Facultatea de Electronică, Telecomunicații și Tehnologia Informației, Politehnica București
Master Ingineria Informației și a Sistemelor de Calcul

Rebuilding RAID
(Redundant Array of Inexpensive Disks)

Student: Ciocoiu Raluca Lavinia
Profesor coordonator: Stăncescu Ștefan

București, 2013

Cuprins

Introducere

Tehnologia RAID este foarte răspandită în data center-urile moderne din întreaga lume pentru a îmbunătăți performanțele sistemelor și pentru a le face mai fiabile. Folosind mecanisme de redundanță, precum mirroring, codarea în functie de paritate și discurile hot-spare, RAID poate să reziste în cazul defectării discurilor și să treacă în mod automat în starea de recovery. Procesul de reconstrucție a RAID-urilor reface datele în cazul discurilor defecte pe noile discuri, iar în tot acest timp procesele de intrare – ieșire sunt în continuare servite. Totusi, procesul de reconstrucție trebuie să se facă intr-un timp cat mai scurt cu putință și să aibă un impact cat mai mic asupra sistemului și a utilizatorilor. Cum pierderile de informații și nefunctionarea sistemului nu sunt tolerate, proiectarea de sisteme cu capacitate mare de stocare rămane încă o provocare.

Noile cercetări în domeniul tehnologiilor de stocare au redus în mod considerabil costurile și au mărit capacitatea de stocare pe discuri, în timp ce alți parametrii, precum timpul de căutare și viteza de rotație a discurilor s-au îmbunătățit mult mai incet. Acestea fac ca procesul de reconstrucție a RAID-urilor să se găsească intr-o "fereastră de vulnerabilitate". Fereastra de vulnerabilitate, în care un al doilea disc poate să suporte pierderi de date, este afectată în mod negativ de cererile utilizatorului. Pe de altă parte, în centre mari de stocare, problemele parțiale sau totale legate de defectarea discurilor devine din ce în ce mai comună și mai frecventă din moment ce numărul de discuri crește foarte repede, iar rata de defectare individuală rămane aproape neschimbată. Aceste lucruri fac ca reconstrucția să devină un eveniment comun și de durată ridicată.

Desi multe abordari de reconstrucție on-line a RAID-urilor sunt eficiente, aproape toate neglijează eficiența cu care se gestionează memoria cache în timpul procesului de reconstrucție. Cel mai frecvent, sistemele moderne folosesc o memorie cache de dimensiune mare pentru a reduce numărul de accesari ale discurilor. Spre exemplu, sistemele de stocare EMC Symmetrix, IBM ESS și HP StorageWorks XP Disk Arrays folosesc drept cache o memorie NVRAM de dimensiuni intre 64GB și 256GB [1].

In ultimii ani s-au dezvoltat mai mulți algoritmi de gestionare a cache-ului, precum LRU, LRFU, LIRS, MQ, ARC, WOW, STOW, PA/PB_LRU, RIMAC și RACE [1]. Multe dintre acestea aveau drept scop îmbunătățirea performanțelor de stocare sau eficiența energetică, însă se ignoră performanța reconstrucției RAID-urilor. Pentru a face procesul de reconstrucție mai eficient, este foarte important să se optimizeze flow-ul de reconstrucție, să se facă intr-un mod secvential pe discuri.

Managementul de stocare în cache are un rol important în procesul de reconstrucție deoarece algoritmul de înlocuire a cache-ului de stocare ar trebui:

- în primul rand să acorde o prioritate mai mare blocurilor de date din discurile defecte prin menținerea lor în cache-ul de stocare din moment ce o cerere de citire va presupune accesarea discurilor bune
- în al doilea rand să evite blocurile de date din imediata vecinătate a blocurilor care necesită reconstrucția, reducand astfel intarzierile privind timpul de răspuns la cererile utilizatorului și timpul de căutare dintre zona de reconstrucție și zona de servire a informațiilor cerute de utilizator.

Din moment ce zona de reconstrucție nu este fixă, este foarte important să se poată face o separare a blocurilor cu probleme în mod dinamic.

Capitolul 1 – UDE (Undetected Disk Errors)

1.1 Prezentare generala UDE

In ciuda fiabilitații discurilor moderne, studii recente au arătat că o nouă clasă de erori, cea a erorilor nedetectate de discuri, este o adevărată provocare pentru sistemele de stocare. Spre deosebire de problemele apărute pe sectoare, UDE reprezintă problemele care nu sunt detectate de disc. Dacă sistemele care folosesc tehnologia RAID și-au dovedit eficiența în privința protectiei datelor în cazurile traditionale de defectare a discurilor, aceste noi modalități de corupere de date reprezintă o problemă majoră care nu a fost luată în considerare de tehnologia RAID. La randul lor, UDE provin din 2 tipuri de erori, erori de citire URE (Unetected Read Errors) și erori de scriere UWE (Undetected Write Errors). Manifestarea URE este trecătoare și este puțin probabil ca apariția lor să afecteze functionalitatea sistemului. Pe de altă parte, UWE sunt erori persistente care sunt detectabile în timpul unui proces de citire ce succedă un proces defectuos de scriere. Capacitățile de scalare din prezent au făcut ca UDE să fie erori comune și destul de des întalnite; astfel, studiile viitoare se axează tocmai pe rata de apariție a acestor erori și pe pe manifestarea lor din punct de vedere ale utilizatorului [2].

Există numeroase studii asupra ambelor erori de tip UDE, însă nici unul nu este complet. Avand în vedere că rata de apariție a UDE este scăzută, realizarea de teste în sistemele reale ar fi mult prea costisitoare și cel mai probabil ar necesita un număr mare de discuri pentru a putea evientia cu adevărat impactul intr-o perioadă de timp rezonabilă. Este deci necesar să existe toolu-uri care să modeleze intocmai astfel de sisteme. Un simplu model analitic ar putea da rezultate eronate cu privire la manifestarea UDE pentru un anumit volum de date. Spre exemplu, un bloc care a intampinat o eroare de tip UWE trebuie citit inainte ca eroare să fie privită ca o corupere de date si, prin urmare, probabilitatea ca sistemul să devină defectuos depinde de fluxul de date. Pentru a surprinde un asemenea comportament se pot face simulări pentru valori mai mici sau mai mari ale volumului de date.

Simulările de eficiență în cazul UDE pentru sisteme de dimensiuni mari sunt o adevărată provocare din cauza rigidității sistemului. Se spune despre un sistem că este rigid atunci când trebuie facute un număr mare de verificari pe durate de timp diferite pentru a se putea obține suficiente informații pentru modelarea sistemului [2]. Pentru a putea intelege în totalitate efectele și modul de manifestare al UDE în sistemele de stocare, trebuie să se aibă toate informațiile despre discuri, până la cele mai mici detalii, să se urmarească scrierea și citirea blocurilor pentru determinarea blocurilor care intampină UDE, propagarea UWE prin intermediul operatiilor normale din sistemul RAID (inclusiv în cazul proceselor de reconstrucție) și nu în ultimul rand eficiența tehnicilor de depistare.

1.2 Clasificare UDE

Din punctul de vedere al sistemelor de stocare, o mare problemă a fost ridicată de erorile pentru care tehnologia RAID nu ofera o protecție adecvată. Printr-o analiză a peste 1.53 milioane de discuri din sistemele de stocare pe o durată de 41 de luni a fost evidențiat faptul că există niște erori care au loc mai rar, dar care au ca manifestare

coruperea blocurilor de informații [2]. Unele dintre acestea sunt LSE (Latent Sector Errors), care reprezintă erorile raportate de disc la un moment ulterior apariției lor, iar altele sunt UDE(Undetected Disck Error), care reprezintă erorile nedepistate de sistem. Diferența dintre cele 2 anerior mentionate este că în cazul apariției unei erori de tip UDE, sistemul raportează succes chiar dacă s-a produs eronarea unui bloc. UDE au fost depistate de sistem prin intermediul unor mecanisme suplimentare de detecție, dar sunt nedepistabile în sistemele de producție clasice. Dacă nu este depistată, o eroare de tip UDE poate fi servită ca raspuns la o cerere venită de la utilizator.

UDE pot fi împărțite în 2 mari categorii, UWE (Undetected Write Errors) și URE (Undetected Read Errors), și pot exista mai mulți factori care să duca la apariția lor, spre exemplu probleme software, hardware, firmware. Tabelul din Figura 1.1 prezintă o serie de factori și modul lor de manifestare.

I/O Type	*UDE type*	*Manifestation*
Write (UWE)	Dropped Write	Stale data
	Near off-track write	Possible stale data on read
	Far off-track write	Stale data on intended track
		Corrupt data on written track
Read (URE)	Near off-track read	Possible stale data
	Far off-track read	Corrupt data

Fig 1.1 – Clasificare UDE și manifestari [2]

Erorile de scriere apar atunci când capul de scriere nu reuseste să transcrie informație peste datele deja inscrise pe pista. în acest caz, discul rămane în starea anterioară, ca și cum nu s-ar fi încercat niciodata scrierea pe el. Aceste scrieri incorecte se pot face în 2 moduri, în apropiere de pistă sau departe de pistă. In ambele cazuri capul de scriere nu este corect aliniat cu pista. în cazul scrierilor în apropierea pistei, data este scrisă în spațiile libere dintre 2 piste, adiacent pistei dorite. în cazul unei operatii de cititre, capul trebuie să se alinieze fie cu pista propriu-zisă, fie cu spațiul dintre piste, fapt ce duce la posibilitatea de a citi informații invechite. Scrierile departe de pistă se produc atunci când inscrierea informației are loc la o distanță destul de mare față de pista dorita, ducand la coruperea totală a informațiilor de pe altă pistă. Citiri succesive de pe pista care a fost scrisă în mod eronat vor duce la corupere de date, în timp ce citiri de pe pista care se dorea a fi scrisă vor furniza date invechite.

Este important de menționat faptul că nu toate erorile UWE trec nedetectate la nivelul utilizatorului. Un șir de scrieri succesive (corecte) pe pista vor inlatura eroarea, conducand la prevenirea erorilor de corupere de date. Ca și în cazul unei scrieri în apropierea pistei, rezultatele unor procese viitoare de citire vor fi fie informații corecte, fie informații invechite. Deși în cazul scrierilor departe de pistă rezultatele operatiilor de citire conduc în ambele cazuri la obținerea de informații invechite, și la coruperea informației pe pista nedorită, dacă aceata din urmă nu este folosită în mod curent nu se va obține o eroare. O eroare de tip UWE se poate manifesta și ca o serie de mai multe evenimente de corupere de informații în conditiile în care acceasi pistă este citită de mai multe ori inainte de a fi corectată printr-o serie de scrieri succesive [2].

1.3 Detecția UDE

Sistemele implementeaza rareori un mecanism de detecție adecvat pentru depistarea UDE. A fost demonstrat faptul că numai detecția de paritate nu este o metodă eficientă de protecție în cazul UDE; din acest motiv, au fost propuse mai multe solutii care să ofere un randament mai bun. Metode de detecție în cazul tehnologiei RAID au fost metodele de paritate cu supliment. în cadrul acestora, meta-datele sunt co-locate cu o parte sau cu toate blocurile asociate unei părți importante din totalul informațiilor și cu blocurile de paritate. în continuare se considera cazul unei metode de paritate cu supliment în care o sectiune dintr-un bloc este folosită peste memorarea unui număr de secvență. Acest număr de secvență este același pentru toate blocurile în timpul unei operatii de scriere. Erorile de tip UDE pot fi detectate în acest caz prin compararea numărului de secventa memorat în blocurile de paritate și de date. Numerele de secvență nu vor coincide atunci când UDE se manifestă ca eroare si se produce o coliziune în ceea ce privește numărul de secvență [2]. Coliziunile se produc cu urmatoarea probabilitate:

$$P(\mathrm{Seq(Parity)} = \mathrm{Seq(UDE)}) = \frac{1}{2^b}$$

unde b este numărul de biti alocat scrierii numărului de secventa. în cazul unei operatii de citire, numărul de secventa pentru fiecare bloc de date se obține în functie de paritate și comparat, permitand acestei tehnici să depisteze erorile de tip UDE cu costul dat de o citire suplimentara.

In continuare vom considera ca fiecare număr de secvență este stocat în blocul de informație și este reprezentat pe 8 biti. Numerele de secvenă sunt luate dintr-o distributie uniformă discretă. Pentru scrierea pe o pistă indepartată, sau pentru o scriere eșuată, blocurile originale pe care se doreste să se facă scrierea au numerele de secvență neschimbate. în cazul unei scrieri departe de pistă, blocurile vor căpăta numărul de secvență al blocurilor originale. în cazul scrierilor în apropierea pistei se vor obține 2 numere de secvență pentru un singur bloc, cate unul pentru fiecare pista. în cazul în care se întalnește o necorespondență între numerele de secvență, blocurile sunt marcate ca fiind necitibile [2] .

Capitolul 2 – Reconstrucția RAID

2.1 când se face reconstrucția RAID-urilor

RAID-urile redundante pot tolera defectarea unuia sau a mai multor discuri [1]. în general, RAID-urile functionează Intr-unul din cele 3 moduri prezentate mai jos:

1. modul normal (optimal), când nu există nici un disc defect
2. modul degraded, când există un disc cu probleme, dar cererile utilizatorului sunt în continuare servite
3. modul recovery, când se face reconstrucția discului cu probleme în background

Figura 2.1 prezentată mai jos ilustrează tranzacțiile dintre stări în cazul RAID5.

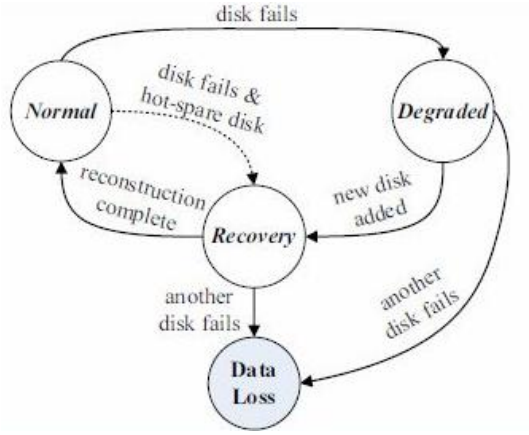


Fig 2.1 – Stări și tranziții pentru nivel RAID5 [1]

Cand un disc se defectează, RAID va opera în modul degraded. După adăugarea unui nou disc, se va trece în modul recovery și se vor inițializa thread-urile de reconstrucție a discului faulty pe noul disc. Dacă se folosește hot-spare disk, o defectare de disc va duce automat RAID-ul în modul recovery.

2.2 De ce se face reconstrucția RAID-urilor

Ar trebui să se pună mai multă atenție pe reconstrucția de RAID-uri din mai multe motive. In primul rand, reconstrucția de RAID-uri în data-center-uri mari nu este un eveniment rar, iar reconstrucția simultană va fi inevitabilă cat de curand. In modul normal, defectarea unui disc va determina RAID-ul să treacă în modul degraded sau în modul recovery, și defectarea unui alt disc pe durata celor 2 moduri va duce la pierderi de date. Defectarea altui disc implică 2 cazuri distincte: defectarea totală a discului și probleme pe un anumit sector. Studii recente au arătat ca rata anuală de inlocuire de discuri este cuprinsa intre 2 și 4%, mult mai mare decat cea specificata în cataloagele producătorilor, iar probabilitatea de defectare parțială, pe anumite sectoare, este de 3.45% din 1.53 milioane de discuri pe o perioada de 32 de luni cat a durat studiul. In plus, defectarea parțiala sau totală a unui disc expune o anumită zonă, ceea ce arată că după defectarea unui disc este foarte posibilă și defectarea unui al doilea disc [1].

In al doilea rand, procesul de reconstrucție se desfasoara pe o perioada destul de indelungată. Odata cu evolutia rapida a tehnologiei hard disk-urilor, capacitatea de stocare a unui singur disc creste cu până la 60% pe an, în timp ce latența se imbunateste cu aproximativ 10% anual. Prin urmare, există un decalaj în ceea ce priveste capacitatea discului și latenta cu care se face accesul la date, lucru care face ca timpul de reconstrucție să fie și mai mare. In plus, intr-un mod recovery tipic, cererile de reconstrucție interne și cererile de date venite de la utilizator împart resursele discurilor, ceea ce duce la un timp de reconstrucție de cateva zile. De exemplu, pentru reconstrucția în cazul unui nivel RAID5 cu 4 discuri SATA de dimensiune 10GB fiecare sunt necesare aproape 3 ore. Luand în calcul natura discurilor de dimensiune foare mare din zilele noastre, reconstrucția unui disc de 500GB de catre un server suprasolicitat poate dura și cateva zile [1].

Nu în ultimul rand, timpul mare de reconstrucție impactează în mod semnificativ sistemul de stocare. Perioada în care RAID-ul se află în modurile degraded sau recovery poartă denumirea de fereastră de vulnerabilitate. Fereastra de vulnerabilitate reprezintă perioada de timp în care aplicatiile utilizatorului au de suferit din cauza degradarii perfomantelor și a probabilitatii crescute de defectare a unui alt disc. Pierderile de date suferite în aceasta perioada sunt cu siguranță intolerabile pentru utilizator. In plus, cu cat este mai mare timpul de reconstrucție, cu atat crește probabilitatea de a pierde informații. Disponibilitatea unui system de stocare este definite ca raportul dintre MTFB (Mean Time Between Failure) și suma dintre acesta și MTTR (Mean Time to Recovery). O data cu cresterea MTTR, disponibilitatea sistemului scade [1].

Eșecul în reconstrucția RAID-urilor poate duce la pierderea de date. In cazuri extreme, când sunt asteptate defectari de discuri, utilizatorii pot crea backup-uri pentru sisteme. Din nefericire, studiile au arătat că 67% din datele de backup nu pot fi recuperate; în plus, în cazul celorlate 33 de procente, există date pierdute (de exemplu informațiile nou scrie intre momentul în care s-a realizat ultimul backup și momentul defectarii) din cauza mecanismelor de sincronizare moderne a backup-ului [1].

Din cele mai sus menționate concluzionăm că reconstrucția este foarte importantă pentru performanța, fiabilitatea și disponibilitatea sistemelor de stocare care folosesc tehnologia RAID. In realitate, un număr mare de utilizatori au pierdut date și au fost dezamagiti de modul de recuperare a datelor. Prin urmare, îmbunătățirea performantelor de reconstrucție a RAID-urilor este o provocare a carei solutie trebuie gasita intr-un timp cat mai scurt pentru cresterea fiabilitatii.

2.3 Cum se face reconstrucția RAID-urilor

Reconstrucția RAID-urilor se poate face în 2 moduri:

- reconstrucția offline (toate resursele sunt folosite pentru reconstrucție, fără a se raspunde cererilor utilizatorilor)
- reconstrucția online (se răspunde cererii utilizatorilor pe durata reconstrucției)

Reconstrucția offline este mai rapida decat cea on-line, însă, cum majoritatea sistemelor trebuie să ofere în permanență servicii utilizatorilor, se foloseste reconstrucția online.

Figura 2.2 prezintă modul de reconstrucție în cazul RAID5 în care există și cereri din partea utilizatorilor. Thread-urile de reconstrucție trimit cereri de reconstrucție

catre toate discurile active și reconstruiesc blocul de date cu probleme pe discul înlocuit. O cerere de citire a utilizatorului către blocul cu probleme este înlocuită cu o cerere de cititre către toate celelalte discurile active, pentru a se putea reface blocul cu probleme.

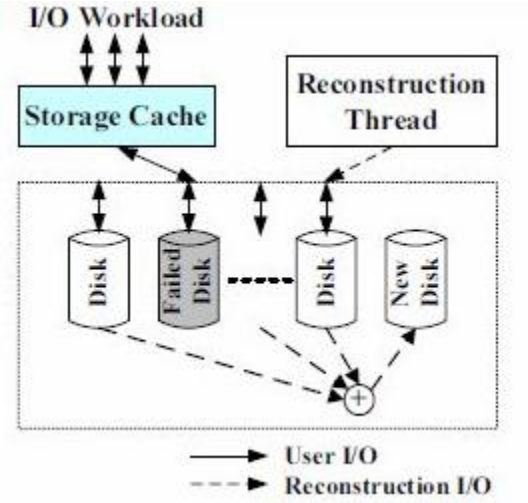


Fig 2.2 Reconstrucția de tip online pentru nivel RAID5 [1]

In mod normal, o cerere de scriere de date venită din partea utilizatorului se concretizează într-o infomatie de tip RMW (Read-Modify-Write) [1] pe unul din discuri și o informație de paritate de tip RMW pe celalalt disc. In timpul reconstrucției, când o cerere de scriere din partea utilizatorului ajunge la discul defect, RAID nu poate inscrie infomatia RMW pe discul în cauză, dar va trebui să calculeze paritatea și să o inscrie pe discul de paritate. Procesul preia toate informațiile de pe blocurile active din grupul de paritate, efectueaza o operatie XOR între acestea din urmă cu informația de paritate avută pentru a obține noua paritate, pe care o inscrie pe discul corespunzator.

2.4 Motivare

Deoarece discurile pot să ofere acces secvential de cateva ori mai rapid decat tipurile de acces aleator, orice algoritm de reconstrucție trebuie să aibă în vedere inerenta secventialitate a procesului de reconstrucție pentru a îmbunătăți performanta. Acesta este și motivul pentru care reconstrucția offline este mult mai rapida decat cea online și pentru care metoda de urmarire a capului de citire este nepractica. Aceasta metoda urmareste minimizarea timpului de pozitionare a capului de citire prin reconstruirea datelor în regiunea ce este accesata în acel moment de utilizatori. Ideea de baza este de a face reconstrucția să urmareasca ceea ce acceseaza utilizatorii. Din nefericire, deoarece volumul de munca al utilizatorului I/O conduce la necorelarea intre ele a capetelor discului, urmarirea capului de citire face ca firul de reconstrucție să invoce o unitate de reconstrucție dintr-un grup diferit de paritate și umple bufferele cu grupuri de paritate diferite, facandu-le foarte greu de eliberat, și conducand astfel catre un deadlock imediat al procesului de reconstrucție.

Sistemele tipice de stocare sunt echipate cu un NVRAM destul de mare care rolul de cache de stocare [1]. Din Figura 2.2 putem vedea ca volumul de munca al

utilizatorului I/O poate fi filtrat de cache-ul de stocare. Cu alte cuvinte, cache-ul de stocare are capabilitatea de a schimba accesul utilizatorului vazut de catre RAID. Pentru a face procesul de reconstrucție mai secvential, se specifica faptul ca accesul utlizatorului ar trebui să urmareasca reconstrucția cat mai mult posibil astfel permitand ca acel cache de stocare să aibă în vedere reconstrucția RAID.

Capitolul 3 - Managementul cache-ului "availability-aware"

Scopul de design al managementului cache-ului availability-aware este acela de a îmbunătăți performanțele reconstrucției RAID prin facerea procesului de reconstrucție mai secvential pe discuri și reducerea reconstrucției suplimentare provenită din comenzile I/O. Managementul cache-ului availability-aware este declanșat de firul de reconstrucție, astfel ca el să nu afecteze performantele normale ale I/O .

Ideea de bază a managementului cache-ului availability-aware este de a folosi cache-ul ca un modelator de trafic astfel încat acesta să fie cat se poate de aliniat cu procesul de reconstrucție I/O. Mai precis, există două aspecte: (1) Inlocuirea cu cost minim -> ce are în vedere reducerea penalitatilor în timpul reconstrucției. In alte cuvinte, blocurile de date ce se află departe de zona curentă de reconstrucție sau pe discul problematic vor trebui ținute în cache-ul de stocare, astfel reducand timpul de căutare pentru citire și overhead-ul în ceea ce privește discul problematic. Mai mult, blocurile de date ce aparțin discului problematic vor trebui să fie tinute în cache un timp mai îndelungat astfel reducandu-se penalitatea pentru un cache miss. (2) Eliminarea blocurilor "murdare" din jurul ariei de reconstrucție curenta pentru a reduce distanța de căutare intre aria de reconstrucție și aria vizata [1].

3.1 Inlocuirea cu cost minim

Timpul de acces al cache-ului este definit ca:

Access_Time=Hit_Time + Miss_Rate * Miss_Penalty

Din punctul de vedere al arhitecturii computerului există șase optimizări de bază a cache-ului ce sunt organizate în trei categorii: (1) reducerea ratei de insucces (miss_rate) (2) reducerea penalitatii de insucces (miss_penalty), și (3) reducerea timpului de acces pentru un succes (hit_time). Pentru cache-ul de stocare, a treia categorie nu mai este principalul scop deoarece timpul de succes în cache-ul de stocare este aproximativ fix; însă, principiile (1) și (2) sunt destul de importante deoarece accesul discului este mult mai costisitor decat accesul cache-ului. De exemplu, timpul de acces măsurat în memoria cache este de 50-250ns în comparație cu timpul de acces de ordinul a 5ms măsurat pe disc [1]. Inlocuirea cu cost minim vizeaza cel de-al doilea principiu, însă il ia în considerare și pe primul. Optimizarea înlocuirii în cache-ul de stocare pentru a mări performantele reconstrucției RAID include și urmatoarele doua aspecte.

In primul rand, inlocuirea cu cost minim oferă o prioritate mai mare blocurilor de date ce aparțin discului problematic și ce nu au fost reconstruite. De exemplu, cum se poate vedea în figura de mai jos, blocurile de date D2 și D3, ar trebui să fie evacuate inaintea blocurilor de date D8 și D9 ce apartin discului problematic. Cererile de citire pe

discurile ce funcționeaza normal și cele de pe discul problematic sunt procesate în mod diferit. O cerere de citire de pe discurile ce funcționeaza normal au nevoie doar de un singur acces de disc, în timp ce o cerere de citire pe discul problematic va necesita acces de citire pe toate celelalte discuri și apoi se va aplica o operatie XOR pe datele citite pentru a genera blocul de date de pe discul problematic. Dandu-se o matrice de discuri RAID5 ce este compus din 8 discuri, overhead-ul unei cereri de citire pe discul problematic este de 7 ori mai mare decat unul pe oricare dintre discurile ce funcționeazaă corect (excluzand overheadul indus de operatia XOR). Desi cererea de citire pe discul problematic este doar 1/8 din volumul de munca, el va mari cu 100% volumul de munca în toate celelalte discuri [1].

In al doilea rand, este foarte important și să se micsoreze căutarea (destul de frecventa și de lunga) dintre cererile I/O ale utilizatorilor și cererile de reconstrucție I/O, ce sunt dăunătoare pentru performantele reconstrucției RAID și chiar și pentru sistemul de stocare. Deoarece reconstrucția RAID este un proces ce implică tot sistemul de stocare, mentinerea în cache-ul de stocare a blocurilor de date ce se afla departe de zona curenta de reconstrucție poate să reduca semnificativ sansa de căutare ce dureaza foarte mult timp. De exemplu, blocurile de date D2 și D3 ar trebui să fie evacuate primele și blocurile de date Dx+1 și Dx+2 ar trebui să fie mentinute în cache-ul de stocare pentru mai mult timp, cum se poate vedea în Figura 3. în alte cuvinte, managementul cache-ului availability-aware mentine în cache-ul de stocare blocurile de date ce se afla la distanta mare de zona de reconstrucție pentru mai mult timp.

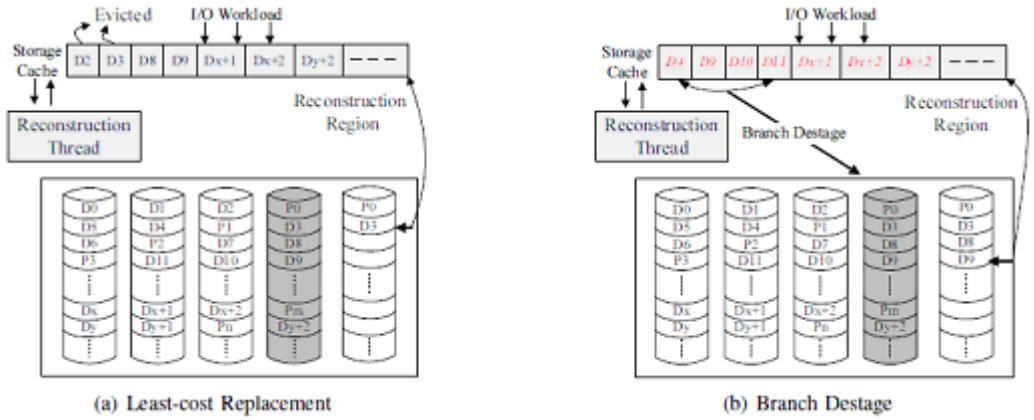


Fig 3.1 – Algoritmi pentru managementul cache-ului “availability-aware” [1]

3.2 Eliminarea blocurilor “murdare”

Mutarea blocurilor “murdare” de date dintr-un dispozitiv de stocare de invel înalt într-unul de nivel jos definește această operație. In designul managementului cache-ului availability-aware, această operație denotă faptul că datele “murdare” din cache-ul de stocare sunt trimise către RAID când nivelul de ocupare al cache-ului de scriere ajunge la un nivel foarte mare. Secvențierea fluxului de reconstrucție pe discuri poate îmbunătăți performanța în mod semnificativ. Operația de eliminare a blocurilor “murdare” poate de asemenea să îmbunătătească performanța de stocare în doua moduri: (1) este de dorit să se accelereze rata de eliminare deoarece se pot bloca cererile de acces ale utilizatorilor și

(2) ordinea de eliminiare ar trebui să fie facută într-un mod secvential pe discuri, astfel folosindu-ne de avantajul accesului de tip secvential al discurilor.

Pentru a satisface cele doua deziderate, aceastaă metodă elimină doar blocurile "murdare" din zona curentă de reconstrucție, reducand astfel timpii mari de căutare dintre zonele îndepărtate și aria de reconstrucție. Mai mult, se organizează o ordine de eliminare făcand procesul mai rapid și mai eficient. De exemplu, blocurile de date D4, D9, D10 și D11 vor trebui eliminate și menținute în această ordine pe discuri, precum se poate vedea în Figura 3b. Blocurile de date Dx+1, Dx+2 și Dy+2 vor trebui să fie mentinute în cache-ul de stocare până când procesul de reconstrucție ajunge în acea zona.

Capitolul 4 - Studii de caz

4.1 Evaluarea performantei reconstrucției RAID folosind aplicatia Sharper

4.1.1 Premisele experimentului

Platforma în cazul experimentului este echipată cu un procesor Intel Xeon 3.0GHz și 1GB DDR RAM; în sistem există un controler 3Ware 9650 SATA ce are 8 discuri WD2500YD SATA. Fiecare disc este partiționat și va avea 10GB – acest aspect nu afectează concluziile experimentului. In plus, un disc separat IDE este folosit pentru a stoca sistemul de operare (Linux kernel 2.6.21.1) și software-uri aditionale (MD, mdadm și RAIDmeter) [1].

Trasele folosite în experimente sunt obținute de Storage Performance Council și HP Storage Research Lab. Cele doua trase au fost colectate de aplicatii OLTP ce funcționează în instituții financiare de anvergură, iar trasa Cell99 reprezintă un volum tipic de munca I/O al unui computer. Tabelul I arată caracteristicile celor trei trase. Evaluarea performantelor este bazată pe RAIDmeter (un tool la nivel de bloc ce analizeaza trasele și evalueaza timpii de raspuns I/O al dispozitivului de stocare).

S-au implementat doi algoritmi de stocare cache, LRU (Inlocuirea cu cost minim) și Sharper, în RAIDmeter. In prezent, modulul Sharper include doar schema de eliminare a blocurilor "murdare". Aria de reconstrucție este luată în mod dinamic din interfața "/sys/block/md0/md/sync_completed"[1]. In sistemele reale, algoritmul LRU sau diferitele sale variante se folosesc la scară largă. O descriere schematică a arhitecturii software experimentale se poate regăsi în Figura 4.2.

Trace	Trace Characteristics		
	Read Ratio	IOPS	Aver. Req. Size(KB)
Fin1	32.82%	69	6.2
Fin2	82.39%	125	2.2
Cello99	23.9%	52	8.9

Fig 4.1 –Caracteristicile trace-urilor [1]

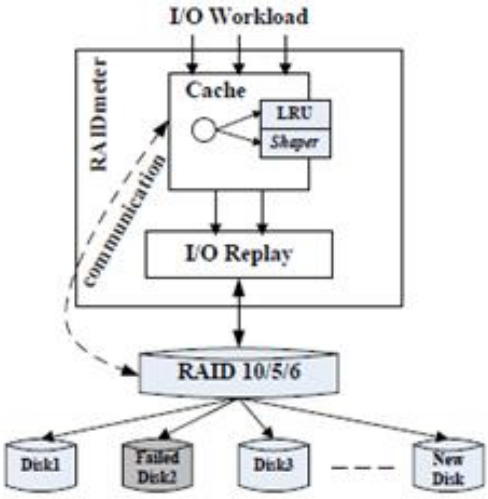


Fig 4.2 – Software-ul sistemului experimental [1]

Se evalueaza performanTa de reconstrucție în ceea ce privește timpul de răspuns I/O și timpul de reconstrucție ; în fiecare rundă de evaluare, se rulează 10 secunde pentru a pregăti bufferul inainte de a incepe procesul de reconstrucție. Astfel eficiența de reconstrucție nu va fi sensibilă la mărimea bufferului [1].

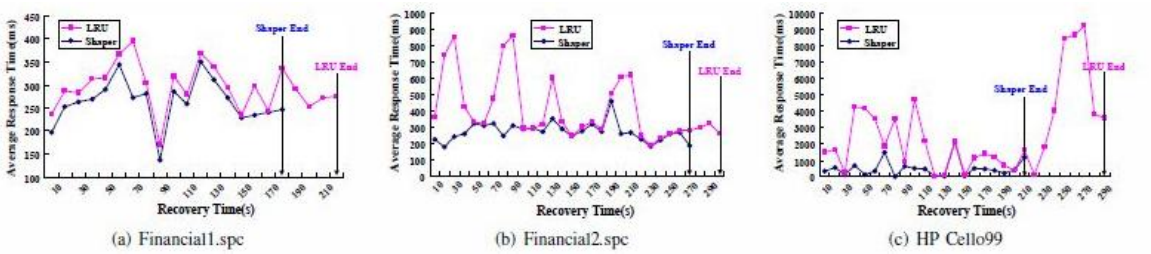
4.1.2 Evaluarile traselor

Tabelul din Figura 4.3 indica media timpului de raspuns I/O și timpii de reconstrucție LRU și Sharper masurati și indica eficacitatea Sharper în ceea ce priveste reducerea timpului mediu de raspuns I/O în timpul procesului de recuperare, dar și timpul de reconstrucție simultan. în toate trasele și în conditiile numerelor diferite de discuri în experimente, Sharper se dovedeste a fi mai eficient decat LRU în ceea ce priveste timpul de raspuns dar și timpul de reconstrucție cu până la 76.7% și 27.67%. Din Tabelul II, se poate de asemenea vedea ca procesul reconstrucției RAID mareste semnificativ timpul mediu de raspuns. Motivul este faptul ca software-ul RAID în Linux 2.6.21.1 favorizeaza fiabilitatea în defavoarea performantei. Majoritatea resurselor discurilor sunt ocupate de procesul de reconstrucție. Exista foarte putin loc lasat pentru reducerea timpului de reconstrucție. Astfel, îmbunătățirea timpului mediu de raspuns este mai mare decat îmbunătățirea timpului de reconstrucție.

Figura 4.4 ilustrează îmbunătățirea semnificativă a performanței Sharper în comparativ cu LRU. Sharper exploatează în mod eficient secvențialitatea reconstrucției prin eliminarea preferențială a blocurilor de date “murdare” din jurul ariei de reconstrucție. Mișcările de mare distanță a capului de citire sunt evitate și reconstrucția secvențiala este garantată, astfel reducand timpul de reconstrucție. Mai mult, deoarece cererile I/O pot fi onorate imediat fără a implica timpi mari de poziționare, timpul mediu de răspuns poate fi de asemenea redus. Din tabelul din Figura 4.3 și din Figura 4.4 putem spune ca îmbunătățirea performanței folosind Sharper în trasa HP Cello99 este cea mai mare.

Traces	Number of Disks	Average User Response Time(ms)				Reconstruction Time(s)		
		Normal	During reconstruction			LRU	Shaper	Improved
			LRU	Shaper	Improved			
Fin1	4	35.31	301.42	268.91	10.79%	220	180	18.18%
Fin2		27.02	405.42	272.69	32.74%	300	270	10%
Cello99		107.8	3531	822.57	76.7%	285.9	206.8	27.67%
Fin1	8	N/A	288.73	267.62	7.31%	208.52	180.7	13.34%
Fin2		N/A	301.58	249.3	17.34%	260.85	248.95	4.56%
Cello99		N/A	2021.57	619.23	69.37%	205.24	185.5	9.62%

Fig 4.3 – Comparatie LRU și Sharper [1]



(a) Financial1.spc (b) Financial2.spc (c) HP Cello99

Fig 4.4 – Comparatie intre timpii de raspuns în cazurile Sharper și LRU în timpul reconstrucției [1]

4.1.3 Discutie

Managementul cache-ului availability-aware transformă modelul simplu binar de eșec ce se regasește în majoritatea sistemelor de stocare în managementul cache-ului de stocare și mărește disponibilitatea sistemelor structurate RAID prin îmbunătățirea performanței de reconstrucție RAID. S-a demonstrat eficiența Sharper în ceea ce privește reconstrucția RAID. Ceea ce se poate deduce prin implementarea și evaluarea Sharper este [1]:

(1) Managementul stocării orientată spre performanța a cache-ului nu este cea mai bună optimizare pentru îmbunătățirea disponibilitatii sistemelor de stocare. El vizează îmbunătățirea ratei de succes a stocarii cache dar nu ia în considerare îmbunătățirea performantei reconstrucției RAID și deci mișcările capului de citire intre zonele necesare pentru deservirea cererilor I/O și zona de reconstrucție vor fi dese și foarte costisitoare ca și timp. Astfel, atat timpul de raspuns cat și timpul de reconstrucție vor fi afectate și disponibilitatea sistemului de stocare este redusă.

(2) Aducerea preemtiva în cache a anumitor blocuri de memorie nu este folosită în design desi este o tehnică importantă în vederea îmbunătățirii performanțelor sistemului de stocare. Aducerea preemtiva a blocurilor de memorie în cache poate reduce cererile suplimentare I/O astfel furnizand mai multe resurse către procesul de reconstrucție RAID în momentul în care un disc incepe să aibă

probleme. Pe de altă parte, aducerea preemtivă a unor blocuri ce se dovedesc a nu fi utile poate cauza multe cereri I/O paguboase ce afectează performanțele de reconstrucție RAID. Intr-un cuvant , aducerea preemtivă a blocurilor în cache este o sabie cu două tăișuri. Evaluarea eficienței diferiților algoritmi de aducere preemtivă a blocurilor în ceea ce privește reconstrucția RAID este un subiect deschis

(3) In acest moment, se consideră doar managementul cache-ului availability-aware pentru matricile de discuri bazate pe paritate. Pentru oglindirea matricilor de discuri precum RAID1 și RAID10, algoritmul de management al cache-ului ar trebui să fie diferit. Deoarece performanța matricii de discuri oglindite degradate poate fi mai bună decat aceea în modul normal, aplicarea managementului cache-ului availability-aware în oglindirea matricilor de discuri poate fi considerată. Evaluări ale performanței diverșilor algoritmi cache în ceea ce privește oglindirea matricilor de discuri în timpul reconstrucției sunt incă în lucru.

4.2 Reconstrucția RAID pe masini cu Windows Server 2003

4.2.1 Prezentare generala a masinilor

In continuare vom prezenta structura mașinilor pe care se va face reconstrucția. Acestea sunt alcătuite din 2 CP-uri (pentru redundanță) care comunică între ele folosind protocolul GCP. La un anumit moment, numai unul din cele 2 CP-uri este activ, iar celălalt se află în starea Stand-by-Hot pentru a prelua aproape instantaneu funcțiile CP-ului activ în cazul în care acesta se defectează. Ambele CP-uri sunt conectate la APG, care are rolul de a pointa catre AP-ul activ. Ca și în cazul CP-urilor, avem 2 AP-uri tot pentru redundanță; la un anumit moment, numai unul din AP-uri este activ, iar celalalt se află în starea Stand-by-Cold pentru a prelua funcțiile AP-ului activ în caz de defectarea. Timpul în care CP-ul pasiv preia functiile geamănului sau este mult mai mic decat cel din cazul AP-ului pasiv, chiar dacă este vorba doar de cateva secunde. Cele 2 AP-uri sunt elementele care au discuri partajate.

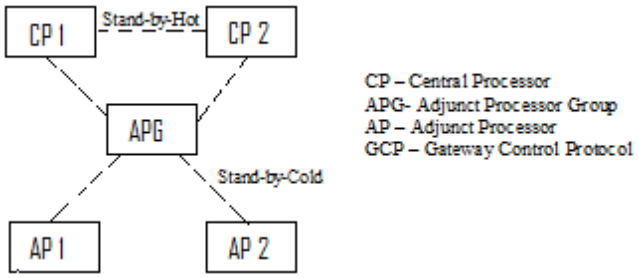


Figura 4.5 – Arhitectura masinii cu Windows Server 2003 pe care s-a facut reconstrucția

4.2.2 Discuri de date. Alocarea discurilor

Masinile (AP-urile) au 3 hard disk-uri care sunt oglindite intre nodul A(AP1) și nodul B (AP2) . Fiecare HD are o capacitate de 18GB și deci un nod are o capacitate maxima de stocare a datelor de 54GB. Nodul activ va avea în posesie discurile de date și este și normal ca numai acesta să poate face scriere sau citire pe discuri. Pe discuri sunt stocate date foarte importante, precum informații de billing și statistica, dar și fisierele de sistem sunt stocate tot aici. Din acest motiv este absolut necesar ca informația de pe discuri să aibă o anumita redundanta, lucru asigurat de tehnologia RAID.

In figura de mai jos este prezentata o modalitate de partitionare a discurilor.

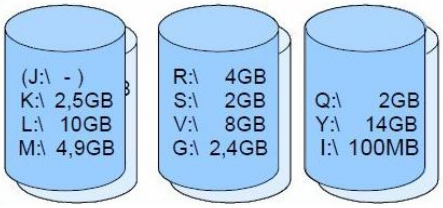


Fig 4.6 - Partiționarea discurilor pe un singur nod [3]

J	Data disk 1	Empty (Only on nodes upgraded from NT)
K	Data disk 1	APM, GOH
L	Data disk 1	CP file system (CP dumps etc)
M	Data disk 1	APIO, APM and AP Backups
R	Data disk 2	STS Counter database
S	Data disk 2	STS Output files
V	Data disk 2	APZ 212 40
G	Data disk 2	General FTP
Q	Data disk 3	ACS Message Store for RTR
Y	Data disk 3	RTR transfer queue
I	Data disk 3	Cluster quorum

Fig 4.7 - Alocarea datelor pe discuri [3]

4.2.3 Hardware RAID pentru un nod

RAID-ul este o colecție de drive-uri care per ansamblu se comportă ca un sistem de stocare unitar, care poate să tolereze defectarea unui disc fără a pierde date, și care are posibilitatea de a opera independent de celelate. Figura de mai jos prezintă configurația RAID din DPT, aplicație folosită pentru vizualizarea stării discurilor.

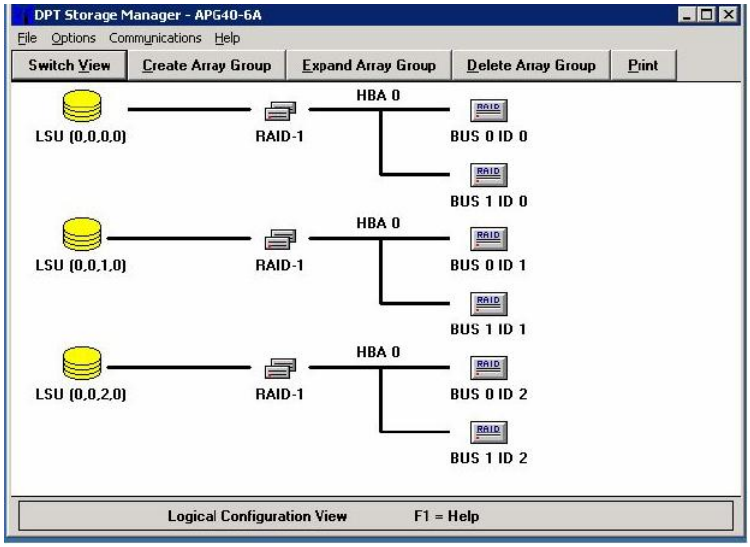


Fig 4.8 –Vizualizare RAID din aplicatia DPT [3]

Pentru fiecare nod există 3 discuri de date unde sunt stocate spre exemplu back-up-urile de CP și informații de charging. Discul 1 de pe nodul A este oglindit cu discul 1 de pe nodul B și așa mai departe. Foarte important de reținut este faptul că numai nodul activ poate efectua operații pe discuri.

4.2.4 Hardware RAID pentru APG

APG-ul este format din 2 noduri identice, deci este și normal ca acesta să aibă dublul dimensiunilor menționate anterior pentru un singur nod. In figura de mai jos este prezentată fereastra de GUI prin care se poate verifica statusul discurilor (se observă existența a 3 discuri logice și a 6 discuri fizice). Culoarea verde indică statusul optim al acestora. Culoarea roșie marchează defectarea unuia sau a mai multor discuri.

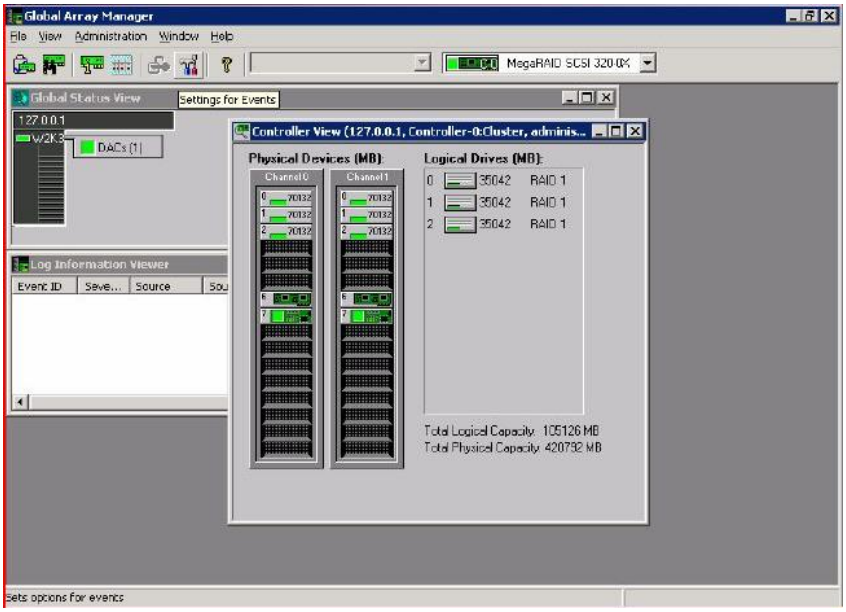


Fig 9 – Partajarea discurilor

4.2.5 Procedeul de reconstrucție al RAID-urilor

In cazul apariției unei probleme pe unul din discuri, unul din cele 2 AP-uri va semnala problema prin generarea unei alarme "AP Fault with Mirrored Disks not Redundant". Procedeul de reconstrucție se bazează pe o constructie de tip online, iar instrucțiunile, precum si output-urile, sunt preluate din librăria ALEX [4] specifică echipamentelor. Analiza problemei va porni începand cu verificarea stării fizice a discurilor. Pentru aceasta comanda *raidutil –L physical* a fost data. Mai jos este prezentat rezultatul acestei comenzi; d0bxtyd0 este numele discului, se mai poate afla fabricantul (în cazul de față, Fujitsu), precum și dimensiunea acestuia (17522MB). "Optimal" este starea normala a discurilor.

```
raidutil -L physical
------------------------------------------------------------------------

d0b0t0d0 Disk Drive (DASD) FUJITSU MAP3367NP 17522MB Optimal

d0b0t1d0 Disk Drive (DASD) FUJITSU MAP3367NP 17522MB Optimal

d0b0t2d0 Disk Drive (DASD) FUJITSU MAP3367NP 17522MB Optimal

d0b1t0d0 Disk Drive (DASD) FUJITSU MAP3367NP 17522MB Optimal

d0b1t1d0 Disk Drive (DASD) FUJITSU MAP3367NP 17522MB Optimal

d0b1t2d0 Disk Drive (DASD) FUJITSU MAP3367NP 17522MB Optimal
```

După analiza fizică, se va face și analiza logica a discurilor. Pentru aceasta se foloseste comanda *raidutil –L logical.* Starea normală a acestora este "Optimal"; apariția starii "Degraded" în cazul blocului d0b0t2d0 evidentiază o problemă de mirroring.

```
raidutil -L logical
------------------------------------------------------------------------

d0b0t0d0 RAID 1 (Mirrored) DPT RAID-1 17522MB Optimal

d0b0t1d0 RAID 1 (Mirrored) DPT RAID-1 17522MB Optimal

d0b0t2d0 RAID 1 (Mirrored) DPT RAID-1 17522MB Degraded
```

Inainte de a se trece la reconstrucția RAID-ului degraded, mai trebuie verificat faptul că toate resursele cluster-ului sunt în starea "Online". Aceasta se face folosind comanda *cluster res*. Abia apoi se poate trece la reconstrucția blocului, folosind comanda

```
raidutil -a rebuild d#b#t#d#[,d#b#t#d#]
```

In cazul de față, problema este pe blocul d0b0t2d0, deci comanda este *raidutil –a rebuild d0b0t2d0.* Dacă exista probleme pe mai multe blocuri, atunci se va folosi aceeaşi instrucțiune, iar blocurile vor fi separate prin virgulă.

```
C:\♥>cluster res
Listing status for all available resources:

Resource             Group                Node            Status
-------------------- -------------------- --------------- ------
Disks J: K: L: M:    Disk Group           APG40-2B        Online
Diskeeper            Disk Group           APG40-2B        Online
Share J              Disk Group           APG40-2B        Online
Share K              Disk Group           APG40-2B        Online
Images               Disk Group           APG40-2B        Online
Share LOGS           Disk Group           APG40-2B        Online
Share FMS            Disk Group           APG40-2B        Online
Share MCS            Disk Group           APG40-2B        Online
Disks R: S: X:       Disk Group           APG40-2B        Online
Share R              Disk Group           APG40-2B        Online
Share S              Disk Group           APG40-2B        Online
Share X              Disk Group           APG40-2B        Online
Disk Y:              Disk Group           APG40-2B        Online
Share Y              Disk Group           APG40-2B        Online
ACS_ALOG_Main        Disk Group           APG40-2B        Online
ACS_PRC_IspLogger    Disk Group           APG40-2B        Online
AES_CDH_server       Disk Group           APG40-2B        Online
AES_AFP_server       Disk Group           APG40-2B        Online
AES_DFO_server       Disk Group           APG40-2B        Online
APIO_TFTP_Server     Disk Group           APG40-2B        Online
APM_DHCP_Server      Disk Group           APG40-2B        Online
FMS_CPF_server       Disk Group           APG40-2B        Online
CPS_BUAP_parmgr      Disk Group           APG40-2B        Online
CPS_BUAP_filemgr     Disk Group           APG40-2B        Online
CPS_BUAP_loader      Disk Group           APG40-2B        Online
CPS_BUFTPD           Disk Group           APG40-2B        Online
CPS_BUTFTPD_butftpd  Disk Group           APG40-2B        Online
```

Reconstrucția unui singur bloc poate dura intre 4 și 6 ore.

Concluzii

Din moment ce discurile pot oferi un acces secvențial mult mai rapid decât accesul aleator, orice algortim de reconstrucție trebuie să țină cont de acest acces secvențial pentru a îmbunătăți performanțele reconstrucției. Acesta este și motivul pentru care reconstrucția offline este mult mai rapidă decât cea online.

Din cauza ratei ridicate de defectare a discurilor, reconstrucția RAID-urilor devine o operație inevitabilă în sistemele de stocare și poate influența funcționalitatea sistemului. Se dorește folosirea unui sistem de management de cache care să știe atunci când se face o reconstrucție de RAID, îmbunătățind astfel performanțele procesului de reconstrucție și fiabilitatea sistemului.

Datorită ratei mari de eșec ale discurilor, reconstrucția RAID devine o sarcină inevitabilă în sistemele de stocare de anvergură și poate afecta disponibilitatea sistemului. Managementul cache-ului availability-aware se bazează pe managementul cache-ului de stocare pentru a face cache-ul conștient de procesul de reconstrucție RAID, astfel îmbunătățind performanța reconstrucției RAID și mărind disponibilitatea sistemelor de stocare structurate în modul RAID. Prototipul implementat folosind Sharper și rezultatele experimentului arată faptul ca managementul cache-ului availability-aware are performanțe mai bune comparandu-l cu LRU atat în ceea ce privește timpul de răspuns cat și timpul de reconstrucție.

Bibliografie

[1] Suzhen Wu, Bo Mao, Dan Feng and Jianxi Chen, "Availability-aware Cache Management with Improved RAID Reconstrucţion Performance", IEEE International Conference on Computational Science and Engineering, pp 229 – 235, 2010

[2] Eric W. D. Rozier, Wendy Belluomini, Veera Deenadhayalan, JimHafner, KK Rao, Pin Zhou, " Evaluating the Impact of Undetected Disk Errors în RAID Systems" , IBM Almaden Research Center, University of Illinois Department of Computer Science, pp 83-84

[3] "APG40 Operation and Maintenance (Windows 2003)" , LZT 123 8452 R1A, 2006

[4] Libraria ALEX