito dai client, che devono cooperare perchè l'accesso concorrente al dispositivo non causi inconsistenza e corruzione dei dati
 - per mezzo di sistemi di locking distribuito ...
 - CLVM (Clustered Logical Volume Manager)
 - ... ed eventualmente di *cluster filesystem*
 - RedHat GFS (Global File System)
 - IBM GPFS (General Parallel File System)
 - Oracle OCFS (Oracle Cluster File System)
 - Le tecnologie SAN più diffuse sono
 - AoE (ATA over Ethernet), **economico, molto efficiente**, solo LAN, MAC auth.
 - iSCSI, **instradabile, auth sofisticata, pesante, un po' più costoso**
 - FC-AL, **molto scalabile, infrastruttura dedicata costosissima**

24

Supporto multi-device in Linux

- **multi-device support: una generalizzazione del RAID**
 - far cooperare più dispositivi reali per presentare all'utente un dispositivo virtuale con caratteristiche migliori
- **Linux offre due sistemi di driver**
 - **md (multiple devices)**
 - erede degli storici raidtools
 - orientato alla costruzione di RAID software
 - supporto al multipath
 - <http://www.linuxfoundation.org/collaborate/workgroups/linux-raid>
 - **dm (device mapper)**
 - architettura più generale
 - fondazione per diverse funzionalità
 - multipath
 - logical volume management (LVM)
 - volumi cifrati
 - supporto RAID limitato a 0, 1, linear
 - tool per l'integrazione con i controller ATARAID
 - <http://sources.redhat.com/dm/>

25

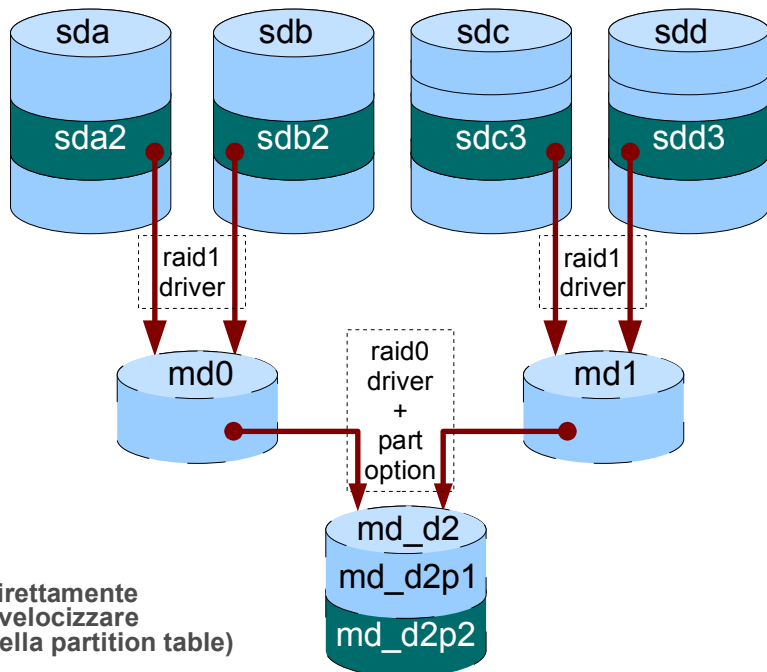
Software RAID in Linux con md

- **md supporta nativamente**
 - linear
 - raid level standard 0, 1, 4, 5, 6
 - raid 10 (sperimentale)
 - più efficiente che sovrapporre manualmente uno strato di striping ad una serie di coppie di dischi in mirror
 - resta comunque la possibilità di combinare dispositivi md per formare altri livelli composti
 - multipath
 - faulty
 - per testing, o far pratica con situazioni di guasto
- **driver modulare nel kernel**
 - si possono caricare singolarmente, oltre al modulo base, i moduli per i diversi livelli
 - 4/5/6 sono in un unico modulo

26

Software RAID in Linux con md

- Il driver permette di combinare qualsiasi *block device* per formare un nuovo *md block device*, che può essere usato come se fosse un normale disco



- può rientrare come device membro di un altro array md
- può essere partizionato purchè predisposto alla creazione

(di solito si creano md per usarli direttamente come volumi, quindi il default è di velocizzare l'accesso eliminando la gestione della partition table)

27

Software RAID in Linux con md

- Il tool per la configurazione e gestione degli array è *mdadm*. Opera in 7 modi diversi:
 - **Create:** creazione di un nuovo array ed inizializzazione dei superblock
 - **Assemble:** avvio di un array esistente (tipicamente al reboot)
 - **Follow/Monitor:** monitoraggio continuo dello stato dell'array
 - **Build:** creazione di un nuovo array senza inizializzazione dei superblock (non potrà essere riassembleato automaticamente successivamente – solo per casi particolari)
 - **Grow:** riconfigurazione della geometria, es:
 - cambiamento della dimensione dei componenti di un RAID 1/4/5/6
 - cambiamento del numero di device attivi in un RAID1.
 - **Manage:** aggiunta di nuovi dischi spare, rimozione di dischi failed
 - **Misc:** everything else...

28

Software RAID in Linux con md

■ Il persistent superblock

- è presente su tutti i dischi dell'array
- contiene metadati
 - sulla struttura dell'array
 - sullo stato dell'array
 - sullo stato di consistenza del disco rispetto all'array
- consente la rilevazione automatica
 - riassettaggio senza bisogno di ricorrere a file di configurazione
 - uso di UUID: indipendenza dai nomi fisici dei dispositivi
 - ancora il default, ma deprecato

■ È sempre bene avere un file `/etc/mdadm.conf` allineato alla configurazione del sistema

- può essere creato con
`mdadm --detail --scan > /etc/mdadm.conf`

29

Software RAID in Linux con md

■ Esempi di base

- inizializzare un array:

```
mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

- avviare un array già inizializzato

```
mdadm -As /dev/md0 /dev/sdb1 /dev/sdc1
```

- arrestare un array

```
mdadm -S /dev/md0
```

- rimuovere un disco da un array

```
mdadm /dev/md0 --fail /dev/sdc1 --remove /dev/sdc1
```

- aggiungere “in corsa” un disco ad un array senza cambiarne la struttura

- riaggiungere un disco precedentemente rimosso
- aggiungere un hot spare

```
mdadm /dev/md0 --add /dev/sdc1
```

- avviare il demone di monitoring

```
mdadm --monitor -f --mail=root@localhost --delay=1800 /dev/md0
```

30

Software RAID in Linux con md

■ Per vedere lo stato degli array in funzione:

```
# cat /proc/mdstat
Personalities : [raid0] [raid1]
md0 : active raid0 sdc1[1] sdb1[0]
      20352 blocks 64k chunks
```

■ Maggiori dettagli

- su di un array

```
mdadm --detail /dev/md0
```

- su di un componente

```
mdadm --examine /dev/sdc1
```

31

Software RAID in Linux con md

■ Quando le cose vanno male, ma non troppo (i.e. è fallito un numero di dischi che, in virtù del livello di RAID scelto, consente all'array di continuare ad operare

- *recovery*: si popola un nuovo disco da cima a fondo coi dati letti dagli altri
- *resync*: si aggiorna un disco che già faceva parte dell'array con i soli dati cambiati

■ Quando si sfiora il disastro (i.e. risulta fallito un numero di dischi che, in virtù del livello di RAID scelto, non lascia dati sufficienti per il funzionamento)

- può capitare se il fail multiplo è causato ad esempio da un canale del controller (ovviamente se sono veramente rotti i dischi non c'è nulla da fare)
- il disallineamento dei superblock previene l'assemblaggio dell'array
- si può tentare un recovery forzato
 - 1) si rendono nuovamente accessibili i dischi
 - 2) con `mdadm -E` si verifica quale ha (quali hanno per raid6) l'event count più basso e lo/li si marca failed nel file di configurazione
 - 3) con `mdadm --assemble --force` si forza il riassemblaggio; se i danni non sono gravi, dai dischi più aggiornati si può ricostruire l'array

32