Gagnasafnsfræði

Hreyfingar

Hallgrímur H. Gunnarsson

Inngangur

Gagnagrunnar eru geymdir á diskum. Diskar vinna með blokkir (512 bæti). Blokk er minnsta einingin í samskiptum við disk.

Diskur heldur utan um auka bókhald, m.a. villukóða (checksum) fyrir hverja blokk. Villukóði uppfærður *eftir* að blokk er skrifuð.

Þrjár mögulegar útkomur þegar blokk er skrifuð á disk: 1) ekkert skrifað, 2) allt skrifað, 3) eitthvað skrifað (og villukóði rangur!)

Villukóði kemur upp um hálfkláraðar breytingar. En hvernig getum við bjargað okkur úr slíku ástandi? Eldri gögn í blokkinni eru töpuð.

Hvað ef við viljum uppfæra margar blokkir samtímis og fá sama eiginleika (allt eða ekkert) ?

Spurningar

Áreiðanleg kerfi:

Hvernig er hægt að smíða áreiðanlega gagnageymslu úr óáreiðanlegum einingum (unreliable components) ?

Atómískar aðgerðir:

Hvernig er hægt að framkvæma atómíska breytingu á gögnum þegar þau hvíla í geymslu sem styður ekki slíka aðgerð?

Hálfkláraðar breytingar:

Hvernig getum við tryggt okkur gegn því að tapa gögnum og enda með hálfkláraðar breytingar?

Endanlegt markmið: Hreyfingar

Hreyfing:

Hreyfing (e. transaction) er þægilegt hugtak.

Gott dæmi um abstraktion í tölvunarfræði. Skilgreinum nýtt hugtak til að geta unnið á hærra stigi.

Leyfir okkur að vinna með breytingar í "áreiðanlegum einingum", án þess að þurfa að hugsa um hvernig það er útfært.

Í dag munum við fjalla um hvernig það er hægt að útfæra hreyfingar (nánar til tekið A og D eiginleikana).

Bottom-up umfjöllun. Byrjum með einfalt líkan og vinnum okkur upp.

Eiginleikar hreyfinga

Atomicity (A): (all-or-nothing atomicity)

Hreyfing er atómísk á þann hátt að annaðhvort er allt gert eða ekkert gert

Consistency (C):

Hreyfing viðheldur réttleika gagnasafnsins. Ef gagnasafn er í löglegri stöðu þá verður það áfram í löglegri stöðu eftir hreyfingu.

Isolation (I): (before-or-after atomicity)

Ef tvær hreyfingar T_1 og T_2 eru framkvæmdar samtímis þá er niðurstaðan sú sama og að framkvæma þær í röð, þ.e. T_1, T_2 eða T_2, T_1

Durability (D):

Eftir að hreyfing hefur verið bókuð (committed) þá helst hún bókuð.

Einfalt líkan

Líkan:

Hugsum okkur einfalt kerfislíkan. Lögleg breyting færir kerfið úr einni löglegri stöðu í aðra löglega stöðu. Stöðubreyting er ekki atómísk (og kerfið gæti hrunið í miðri breytingu.)

$S_1 \to S_2 \to S_3 \to ... \to S_n$

Kerfishrun:

Grundvallarspurning: hvað gerist ef kerfið hrynur skyndilega á tímapunkti T? Í hvaða stöðu er kerfið?

Ef kerfið hrynur þegar það er í löglegri stöðu þá er ekkert vandamál.

En hvað ef kerfið hrynur í miðri færslu $S_n \to S_{n+1}$? Það er hvorki í S_n né í S_{n+1} heldur í e-u óskilgreindu millibilsástandi.

Líkan frh.

Endurreisn eftir hrun:

Endurreisn eftir hrun (e. crash recovery) snýst um að taka kerfi í ólöglegu millibilsástandi og endurreisa það þ.a. það sé í löglegu ástandi.

Ef kerfið hrynur í færslu $S_n \rightarrow S_{n+1}$ þá mun endurreisn kerfisins færa það annaðhvort í stöðu S_n eða S_{n+1}

Samhliða vinnsla:

Einfalda líkanið gerir bara ráð fyrir einni stöðubreytingu í einu, en hvað ef við viljum leyfa margar breytingar samtímis?

Hvernig skilgreinum við löglegu stöðurnar í slíku líkani? Hvernig má flétta breytingar saman?

Dæmi: Einfalt bankakerfi

Lýsing:

Skulum hugsa okkur einfalt bankakerfi þar sem innistæður eru geymdar á disk.

Höfum aðgerð xfer(a,b,amount) sem millifærir upphæð frá reikningi a yfir á reikning b.

Millifærsla felur í sér tvær breytingar. Millifærsla er því ekki atómísk breyting.

$S_1 \overset{\text{xfer}}{\rightarrow} S_2 \overset{\text{xfer}}{\rightarrow} \ldots \overset{\text{xfer}}{\rightarrow} S_n$

Hrun:

Ef kerfið hrynur í miðri millifærslu þá er það í ólöglegu millibilsástandi.

Hvernig getum við útfært xfer þ.a. það sé atomískt (all or nothing) ?

Einföld útfærsla á atomic xfer (shadow copy)

Forsendur:

Allar innistæður eru geymdar í einni stórri skrá í skráarkerfinu.

Atomic rename aðgerð í skráarkerfinu.

Linux man page: rename(oldpath,newpath): If newpath already exists it will be atomically replaced

Atomic xfer:

Afritum skrána. Breytum afritinu. Framkvæmum atomic rename.

Hrun:

Ef kerfið hrynur í framkvæmd xfer þá eru tveir möguleikar: búið að framkvæma rename eða ekki. Kerfið er þá annaðhvort í S_{n+1} eða S_n. Það eru engir aðrir möguleikar.

Kostir og gallar við shadow copy

Ein skrá:

Aðgerðin virkar fyrir eina skrá en erfitt að útvíkka og nota fyrir margar skrár.

Dýrt:

Dýrt að afrita alla skrána fyrir hverja breytingu (sama hversu stór eða lítil)

Raðbundið:

Raðbundin vinnsla, þ.e. ein aðgerð í einu. Styður ekki að óbreyttu samskeiða vinnslu þar sem margar xfer aðgerðir eru í gangi samtímis.

Notendur:

Þrátt fyrir ýmsa ókosti þá er shadow copy mjög einföld og þægileg útfærsla á "all or nothing" atomískum aðgerðum. Mörg forrit sem nota shadow copy, t.d. ritlar (editors).

Commit point

Bókun (e. commit point):

Í tilfelli shadow copy þá er atomíska rename kallið svokallað *commit point*.

Hrun fyrir bókun gefur gömlu stöðuna; hrun eftir bókun gefur nýju stöðuna.

Hætt við bókun (e. abort):

Ef við erum byrjuð á breytingu en viljum síðan hætta við hana (abort) þá gætum við einfaldlega hent afritinu og sleppt rename.

Gullna reglan:

Gullna reglan um útfærslu á atómískum aðgerðum: never modify the only copy!

Í shadow copy er gert afrit af skránni og breytingin er framkvæmd á afritinu (síðan atomic rename). Fylgir gullnu reglunni.

Önnur útfærsla á atomic xfer

Dagbók (e. log):

Skrifum allar breytingar í svokallaða dagbók. Yfirskrifum aldrei gögn í dagbókinni, bætum bara inn nýjum færslum (dagbókin er append-only).

Dæmi: xfer(a,b,50) - fjórar færslur skráðar í dagbókina
Dagbók:

```
begin
a = a - 50
b = b + 50
commit
```

Breytingar sem tilheyra sömu atómísku einingu (s.k. *hreyfingu* eða *transaction*) eru settar saman í blokk sem byrjar á begin og endar á commit.

Færslutegundir

BEGIN:

Tilgreinir byrjun á nýrri hreyfingu

COMMIT:

Bókar hreyfingu. Skráning á commit færslu er *commit point* fyrir hreyfinguna.

ABORT:

Hætt við bókun á hreyfingu.

UPDATE:

Uppfærir innistæðu. Inniheldur bæði nýja og gamla gildið.

Til að finna núverandi innistæðu á reikningi þá lesum við yfir alla dagbókina og finnum síðustu bókuðu breytinguna

Skráning í dagbók

Skráning:

Skráning í dagbókina verður að vera atómísk (all or nothing). Litlar færslur og líklegt að færsla komist fyrir í einni blokk.

Samt mögulegt að lítil færsla spanni tvær blokkir (byrjar í lokin á einni blokk og heldur áfram í þeirri næstu)

Hrun:

Ef það verður hrun í miðri skráningu á færslu þá er mögulega síðasta færslan í dagbókinni hálfkláruð.

Einföld lausn: lesum yfir alla dagbókina þegar kerfið er ræst aftur eftir hrun, ef það er hálfkláruð færsla í lokin þá skerum við hana burt (log truncate)

Dæmi um dagbók

```
begin
A = 100
B = 50
commit

begin
A = A - 20
B = B + 20
commit

begin
A = A + 30
--CRASH--
```

Betri útfærsla (dagbók + staða)

Núverandi útfærsla:

Í núverandi útfærslu erum við bara með dagbók.

Mjög dýrt að lesa innistæðu. Þarf að spóla í gegnum alla dagbókina.

Betri útfærsla:

Höldum bæði dagbók yfir breytingar og geymum núverandi stöðu á disk.

$S_1 \stackrel{T1}{\rightarrow} S_2 \stackrel{T2}{\rightarrow} ... \stackrel{Tn}{\rightarrow} S_n$

Dagbókin inniheldur allar hreyfingar: $T_1, T_2, ..., T_n$

Geymum aukalega S_n á disk. Ódýr lestur: lesum bara núverandi gildi af disk.

Hreyfingar

Framkvæmd:

Til að framkvæma hreyfingu $S_n \xrightarrow{Tk} S_{n+1}$ þá þarf að skrá T_k í dagbókina (e. log updates) og uppfæra núverandi stöðu á disk (e. install updates)

Röðun:

Skiptir röðin á log og install máli?

Eigum við að uppfæra stöðu á disk fyrst og síðan skrá færslur í dagbókina

eða skrá fyrst færslur í dagbókina og síðan uppfæra stöðuna?

Hreyfingar frh.

Röðun:

Skiptir röðin á log og install máli? Já!

Tveir möguleikar:

Ef við uppfærum stöðuna fyrst: ef það verður hrun í miðri uppfærslu þá erum við í vondum málum! Brýtur gullnu regluna: ekki breyta eina eintakinu.

Ef við skráum í dagbókina fyrst (Write-Ahead Logging) og uppfærum síðan stöðuna þá eru tveir möguleikar:

1. Hrun við skráningu í dagbók: þá er hætt við bókun á hreyfingunni
2. Hrun við uppfærslu á stöðunni: ekkert vandamál, notum dagbókina til að leiðrétta stöðuna

Write-Ahead Logging (WAL)

WAL:

Einföld regla: "Always log the update before installing it"

Trygg röðun:

Hvernig getum við tryggt að það sé búið að skrifa dagbókina á disk áður en við byrjum að breyta stöðunni?

Við viljum ekki að dagbókarfærslurnar hangi í page cache stýrikerfisins. Gætum lent í ófyrirsjáanlegri I/O verkröðun sem brýtur WAL regluna.

fsync(fd): skrifar allar dirty síður sem tengjast skránni *fd* út á disk. Kallið snýr ekki til baka fyrr en allt hefur verið skrifað.

(Notum frekar fdatasync(fd) ef það er til staðar, það skrifar ekki metadata út á disk, t.d. last modified time, bara dirty síður sem tengjast gögnum í skránni)

Endurbót 1: frestum install og notum cache

Lazy update:

Uppfærum alltaf dagbókina á disk en uppfærum ekki stöðuna á disk strax.

Uppfærum í staðinn stöðuna í minni (fáum dirty page í buffer cache) og uppfærum síðan stöðuna á disk lazily (t.d. reglulegt flush í bakgrunnsþræði.)

Lestur:

Lesum stöðuna úr minni ef hún er í cache, en förum annars út á disk (og sækjum síðu af disk yfir í buffer cache).

Misræmi:

Möguleiki á misræmi, þar sem staðan á disk er ekki endilega sú nýjasta.

Endurbót 1 frh.

Misræmi:

Möguleiki á misræmi, þar sem staðan á disk er ekki endilega sú nýjasta.

Gætum haft uppfærslur á stöðu í minni (dirty pages) sem á eftir að skrifa út á disk.

Ef við uppfærum jafnóðum þá gæti staðan á disk gæti innihaldið uppfærslur sem var hætt við að bóka (aborted updates).

Endurreisn:

Göngum yfir dagbókina og leiðréttum stöðuna á disk.

Afbókum (undo) allar breytingar í hreyfingum sem var hætt við að bóka.

Endurgerum (redo) allar breytingar í hreyfingum sem voru bókaðar.

Endurbót 2: stytta dagbókina

Núverandi hönnun:

Núverandi hönnun gerir ráð fyrir að dagbókin stækki endalaust. Ekki mjög praktískt.

Hvaða hluta af dagbókinni má henda?

Megum henda öllum hreyfingum úr dagbókinni fram að tilteknum punkti ef þær eru allar lokaðar, þ.e. endanleg niðurstaða (commit/abort) er skráð, og staðan á disk endurspeglar stöðuna í dagbókinni í þeim punkti.

Endurbót 2 frh.

Stytting:

Grf. að engin hreyfing sé í gangi, þ.e. stytting er framkvæmd á milli hreyfinga.

Þrjú skref:

1. Skráum svokallaða *checkpoint* færslu í dagbókina
2. Skráum allar breytingar á disk (flush updates).
3. Styttum dagbókina: hendum öllum hreyfingum fram að checkpointinu.

Opnar hreyfingar:

Ef það eru hreyfingar í gangi þá framkvæmum við sömu skref, nema í þriðja skrefi styttum við fram að fyrstu opnu hreyfingu á undan checkpoint

Samantekt

Dagbækur:

Skráning í dagbók er almenn aðferð til að útfæra all-or-nothing atomicity.

Notað í gagnasafnskerfum til að útfæra hreyfingar.

Runubundin skrif í dagbók (sequential logging), því hún er append-only

Hægt að fá hraðan lestur með því að geyma einnig stöðuna á disk (og enn betri hraða með því að geyma stöðu í minni)

Lykilatriði:

WAL: einföld regla sem tryggir örugga uppfærslu og möguleika á endurreisn eftir hrun

Endurreisn: undo losers, redo winners