Pád a obnovenie súborového systému ext3

11. prednáška

Operačné systémy

2.12.2019

Prečo logovanie?

Príklad: rozširovanie súboru (operácia *append*)
Súborový systém zapíše niekoľko blokov na disk:

- blok v bitmape označí ako použitý
- pridá číslo bloku do poľa `addrs[]` daného *inodu*

Pre súborový systém je nebezpečné robiť priamo zápisy na disk. Po páde a reštarte bude FS v nekonzistentnom stave. Niektoré bloky môžu byť naraz voľné (v bitmape) a používané (odkaz v *inode*).
Logovanie zabezpečí atomické zápisy – buď sa zapíše všetko alebo nič.

Logovanie v xv6

Každé systémové volanie je transakcia: začiatok, zápisy, koniec. Spočiatku sú zmeny zapísané iba do blokovej cache.
Na konci systémového volania:

1. Všetky modifikované bloky sa zapíšu do logu na disku
2. Do logu sa zapíšu čísla blokov a potvrdenie transakcie
3. Modifikované bloky sa zapíšu na svoje miesta na disku
4. Potvrdenie sa zmaže – obsah transakcie je na disku

Pri zlyhaní napájania:

- po reštarte sa načíta log
- ak je v logu zapísané potvrdenie:
  - zalogované operácie sú zapísané na disk
  - príznak potvrdenia je zmazaný

Pravidlo write-ahead: nezapisuj zmeny na disk, kým nepotvrdíš transakciu v logu
Pravidlo uvoľnenia: nezmaž log, až kým nie sú všetky zmeny zapísané na disk

Problémy s xv6 logovaním

- Je pomalé!
- Každé systémové volanie musí počkať na ukončenie operácií disku (synchrónny update)
- Nové systémové volania musia počkať na dokončenie potvrdenia – viaceré procesy nemôžu bežať konkurentne.
- Každý blok je na disk zapísaný dvakrát: raz do logu a raz do FS
- Vytvorenie prázdneho súboru vyžaduje 6 synchrónnych zápisov – 60 ms – možno vytvoriť iba 10-20 súborov za sekundu – veľmi málo!

ext3 z Linuxu

ext2 rozšírený o logovací systém
Čítanie/počúvanie pre záujemcov:

- `http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html`
- `https://pdos.csail.mit.edu/6.828/2019/readings/journal-design.pdf`

Funguje v niekoľkých módoch, začneme žurnálovanými dátami – log obsahuje metadáta aj bloky s obsahom súborov

Štruktúry ext3

v pamäti:

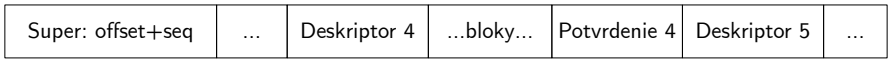
- bloková cache
- informácie o transakciách:
  - číslo sekvencie (seq)
  - čísla zalogovaných blokov
  - *handles* pre každé systémové volanie

na disku:

- samotný FS
- kruhový log

Čo obsahuje ext3 log (žurnál)?

- superblok logu: offset logu a sekvenčné číslo najstaršej validnej transakcie (nejedná sa o superblok FS, je to blok na začiatku logu)
- deskriptor transakcie: *magic*, seq, čísla blokov
- dátové bloky (tie, ktoré boli uvedené v deskriptore)
- potvrdenie: *magic*, seq



Ako dosahuje ext3 dobrý výkon?

- asynchrónna aktualizácia disku
- dávkové spracovanie
- konkurencia

Asynchrónna aktualizácia disku

- systémové volania neaktualizujú disk okamžite, iba modifikujú cachovaný blok

Asynchrónne aktualizácie zvyšujú výkon, lebo

- systémové volania sa môžu okamžite vrátiť
- konkurencia I/O, aplikácie môžu robiť niečo iné, zatiaľ čo disk zapisuje
- dávkovanie
- ale:
  - pri páde sa môže stratiť niekoľko posledných systémových volaní
  - výnimka: `fsync(fd)` prinúti aktualizáciu logu (a zároveň potvrdenie) pred návratom zo systémového volania
    - používajú ho databázy, textové editory, mailové spoolery

Dávkovanie

- ext3 potvrdzuje každých niekoľko sekúnd – každá transakcia obsahuje veľa systémových volaní

Dávkovanie zvyšuje výkon, lebo

- amortizácia fixných nákladov viacerých systémových volaní
  - fixné náklady sú: deskriptor a commit bloky, behanie hlavičky, rotácia disku, bariéra systémového volania
- absorbcia zápisov – veľa systémových volaní v transakcii modifikuje ten istý blok (bitmapu, dirent), čiže stačí jeden zápis na veľa *syscallov*
- plánovanie disku: zoradenie zápisov pre kratšiu cestu hlavičky disku

Konkurencia

Viaceré systémové volania v jednej transakcii

Niekoľko transakcií naraz v rôznych stavoch:

- jedna otvorená transakcia – akceptuje nové *syscally*
- potvrdzované transakcie, ktoré zapisujú do logu
- potvrdené transakcie, ktoré sú zapisované do FS
- uvoľňovanie starých transakcií

Konkurencia pomáha výkonu, lebo:

- veľa procesov môže naraz používať FS
- nové systémové volania môžu byť spracovávané, zatiaľ čo sú staré transakcie zapisované na disk

Kód systémového volania v ext3

```
sys_unlink() {
  h = start()
  get(h, cislo bloku) ...
  modifikuj bloky v cache
  stop(h)
}
```

`start()`

- ohlási logovaciemu systému nové systémové volanie
  - transakcia nebude potvrdená, až kým všetky systémové volania nezavolajú `stop()`
- môže blokovať systémové volanie (viď. bod 1. na ďalšom slajde)

`get()`

- oznámi logovaciemu systému číslo bloku, ktorý upravíme
- pridá ho do zoznamu blokov, ktoré budú zalogované

`stop()`

- nepotvrdí transakciu
- transakcia bude potvrdená len vtedy, keď všetky systémové volania zavolali `stop()`

Potvrdenie transakcie na disku

1. zablokuj systémové volania
2. počkaj kým bežiace systémové volania nezavolajú `stop()`
3. otvor novú transakciu, odblokuj nové systémové volania
4. zapíš deskriptor do logu so zoznamom čísel blokov
5. zapíš modifikované bloky do logu
6. počkaj na dokončenie diskových zápisov na logu
7. zapíš potvrdzovací blok do logu
8. počkaj na zapísanie potvrdzovacieho bloku
9. transakcia potvrdená
10. keď sa bude dať, bloky môžu byť zapísané na disk

Čo sa stane pri páde?

- bloková cache na RAM sa stratí
- predpokladáme, že disk zostal nepoškodený (mechanicky a elektronicky)
- pád mohol:
  - prerušiť zápis transakcie do logu (v logu môže byť niekoľko potvrdených transakcií a jedna nepotvrdená)
  - prerušiť zápis blokovej cache na disk (ale to iba pre potvrdenú transkaciu)

Ako funguje obnova ext3

1. naštartuj po páde
2. načítaj superblok logu a offset najstaršej transakcie v logu
3. nájdi koniec logu
   - 3.1 prechádzaj log až kým nenarazíš na chybný *magic* (chýbajúce potvrdenie) alebo neočakávané číslo transakcie (menšie ako najstaršie)
   - 3.2 ak chýba potvrdenie – transakcia je neplatná
4. zapíš všetky potvrdené transakcie na disk, počnúc najstaršou

Poškodenie dát v logu

Čo ak blok po poslednej potvrdenej transakcii vyzerá ako deskriptor?

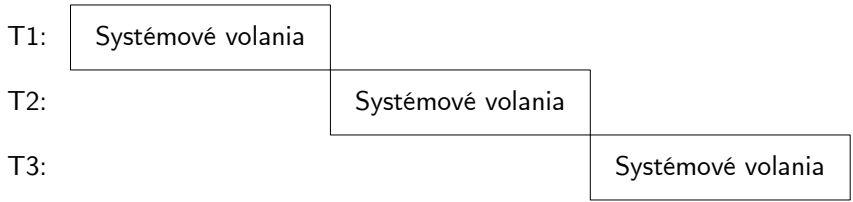
(teda ako začiatok novej transakcie)
Ak sa jedná o staršiu transkaciu, jej seq bude príliš nízke.
Môže dátový blok vyzerať ako deskriptor?
Nie, ext3 zakazuje magické číslo v dátových blokoch v logu.

Čo ak nastane výpadok pri obnove dát z logu?

Pri ďalšom štarte obnova začne odznova.

A teraz zákerné detaily, na ktoré museli tvorcovia ext3 myslieť!

Prečo ext3 oneskorí *syscally* T2, kým sa nedokončia *syscally* T1?



Bariéra obetuje nejaký výkon, aby zamedzila situácii na ďalšom slajde.

Chybový scenár

- súbor `y` má inode 17
- otvorená T1
- systémové volanie T1a vytvorí súbor `x`
- T1 sa uzavrie, čaká na T1a
  - otvorená T2
  - T2a vymaže `y`
  - T2a označí inode 17 ako voľný
- T1a alokuje inode 17
- T1a vytvorí `f1` na inode 17
- T1a sa ukončí
- T1 je potvrdená
- pád (T2 nestihla potvrdiť)
- obnova vidí iba T1
- `y` aj `x` budú existovať a budú ukazovať na ten istý inode!

Aby sa toto nestalo, ext3 vyžiada T2a, aby počkalo na dokončenie T1a.

Málo miesta!

Čo ak sa do logu nezmestia ďalšie bloky?

Uvoľni najstaršiu transkaciu (zapíš jej bloky na disk)

Čo ak prebieha toľko systémových volaní, že na ne log nestačí (a ani uvoľnenie starších transakcií nepomôže)?

(malá pravdepodobnosť, že sa to stane)
Riešenie: systémové volanie dá pri volaní `start()` vedieť, koľko blokov použije.
`start()` počká, kým bude v logu dostatok voľných blokov

Čo by sa stalo, keby sa miesto v logu nerezervovalo

- T1 zapíše blok 17
- T1 potvrdí
- T1 ukončí zápis do logu, ale blok 17 ešte nezapíše na disk
- T2 upraví blok 17 v cache
- T2 vykoná systémové volanie, ktorého zápis sa nezmestí do logu
- log nemôže aplikovať T1, lebo blok 17 bol v cache upravený (ext3 ho neprečíta z logu)

Rezervácia by zastavila systémové volania v T2, kým sa neuvoľní miesto v logu.

Všetko doteraz bol mód žurnálovania dát (journaled data)

Vlastnosti:

- Bloky s obsahom súborov sú zapísané do logu
- metadáta tiež (*inody*, obsah priečinkov, bitmapy)
- zaručená atomicita obsahu súborov – keď rozširujem súbor, nové dáta v nových blokoch, upravená bitmapa, upravený inode sú v transakcii, na disk sa dostane všetko alebo nič

Nevýhoda:

- je to pomalé, každý blok je zapísaný dvakrát

Mód zoradených dát (ordered data)

Čo keby sme do logu zapisovali iba metadáta?
Kedy zapíšeme skutočné dáta do FS? Záleží na tom?
Áno, ak metadáta potvrdíme pred tým, ako zapíšeme bloky na disk, po páde môžu niektoré súbory ukazovať na nezapísané bloky z predtým zmazaných cudzích súborov

Mód zoradených dát

- nezapisuje dátové bloky do logu
- dátové bloky zapíše na disk predtým ako potvrdí *inode* s novou veľkosťou a číslom bloku

Stav bez pádu

Čitatelia uvidia zapísané dáta

Pád po zápise blokov, ale pred potvrdením

- na disku sú nové dáta
- nie sú viditeľné, lebo veľkosť *inodu* a zoznam blokov nebol aktualizovaný
- metadáta zostávajú neporušené
- štandardne na väčšine Linuxových distribúcií

Korektnosť módu zoradených dát

rmdir, znovupoužitie bloku s obsahom adresára pre iný súbor

Ak pád nastane pred potvrdením *rmdiru* – priečinok bude stále existovať, ale obsah jeho bloku bude prepísaný!

Riešenie

Nepoužívať uvoľnené bloky až kým uvoľňujúce systémové volanie nepotvrdí transakciu.

rmdir, potvrdenie, znovupoužitie bloku, zápis, potvrdenie, pád, obnovenie, zopakovanie rmdir

- zopakovaním `rmdir` prepíšeme súbor na disku
- obsah súboru neobnovíme, lebo ten v logu nie je!

Riešenie

Vložiť zápisy o odvolaní do logu na zabránenie zopakovania daného bloku
Oba problémy vznikli, lebo sa menil typ bloku (obsah vs metadáta). Takže iné riešenie by bolo zakázať zmenu typu bloku.

Súhrn ext3 pravidiel

- Nezapisuj metadáta na disk až kým nie sú potvrdené v logu.
- Dokonči všetky systémové volania v T1 pred začatím T2.
- Neprepisuj blok v cache, pokým nie je v logu.
- Neuvoľnuj transkaciu v logu pokým nie sú všetky bloky zapísané do FS.

Zoradený mód

- Zapíš dátový blok pred potvrdením
- Nepoužívaj uvoľnený blok kým uvoľňujúce systémové volanie nepotvrdilo
- Neopakuj odvolané systémové volania

Osirotené súbory

- otvor súbor, `unlink`-uj ho
- súbor je otvorený, `unlink` odstráni *dirent*, ale neuvoľní inode alebo bloky
- `unlink` potvrdí odstránenie *dirent*
- pád
- log neobsahuje zápisy, ktoré uvoľnia *inody* alebo bloky
- *inode* a bloky nie sú označené ako voľné a nie sú prístupné cez FS

Riešenie

Ext3 pridáva novú štruktúru do superbloku – zoznam blokov na zmazanie pri reštarte.

Kontrolné súčty

- riziko: disky majú vlastné cache a preusporiadavanie zápisov pre zlepšenie výkonu
- niekedy je to ťažké vypnúť, alebo disk o tom klame
- zlé to je, ak disk zapíše blok potvrdenia pred zvyškom transakcie

Riešenie

- potvrdzovací blok obsahuje kontrolný súčet všetých dátových blokov
- transakcia bude počas obnovy aplikovaná len vtedy, ak sedí kontrolný súčet
- kontrolný súčet pridaný v ext4

Ako ext3 vyriešil problémy s výkonom, ktoré máme v xv6?

synchrónny zápis na disk vyriešené asynchrónnymi volaniami

väčšina *syscallov* generuje veľa zápisov vyriešené asynchrónnym dávkovaním

málo konkurencie vyriešené dávkovaním a konkurentným potvrdením

každý blok zapísaný dvakrát mód zoradených dát

ext3 to všetko vyriešil!