Universitatea *Transilvania* Brașov
Facultatea de Inginerie Electrică și Știința Calculatoarelor
Departamentul de Electronică și Calculatoare

Sisteme de operare

– curs 13 –

2010.06.05

ș.l. dr. ing. Kertész Csaba-Zoltán

4.3.5. Fiabilitatea sistemului de fișiere

- distrugerea unui sistem de fișiere este mai dezastruoasă decât distrugerea componentelor unui calculator
  - componentele sau chiar calculatorul pot fi înlocuite rapid, informația pierdută poate fi refăcută în timp mult mai lung și cu costuri mult mai mari sau poate fi chiar imposibil de refăcută

Bad Block Management

- foarte multe discuri au bad blockuri chiar de la fabricație
- în timpul utilizării numărul bad blockurilor va crește exponențial
- soluții pentru managementul bad blockurilor
  - hardware: sa alocă un sector de pe disc pentru a memora lista bad blockurilor, controlerul de disc va evita utilizarea bad blockurilor pe baza acestei liste
  - software: sistemul de fișiere va elimina bad blockurile din lista blocurilor libere, astfel nu vor mai putea fi folosite

Backup-ul

- este important ca backup-ul pentru un sistem să fie realizat frecvent
- backup-ul în general este realizat pe benzi magnetice sau suport optic => operație lentă
- backup-ul poate fi realizat în 2 moduri
  - integral
    - tot conținutul sistemului de fișiere este salvat
  - incremental
    - diferența față de ultima salvare este salvată
- tar (Tape ARchiver)

Consistența unui sistem de fișiere

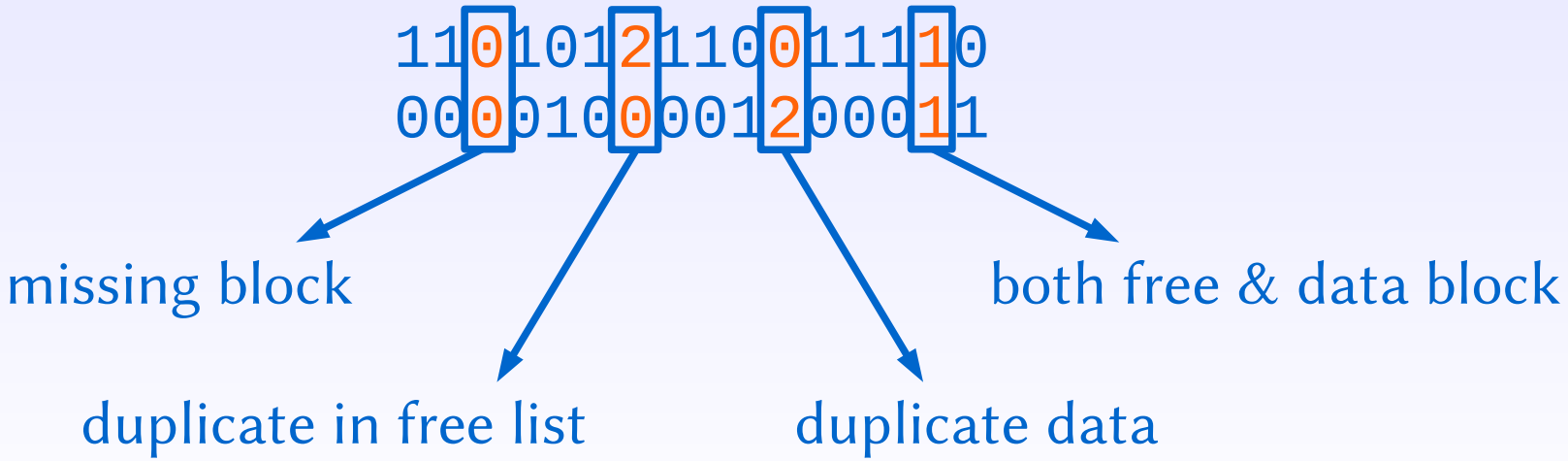
- în multe sisteme blocurile sunt citite, modificate și după un interval de timp vor fi scrise pe disc
- dacă sistemul cade înainte ca toate blocurile modificate să fie scrise pe disc, sistemul de fișiere poate rămâne într-o stare inconsistentă
- pentru a rezolva problema inconsistenței majoritatea sistemelor au un utilitar care verifică dacă sistemul de fișiere este consistent

Consistența blocurilor

- programul construiește un tabel cu 2 contoare inițializate la 0 pentru fiecare bloc
  - primul contor memorează de câte ori este prezent un bloc într-un fișier
  - al doilea contor memorează cât de des este prezent blocul în lista blocurilor libere
- programul va citi toate i-nodurile, plecând de la un i-nod construiește lista blocurilor utilizate în fișier și se incrementează contoarele asociate acestor blocuri

Consistența blocurilor

- programul examinează lista blocurilor libere și va incrementa cel de al doilea contor pentru fiecare bloc din listă
- sistemul de fișiere este consistent dacă fiecare bloc va avea un 1 numai într-unul din tabele



Consistența blocurilor

- missing block
  - blocul nu se regăsește nici în lista blocurilor ocupate nici în lista blocurilor libere
  - blocul se adaugă la lista blocurilor libere
- duplicate in free list
  - blocul apare de mai multe ori în lista blocurilor libere
  - se reconstruiește lista blocurilor libere

Consistența blocurilor

- both free & data block
  - blocul este utilizat de un fișier, dar apare și în lista blocurilor libere
  - blocul se va elimina din lista blocurilor libere
- duplicate data
  - același bloc este utilizat de 2 fișiere
  - se copiază conținutul blocului într-un bloc nou și blocul nou se va aloca unuia din fișiere în locul blocului duplicat

Consistența directoarelor

- se vor utiliza contoare similar cu metoda pentru blocuri, dar pentru fișiere
- se pornește de la rădăcină și se parcurge arborele de directoare
- pentru fiecare fișier din fiecare director se va incrementa contorul pentru i-nodul respectiv
- se compară contorul cu numărul de linkuri stocat în i-nod
- cele două numere trebuie să fie egale pentru un sistem de fișiere consistent

4.3.6. Performanțele sistemelor de fișiere

- multe sisteme de fișiere au fost proiectate pentru a reduce numărul de accese la disk
- tehnici utilizate:
  - block cache
  - buffer cache
- **cache**: colecție de blocuri care aparțin discului dar sunt menționate în memorie pentru a crește performanța sistemului de fișiere (acces mai rapid la date)

Algoritm pentru managementul unui block cache

- se verifică toate apelurile de citire a datelor de pe disc pentru a vedea dacă blocul necesar este în cache
- dacă aceasta se află în cache nu mai este necesar accesul la disc
- dacă nu, blocul va trebui adus de pe disc în cache
- dacă cache e plin trebuie eliminat un bloc: se folosește un algoritm de paginare (FIFO, Second Chance, LRU)

Algoritm pentru managementul unui block cache

- de obicei se folosește un algoritm LRU modificat în care blocurile care sunt esențiale pentru consistența sistemului de fișiere (toate celelalte blocuri în afara celor de date) și care au fost modificate vor trebui scrise pe disc imediat, indiferent de poziția lor în lista LRU
- se poate întâmpla ca pe o perioadă lungă de timp nu mai sunt necesare blocuri de pe disc iar sistemul cade => blocurile din cache se pierd
  - în UNIX există un daemon care apelează SYNC în mod regulat (la fiecare 30 secunde)

Tehnici pentru reducerea mișcării capului harddiscului

- blocurile ce se vor accesa în secvență vor trebui să fie apropiate ca localizare fizică pe disc (preferabil în același cilindru)
- se poate realiza o grupare a blocurilor din lista blocurilor libere => grupuri de blocuri consecutive
- o deficiență a sistemului cu i-noduri este că accesul la un fișier necesită două accese pe disc (unul pentru a citi i-nodul și unul pentru blocul respectiv)

Disk Arm Scheduling Algorithms

- timpul necesar pentru a citi sau scrie un bloc de pe disc este determinat de 3 factori
  - seek time (timpul necesar pentru a plasa brațul pe cilindrul corespunzător)
  - rotational delay (timpul necesar pentru a roti sectorul corespunzător sub cap)
  - transfer time (timpul necesar pentru transferul blocului)
- din acești factori seek time este cel mai dominant => reducerea acestui timp va conduce la îmbunătățirea performanțelor

First Come, First Served (FCFS)

- cea mai simplă strategie
- nu prea poate face nimic pentru a optimiza seek time
- discul menține o tabelă indexată la numărul cilindrilor cu toate cererile apărute pentru fiecare cilindru
- cererile sunt plasate într-o listă înlănțuita
- capătul listei va fi indexat în tabelă
- ex.: cereri pentru cilindrii: 11,1,36,16,34,9,12

seek time total – 111 cilindri

Shortest Seek First (SSF)

- o îmbunătățire a algoritmului FCFS
- se va alege din lista cererilor cilindrul cel mai apropiat de cilindrul curent
- ex. (precedent)

  seek time total – 61 cilindri

- SSF are o problemă majoră: dacă mai apar cereri între timp care sunt mai aproape de cilindrul curent, o cerere mai vechi apărută va fi întârziată

SSF

- pentru un disc având multe cereri, brațul tinde să rămână în mijlocul discului
- cererile pentru cilindri aflați la extremități vor avea un timp de răspuns foarte prost

Algoritmul liftului

- problema planificării brațului este asemănătoare planificării cererilor pentru un lift
- lifturile folosesc un algoritm care este un compromis între eficiență și fairness
- se mișcă brațul într-o direcție până când nu mai există cereri în acea direcție
- avem nevoie de un direction bit (up sau down)
- după fiecare cerere discul verifică acest bit și mută brațul în direcția corespunzătoare la următoare cerere

Algoritmul liftului

- dacă nu mai există cereri în direcția aceea, bitul de direcție va fi inversat
- ex. (precedent)

  seek time total – 60 cilindri (bit inițial up)

  seek time total – 45 cilindri (bit inițial down)
- în mod uzual algoritmul este mai slab decât SSF, dar există și multe excepții
- pentru foarte multe cereri limita superioară a numărului de cilindri pe care trebuie să parcurgă pentru secvența respectivă este jumătate din numărul total al cilindrilor

4.4. Securitatea sistemelor de fișiere

- securitate: problema generală a protecției informației împotriva accesului neautorizat
- mecanisme de protecție: mecanismele utilizate de un sistem de protecție la accesul neautorizat
- aspecte ale securității:
  - pierderea datelor
  - intruși

Pierderea datelor

- cauze comune ale pierderii datelor
  - catastrofe
  - erori hardware sau software: benzi și discuri ce nu mai pot fi citite, erori în program
  - erori umane: montarea incorectă a discurilor sau benzilor, rularea greșita a unor programe
- se contracarează prin menținerea unor backupuri

Problema intrușilor

- tipuri de intruși:
  - pasivi: doar citesc date la care nu au acces în mod normal
  - activi: efectuează modificări neautorizate asupra anumitor date
- categorii de intruși:
  - nespecialiști: citesc fișierele neprotejate ale altor utilizatori
  - specialiști: consideră spargerea unui sistem ca o provocare profesională
  - oameni care încearcă să facă bani
  - spioni

Probleme celebre în securitatea sistemelor

- link la /etc/passwd
  - intrusul va forța un core dump la un program setuid și citește din fișierul core începutul fișierului /etc/passwd
- mkdir
  - programul mkdir creează un director ca root după care realizează chown la user
  - dacă sistemul este lent se poate șterge i-nodul pentru noul director și făcut un link către /etc/passwd

The Internet Worm

- lansat în 2 noiembrie 1988 de Robert Tappan Morris, student la Cornell University
- se folosea de 2 buguri descoperite de Morris în sistemul BSD
- aceste buguri făceau posibil accesul neautorizat în sistem
- când Worm intra în sistem, se autocopia, după care folosind routing table, căuta alte mașini pe care putea infecta

Viruși

- un virus este un fragment de program care este atașat unui program normal cu scopul de a infecta și alte programe
- programul infectat trebuie rulat de un utilizator autentificat pentru ca virusul să poate intra în sistem
- după execuție virusul caută alte programe aflate în sistem și le va infecta
- un virus poate infecta și sistemul de fișiere sau chiar sistemul de operare, dacă a fost rulat cu drepturi suficiente

Atacuri generice

- datele aflate pe disc care au aparținut unor fișiere șterse, pot fi citite
- răspunsul sistemelor la apeluri de sistem ilegale este imprevizibil
- apăsarea tastelor speciale în timpul procedurii de login poate rezulta în login realizat în cazul unor sisteme
- încercarea de a modifica structurile sistemului de operare aflate în spațiul utilizator, de exemplu structurile de date pasate la kernel

Atacuri generice

- realizarea unui program care să imite procesul login
- identificarea în manualul sistemului operațiile interzise (Do not do X) și efectuarea în mod repetat a operațiilor respective
- convingerea administratorului de a permite unui utilizator de trecerea peste anumite verificări
- social engineering