Engineer

RAID

☰ flera diskar som körs som en, oftast för att tåla diskfel

Vad det är

RAID kör flera diskar som en enda blockenhet, så att gruppen blir snabbare, större eller svårare att slå ut än någon av diskarna den består av. Namnet kommer från en teknisk rapport från University of California, Berkeley, år 1987, skriven av David Patterson, Garth Gibson och Randy Katz: *A Case for Redundant Arrays of Inexpensive Disks (RAID)*. Linux skriver nivåerna med små bokstäver och utan mellanslag, raid1 eller raid5, vilket är hur dess md-drivrutin rapporterar dem.

Rapporten utgår från ett missförhållande. Processorer och minne blev allt snabbare, och sammanfattningen varnar för att vinsten går förlorad om in- och utmatningen inte hänger med. Den ställer Single Large Expensive Disk, SLED, vars kapacitet hade vuxit snabbt medan prestandan bara förbättrats måttligt, mot arrayer byggda av de billiga diskar som tillverkades för persondatorer, och väntar sig att sådana arrayer ska slå den stora disken ungefär tio gånger om i fråga om hastighet, tillförlitlighet, strömförbrukning och utrymme för tillväxt. Jämförelsen är konkret: en IBM 3380 för stordatorer, en Fujitsu Super Eagle för minidatorer och den 100 MB stora Conner CP 3100 för persondatorer.

Sedan kommer haken, i ett avsnitt som författarna själva betecknar som dåliga nyheter. Om diskar går sönder oberoende av varandra och i konstant takt, vilket rapporten påpekar att disktillverkarna förutsätter när de anger en medeltid till fel (MTTF), är arrayens MTTF lika med en disks MTTF delat med antalet diskar. Hundra CP 3100, var och en angiven till 30 000 timmar, ger 300 timmar, under två veckor; tusen ger 30 timmar. Rapporten drar en rak slutsats: utan feltolerans är stora arrayer av billiga diskar för opålitliga för att vara till någon nytta.

Svaret är redundans. Diskarna delas in i tillförlitlighetsgrupper, var och en med extra kontrolldiskar som rymmer redundant information. När en disk går sönder byts den ut, och innehållet byggs upp på den nya disken ur den informationen; tiden det tar är medeltiden till reparation (MTTR), och varma reservdiskar (hot standby spares), som kopplas in elektroniskt i samma stund som en disk dör, kortar den. Därefter lägger rapporten fram fem sätt att organisera diskarna och kallar vart och ett för en nivå. Nivå 1 speglar varje disk. Nivå 2 rättar fel med en Hammingkod, som minnessystem gjorde. Nivå 3 klarar sig med en enda kontrolldisk per grupp. Nivå 4 låter små läsningar och skrivningar gå till olika diskar samtidigt. Nivå 5 sprider data och kontrollinformation över alla diskar, så att ingen enskild kontrolldisk håller tillbaka skrivningarna.

Praktiken har byggt på listan. mdadm, verktyget som hanterar mjukvaru-RAID i Linux, godtar bland annat linear, raid0 (eller stripe), raid1, raid4, raid5, raid6 och raid10. Striping utan redundans, som numera kallas nivå 0, finns inte i indelningen från 1987, som börjar med speglade diskar. Rapporten lämnade öppet om en array byggs i hårdvara eller i mjukvara och kallade det enbart en fråga om kostnad och nytta. Linux md är mjukvaruvarianten; ett kontrollerkort som visar operativsystemet en enda stor disk är hårdvaruvarianten.

Använd begreppet i ett beslut

Använd ”Vad är RAID?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Säkerhetskopior som testas i stället för att tas för givna, och en skriftlig plan för den morgon då en server inte startar igen.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Automatiserade 100 kritiska säkerhetskopior — 100 automatiserade kritiska säkerhetskopior av databaserna

Automatiserade 100 kritiska säkerhetskopior Hör av er →

Varför det spelar roll

En disk slits ut, och på en server med en enda disk stoppar haveriet maskinen och lämnar data åt den säkerhetskopia som råkar finnas. Rapporten inleder sitt avsnitt om tillförlitlighet med just detta: diskar är så opålitliga att systemansvariga säkerhetskopierar allt varje natt. Med redundans blir en trasig disk ett reparationsjobb i stället för ett avbrott. Arrayen fortsätter på de diskar som finns kvar, i ett tillstånd som Linux kallar degraded (degraderad), och återgår till full styrka när en ersättare har lagts till och byggts upp.

Det skyddet varar bara så länge de återstående diskarna gör det. Rapportens kalkyler räknar data som förlorade när ytterligare en disk i samma grupp går sönder innan den första har reparerats, så längden på reparationsfönstret väger lika tungt som antalet kontrolldiskar. Med varma reservdiskar och korta reparationer blir siffrorna imponerande: rapporten räknar ut att en nivå 5-array med 1 000 diskar, i grupper om tio och med 5 procent reservdiskar, fortfarande skulle ha en MTTF på 90 år. På ett annat ställe varnar den för att en jordbävning eller en spänningstopp kan få diskarna i en array att gå sönder samtidigt, och dess formler förutsätter att det inte händer.

Valet av nivå är en avvägning mellan kapacitet, hastighet och kostnad. Rapportens egen slutsats är att speglade diskar passar transaktionsbehandling som använder högst hälften av lagringen, medan nivå 5 ser bäst ut när mer av kapaciteten behövs, för arbete på superdatorer eller för en blandning av sådant arbete och transaktionsbehandling.

När redundansen tar slut återstår räddning. Portföljposten om räddning av data på SD-kort, hårddiskar, SSD:er och RAID behandlar en trasig RAID-array som ett problem för sig, skilt från en död SSD eller en havererad disk med skivor, där tillvägagångssättet beror på mediet.

Så ser det ut i praktiken

Ta en server med en partition på var och en av två diskar, `/dev/sda1` och `/dev/sdb1`, som ska speglas till en datavolym. Med Linux md bygger tre kommandon arrayen, lägger ett filsystem på den och för in den i konfigurationen:

```
mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sda1 /dev/sdb1
mke2fs -t ext4 -L data /dev/md0
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

Det första skapar `/dev/md0`, en RAID1-array av de två partitionerna, och kärnan startar en omsynkronisering (resync) som räknar om redundansen, precis som efter varje nyskapande. Det andra formaterar arrayen exakt som det skulle formatera en enskild disk; filsystemet ser en enda enhet. Det tredje lägger till en rad som beskriver arrayen i mdadm:s konfigurationsfil, som ligger i `/etc/mdadm.conf` eller `/etc/mdadm/mdadm.conf`, och manualen ber att utdata granskas före användning, eftersom de kan innehålla detaljer som ingen vill ha där.

Tre filer visar hur arrayen mår:

```
cat /proc/mdstat
cat /sys/block/md0/md/degraded
cat /sys/block/md0/md/sync_action
```

`/proc/mdstat` listar alla aktiva md-enheter med deras tillstånd, och mdadm läser den för att hitta arrayer och följa återuppbyggnaden. Varje array har också en katalog under `/sys/block/`, och där innehåller `degraded` antalet medlemmar som saknas: 0 för en hel array, 1 när en disk är borta. `sync_action` visar `idle` när ingenting pågår, `resync` efter nyskapande eller en okontrollerad avstängning, `recover` medan en varm reservdisk tar över innehållet från en förlorad medlem, och `check` eller `repair` under en fullständig kontroll. Att skriva `check` till den startar den kontrollen, som läser varje block i arrayen:

```
echo check > /sys/block/md0/md/sync_action
cat /sys/block/md0/md/mismatch_cnt
```

Efteråt anger `mismatch_cnt` antalet sektorer som skrevs om, eller som skulle ha skrivits om. Kärndokumentationen påpekar att de flesta nivåer arbetar med sidor snarare än sektorer, så siffran kan bli större än antalet verkliga fel.

Någon måste också få höra om ett haveri. `mdadm --monitor` frågar arrayerna med jämna mellanrum och gör om det den märker till händelser, som den kan skicka med e-post till en adress eller lämna till ett program. På ett system som styrs av systemd pekar manualen ut `mdmonitor.service` som huvudvägen och `MAILADDR` i `mdadm.conf` som inställningen att fylla i för e-post. Bara fem händelser skickar e-post: `Fail`, `FailSpare`, `DegradedArray`, `SparesMissing` och `TestMessage`. Den sista finns för att bevisa att vägen fungerar:

```
MAILADDR ops@example.com
```

```
mdadm --monitor --scan --oneshot --test
mdadm --detail --test /dev/md0
```

Det första kommandot skickar ett testmeddelande för varje array det hittar och avslutar sedan. Det andra passar i ett skript: med `--test` är slutstatusen 0 för en frisk array, 1 när minst en enhet har gått sönder, 2 när så många har gått sönder att arrayen är oanvändbar, och 4 när mdadm stötte på ett fel under insamlingen av uppgifter om arrayen.

Arrayen täcker en enda händelse, att en disk dör, och ingenting annat. Portföljposten om automatiserad säkerhetskopiering och återställningstest beskriver en separat återställningsdesign: krypterade, deduplicerade restic-ögonblicksbilder via SFTP till lagring hos en annan leverantör, systemd-timerenheter och ett månatligt arbetsflöde som återställer en känd fil till ett tillfälligt arbetsutrymme och jämför dess byte. Det är ett test av återställning i designen, inte belägg för att det har körts. Implementeringsbeskrivningen daterad 2026-10-27 anger att timerenheterna var inaktiverade utan ett konfigurerat repo och redovisar ingen genomförd återställning i drift.

Var det går fel

En array byggs för en enda händelse, att en av dess diskar dör, och de flesta sätt den förlorar data på ligger alldeles bredvid den händelsen.

Arrayen tas för en säkerhetskopia. En spegel skriver varje ändring till båda diskarna samtidigt, och det gäller också en fil som raderats av misstag, en tabell som en misslyckad migrering tagit bort eller en volym som utpressningsprogram har krypterat. Då bär varje medlem på samma skada. Lösningen är att ha en separat säkerhetskopia utanför värden, med återställningar som testas enligt ett schema, så att arrayen skyddar mot hårdvarufel och säkerhetskopian mot allt annat.

En degraderad array går obemärkt förbi. En degraderad array fortsätter att leverera data, och ingenting i den ser annorlunda ut förrän nästa disk går sönder. mdadm skickar ingen e-post om det inte har en adress eller ett program: med `--scan` och ingetdera inställt övervakar det ingenting, enligt manualen. `DegradedArray` gäller dessutom bara arrayer som redan är degraderade när mdadm ser dem första gången; en disk som dör senare ger en `Fail` -händelse. Lösningen är att sätta `MAILADDR` , se till att `mdmonitor.service` körs och skicka ett meddelande med `--test` för att bevisa att e-posten kommer fram.

Den andra disken går sönder under återuppbyggnaden. En återuppbyggnad rekonstruerar den saknade disken ur alla de andra, och rapportens modell förlorar data precis när ytterligare en disk i gruppen går sönder inom reparationstiden. En död disk som får sitta kvar i en vecka sträcker ut fönstret till en vecka. Lösningen är att korta fönstret: med en varm reservdisk i arrayen, som mdadm kan lägga till redan när arrayen skapas med `--spare-devices`, med snabba byten av döda diskar och med en `check` som körs enligt ett schema, så att en disk med problem hittas medan arrayen fortfarande kan rätta dem.

Diskarna är inte oberoende. MTTF-kalkylen förutsätter att diskar går sönder var för sig, och rapporten nämner själv jordbävning och spänningstopp som fall där de inte gör det. Två diskar i samma chassi delar nätaggregat, kontroller, rum och byggnad. Lösningen är att behandla den beräknade siffran som en övre gräns och att täcka gemensamma orsaker med något utanför chassit, till exempel en kopia på en annan plats.

En RAID 5- eller 6-array är både smutsig och degraderad. Om maskinen stannar okontrollerat medan en medlem saknas går det inte att lita på pariteten, och vissa block kan inte byggas upp igen, så kärndokumentationen varnar för korruption som inte går att upptäcka, och md vägrar starta arrayen. Att tvinga igång den med `mdadm --assemble --force`, eller för ett rotfilsystem med modulparametern `start_dirty_degraded`, är att godta den risken. Lösningen är att välja en konsistenspolicy i förväg: md kan låta skrivningar gå via en separat journalenhet för nivå 4, 5 och 6, och för nivå 5 erbjuder den en partial parity log, som enligt dokumentationen täpper till det så kallade write hole. När tvång är den enda vägen in bör data betraktas som tvivelaktiga och kontrolleras mot en säkerhetskopia.

RAID 0 räknas som skydd. md räknar raid0- och linear-arrayer till dem som saknar redundant information, och övervakningen i mdadm rapporterar dem med `Wrong-Level`, eftersom de inte stöder någon av de operationer för haveri, reservdisk och omsynkronisering som den håller koll på. Det finns ingenting att bygga upp en saknad medlem ur. Lösningen är att använda nivå 0 bara för data som kan kastas eller återskapas, och att aldrig räkna den som RAID i den mening som rapporten från 1987 gav ordet.

En hårdvarukontroller döljer diskarna. Bakom ett kontrollerkort ser operativsystemet en enda logisk enhet, och en fråga om diskhälsa som riktas mot den enheten når kontrollern, inte diskarna. smartctl behöver till exempel `-d megaraid,N` för att nå disk N bakom en MegaRAID-kontroller, och enligt manualen

täcker det också Dell PERC-kort. Lösningen är att övervaka varje fysisk disk via kontrollerns eget gränssnitt, så att en medlem som börjar fallera upptäcks medan arrayen fortfarande är hel.

Läs mer

- A Case for Redundant Arrays of Inexpensive Disks (RAID) — sidan för den tekniska rapporten från Berkeley, med sammanfattning och PDF.
- UCB/CSD-87-391, hela rapporten (PDF) — MTTF-kalkylen, tillförlitlighetsgrupperna och de fem nivåerna.
- RAID arrays, Linuxkärnans dokumentation — md:s filer under `/sys`, smutsiga degraderade arrayer och konsistenspolicyer.
- mdadm(8) — att skapa arrayer, övervakningshändelser, e-post och slutstatus för `--detail --test`.
- smartctl(8), smartmontools — att nå diskar bakom RAID-kontroller med flaggan `-d`.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är RAID?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/raid/