Johdanto

- David Lomet (Microsoft Corporation), Mark R. Tuttle (Digital Equipment Corporation): Redo Recovery after System Crashes

- Tarkoituksena määritellä Redo -toipumismenetelmän perusteet järjestelmän kaatumisen jälkeen.

- Redo -menetelmää tutkittaessa käytetään hyväksi installointigraafia (installation graf), joka kuvaa sitä järjestystä, jossa operaatiot pitää kirjoittaa stabiiliin tietokantaan (levylle), jotta tietokanta pystyy toipumaan kaatumisen jälkeen.

- Installointigraafia käytetään suunnittelemaan
  - Kätkömuistin hallinta-algoritmi datan kirjoittamiseen epävakaasta kätkömuistista stabiiliin tietokantaan
  - REDO -testi, jota käytetään valitsemaan logista ne operaatiot jotka suoritetaan uudelleen järjestelmän toipumisen yhteydessä
  - Toistuvastikäytettävä (idempotent) toipumisalgoritmi, joka perustuu REDO -testiin.

- Useimmat käytetyistä toipumismenetelmistä asettavat rajoituksia sille kuinka paljon erilaisia operaatioita logissa voi esiintyä, mutta REDO -toipumisessa ei erilaisten operaatioiden määrälle aseteta rajoituksia.

Miten kaatumisesta toivutaan?

- Tarvitaan vastaukset seuraaviin kysymyksiin:
  - Miten stabiili tila voidaan määritellä levylle talletettujen ja levylle tallentamattomien operaatioiden avulla?
  - Miten toipumisalgoritmin tulee valita uudelleen suoritettavat operaatiot, jotta se pääsisi määritettävään (explainable) tilaan?
  - Miten kätkömuistin hoitajan (cache manager) tulisi tallettaa operaatioita stabiiliin tietokantaan siten, että järjestelmän tila pysyisi määritettynä ja siten toipumiskykyisenä?
- Yksi vastaus näihin kysymyksiin on suoritettujen operaatioiden tallentaminen logiin.
- Tämän tutkielman perustana on installointigraafi -logi, joka rajoittaa missä järjestyksessä operaatioiden aikaansaamat muutokset talletetaan stabiiliin tietokantaan.
- Jos järjestelmän kaatumisen hetkellä järjestelmän tila pystytään määrittelemään installointigraafin avulla, saadaan järjestelmä toipumaan logissa olevien tietojen avulla.
- Sekä kätkömuistin hallinta -algoritmi, että toipumisalgoritmi perustuvat installointigraafista johdettuihin ehtoihin.

REDO -toipumisen perusteet

- Kun tietokone kaatuu muistin sisältö joko häviää kokonaan tai ainakin korruptoituu. Ainoastaan levylle talletetun tiedon voidaan olettaa säilyvän.

- Jotta tietokanta toipuisi järjestelmän kaatumisesta, on sen tila tallennettava levylle, stabiiliin tietokantaan.

- Levyltä lukeminen ja kirjoittaminen on kuitenkin hidasta ja tämän takia osia tietokannan tilasta säilytetäänkin kätkömuistissa. Kätkömuistin hoitaja on vastuussa kätkömuistissa olevien tietojen kirjoittamisesta levylle joskus tulevaisuudessa.

- Koska ei voida olla varmoja siitä, että kätkmuistin hoitaja pystyisi kirjoittamaan kaikki kätkömuistissa olevat tiedot levylle ennen järjestelmän kaatumista, käytetään logia, johon kaikki tehdyt operaatiot kirjataan ylös. Järjestelmän kaaduttua voidaan toipua login avulla.

- Login loppuosaa säilytetään usein levyllä ja alkuosaa muistissa. Login hoitajan (log manager) tehtävänä on siirtää logitietueita muistista levylle.

- Yleensä login hoitaja käyttää 'write-ahead' logiprotokollaa (WAL). Se toimii siten, että ennen kuin operaation O suorittamisen aikaansaamia muutoksia kirjoitetaan stabiiliin tietokantaan, kirjoitetaan logiin merkintä O:sta ja kaikki sitä edeltävät merkinnät logissa kirjoitetaan stabiiliin tietokantaan.

- REDO -toipuminen toimii siten, että aloitetaan login lukeminen jostain kohtaa logia ja luetaan login loppuun saakka. Tutkiessaan logimerkintöjä, logitettu operaatio joko käynnistetään uudelleen tai hypätään sen yli.

- REDO -toipuminen on vain yksi toipumistapa muiden joukossa, mutta se on tärkeä tekniikka, koska kaikki tehokkaat toipumismenetelmät käyttävät sitä. Se on tehokas, koska kätkömuistin hoitajan ei tarvitse kirjoittaa kaikkia muutoksia heti transaktion päätyttyä stabiiliin tietokantaan.

REDO -toipumisen ongelmat

- Käyttäjän suorittamat operaatiot ja logiin kirjattavat operaatiot saattavat olla hyvinkin erinäköisiä. Esim. Tietueen lisääminen B-puuhuun saattaa käyttäjästä olla yksittäinen tapahtuma, mutta saattaa todellisuudessa käsittää useita kirjoituksia logiin, ainakin jos puuta joudutaan pilkkomaan lisäyksen takia.

- Kaksi yleistä logitusstrategiaa: After-image logitus, joka tunnetaan myös fyysisenä logituksena ja looginen logitus.

- Fyysisessä logituksessa logimerkintä koostuu muuttujan nimestä ja muuttujalle annetusta arvosta. Ne ovat siis muotoa $x \leftarrow y$, mikä kuvaa sokeaa kirjoitusta (blind write)(tyypillisesti yksi levysivu). Tällä tekniikalla on se etu, että kätkömuistin hallinta ja toipuminen ovat helppoja: Kätkömuistin hoitaja voi installoida muutoksia stabiiliin tietokantaan missä järjestyksessä tahansa ja toipuminen on silti mahdollista. Toipumisprosessin tarvitsee vain aloittaa tutkimaan logia siitä kohden mistä installoimattomat operaatiot alkavat ja kirjoittaa sitten muuttujille niiden arvot login kertomassa järjestyksessä. (esimerkki 1)

- Loogisessa logituksessa logimerkintä koostuu operaation nimestä ja niistä parametreistä, joilla operaatiota kutsuttiin. Tällä tekniikalla on se etu, etä logimerkinnät ovat paljon pienempiä kuin fyysisessä logituksessa, sillä operaation nimi ja parametrit ovat yleensä paljon pienempiä kuin levysivun kopio. Tällöin kätkömuistin hallinta on kuitenkin hankalampaa. (esimerkki 2)

Ratkaisu ongelmaan

- Installointigraafi kuvastaa sitä kuinka paljon kätkömuistin hoitaja voi joustaa siitä järjestyksestä jossa dataa kirjoitetaan kätkömuistista levylle.

- Tietokanta on toipuva jos tietokannan tila voidaan määrittää (explained) installointigraafin alkuosassa olevien installoitujen operaatioiden, sekä logista löytyvien installoimattomien operaatioiden avulla.

- Hyvin tunnettu operaatioiden osittaisjärjestys on Konfliktigraafi (conflict graph). Operaatiot O ja P konfliktoivat jos toinen niistä kirjoittaa muuttujaan x, jota toinen lukee tai kirjoittaa.

- Konflikteja on kolmea lajia: luku-kirjoitus, kirjoitus-luku ja kirjoitus-kirjoitus.

- Jos operaatiot kirjoitetaan stabiiliin tietokantaan konfliktigraafin mukaisessa järjestyksessä pysyy tietokanta toipuvana. Suorittamalla installoimattomat operaatiot uudelleen konfliktigraafin mukaisessa järjestyksessä, saadaan tietokannan tila toipumaan.

- Kaikkia konfliktityyppejä ei kuitenkaan tarvitse ottaa huomioon: Kirjoitus-luku -konfliktia voidaan pitää tarpeettomana. (esimerkki 3)

- Joskus myös kirjoitus-kirjoitus konfliktit ovat turhia. (esimerkki 4)

Tietokantamalli 1/3

- Tietokannan tila on funktio joka kuvaa joukon V arvot joukolle γ. Yksi tila valitaan alkutilaksi.

- Operaatio on funktio, joka kuvaa tilat toisille tiloille.

- Operaation kutsuminen kuvaa tietyn operaation kutsumista. Se on monikko, joka koostuu kutsuttavasta operaatiosta, operaation luku- ja kirjoitusjoukoista, sekä niiden ennen ja jälkeen olevista 'kuvista' (before and after images).

- Muutoksia jotka operaatio tekee sen kirjoitusjoukkoon, ei tarvitse tallettaa tietokantaan atomisesti, ne voidaan sen sijaan tallettaa sarjana atomisia talletuksia, mahdollisesti sivu kerrallaan. Tämä voidaan mallintaa jakamalla operaation O kirjoitusjoukko sarjaan $U_1, \ldots U_u$ erillisiä alijoukkoja. U_i tulkitaan niin että muutokset, jotka O tekee U_i:ssä oleviin muuttujiin pitää tallettaa atomisesti. Kutsutaan paria $< O, U_i >$ päivitykseksi.

- Yleensä ei ole välttämätöntä tai toivottavaa suorittaa toipumistilanteessa uudelleen operaation koko kirjoitusjoukkoa, sillä voimme käyttää jonkinlaista testiä tutkimaan mitkä operaation suorittamista päivityksistä pitää suorittaa uudelleen. Mitä tarkempi tämä testi on sitä pienempi määrä tarvitsee uudelleen kirjoituksia suorittaa. Tätä voidaan mallintaa siten että operaation O päivitysjoukko on jaettu joukkoon REDO -operaatioita. Toipumisprosessi voi valita uudelleen suoritettaviksi vain muutaman operaation REDO:n.

- Toipumisen aikana REDO on kaikki tai ei mitään -suoritus: Joko kaikki REDO-operaation tekemät päivitykset talletetaan uudelleen tai sitten ei mitään niistä.

- Merkitään: REDO $[O, U]$ sisältää päivityksen $< O, U >$

Tietokantamalli 2/3

- Kaksi operaatiokutsua ovat konfliktissa, jos toisen kirjoitusjoukko leikkaa toisen luku- tai kirjoitusjoukkoa.

- Konfliktigraafi järjestää konfliktissa olevat operaatiokutsut O:n ja P:n polulla O:sta P:hen, jos O esiintyy ennen P:tä, tällöin sitä merkitään $O \leq P$.

- Historia H on sellainen konfliktigraafi, joka kuvaa tietokannan suoritusta. Eli konfliktigraafi, joka joka sisältää kaikki koskaan tietokannassa suoritetut operaatiot.

- Oikeassa tietokantajärjestelmässä ei yleensä tarvita historiaa, koska tila ja logi sisältävät kaiken sen tiedon mitä tarvitaan. Historiaa voidaan kuitenkin käyttää toipumisprosessin oikeellisuuden osoittamiseen. Sanotaan että D on toipunut, jos sen tila S on seuraus kun lähdetään alkutilasta ja suoritetaan historiassa kerrotut tapahtumat historian määräämässä järjestyksessä. Lisäksi sanotaan että D on toipuva jos se voidaan muuttaa toipuneeksi tietokannaksi tallettamalla jotkin REDO:t joillekin operaatioille, jotka ovat logissa, jossain login määräämässä järjestyksessä.

- Logi L on H:n aligraafi, joka sisältää osittaisen kuvauksen historiasta.

- Tietokantajärjestelmä D on kolmikko (S, L, H), joka koostuu tilasta S, historiasta H, sekä logista L.

Tietokantamalli 3/3

- Tietokannan toteutus jakaa tietokannan stabiiliin ja epävakaaseen komponenttiin. Stabiili tietokanta (S_S, L_S, H_S) kuvaa levyllä sijaitsevaa osaa tietokannasta. Epävakaa tietokanta tai kätkömuisti $(S_V, L_V, H_V, U_V, B_V)$ sisältää kaksi lisärakennetta:
  - Päivityspuskuri U_V, joukko päivityksiä joita ei vielä ole talletettu stabiiliin tilaan S_S.
  - Logipuskuri B_V, työskentelylogi joka koostuu operaatioista, joita ei vielä ole merkitty stabiiliin logiin L_S
- Looginen tietokanta on tietokanta (S_l, L_l, H_l), jossa tila S_l on yhdiste stabiilista ja epävakaasta tilasta, logi L_l on epävakaa logi ja historia H_l on epävakaa historia. Muuttujan x arvo loogisessa tilassa on x:n arvo epävakaassa tilassa, jos se on siellä määritelty ja muuten x:n arvo stabiilissa tilassa.
- Toipumisen alussa epävakaa tietokanta resetoidaan seuraavasti: Epävakaa logi ja historia asetetaan samoiksi kuin stabiili logi ja historia, ja epävakaa tila, päivityspuskuri ja logipuskuri asetetaan tyhjiksi. Yksi toisensa jälkeen operaatioita O valitaan login alusta ja valitaan jokin sarja O:n REDO:sta $[O, U]$. Jokaiselle valitulle REDO $[O, U]$:lle suoritetaan sen päivitysoperaatiot $< O, U >$ ja lisätään ne päivitysbufferiin. Lopulta O poistetaan logista. (esimerkki 5)

Must(O) ja Can(O)

- Operaatiolle O määritellään uudelleen suoritettavien operaatioiden joukko (must redo set of O), jota merkitään $must(O)$ seuraavasti: $must(O)$ sisältää kaikki ne operaatiota O konfliktigraafissa seuraavat operaatiot, joihin tehdyt muutokset resetoituvat jos O suoritetaan uudelleen toipumisen aikana. Siksi nämä kaikki seuraajatkin pitää suorittaa uudelleen toipumisen yhteydessä.

- Operaatiolle O määritellään myös joukko can redo set of O, jota merkitään $can(O)$, joka sisältää ne operaatiot jotka resetoituvat O:n uudelleen suorittamisen yhteydessä ja jotka voidaan suorittaa uudelleen operaation O uudelleensuorittamisen jälkeen.

- Installointigraafi saadaan nyt konfliktigraafista, siten että kaikki luku-luku kärjet säilytetään, kirjoitus-luku -kärjet poistetaan ja sellaiset kirjoitus-kirjoitus -kärjet operaatiosta O operaatioon P säilytetään, joissa $P \in must(O) \setminus can(O)$

- $A \setminus B$ tarkoittaa joukkojen A ja B erotusta.

Määritettävä tila

- Tietokanta $D = (S, L, H)$ on määritetty jos installointigraafin alussa on prefiksi σ, siten että σ määrittää tilan S ja σ sisältää kaikkien operaatioiden kaikki päivitykset, jotka ovat historiassa O, mutteivät logissa L.

- Määritetty tietokanta on toipuva tietokanta.

- Jotta voidaan todistaa että kaatumisesta toipuminen on mahdollista, pitää osoittaa miten stabiili tietokanta saadaan pidettyä määritettynä. Eli miten stabiilin tietokannan tila pysyy määritettynä ja stabiili logi sisältää kaikki installoimattomat operaatiot.

- Onneksi WAL protokolla ja tarkistuspiteet takaavat että kaikki installoimattomat operaatiot ilmestyvät stabiiliin logiin. Yksityiskohtaisemmin WAL -protokolla takaa että operaatio lisätään L_S:ään samalla hetkellä kun se lisätään H_S:ään ja tarkistuspisteet takaavat sen, että vain installoidut operaatiot poistetaan L_S:stä.

Seuraukset

- Installointigraafin avulla voidaan osoittaa, että jos stabiilin tietokannan tila voidaan määrittää installointigraafin alkuosan avulla, voidaan stabiilia logia käyttää tietokannan toipumiseen.
- Tila S voidaan määrittää graafin alkuosan avulla jos graafin alkuosassa on olemassa sellainen operaatioiden järjestys, joka antaa alttiille muuttujille samat arvot kuin niillä on tilassa S.
- Jos ajatellaan graafin alkuosassa olevat operaatiot installoiduiksi ja muut operaatiot installoimattomiksi, niin alttiit muuttujat ovat kaikki ne muuttujat, jotka jotka joku installoimaton operaatio lukee ennen kuin joku toinen installoimaton operaatio kirjoittaa.
- Koska määritetty tila on tila josta voidaan toipua, täytyy kätkömuistin hoitajan pitää huoli vain siitä että se installoi operaatioita sellaisessa järjestyksessä joka takaa tilan säilymisen määritettynä.
- Määritetystä tilasta toipuminen voi olla niinkin yksinkertaista että käydään läpi logia ja suoritetaan uudelleen operaatiot, joita ei ole installoitu.
- Toisaalta voidaan joutua suorittamaan uudelleen paljon muitakin operaatioita. Jotkin toipumismenetelmät käyvätkin läpi login ja ja suorittavat uudelleen kaikki ne operaatiot jotka voidaan suorittaa uudelleen. Tämä tietysti vaatii tiedon siitä aiheuttaako installoidun operaation O uudelleen suorittaminen toisen installoidun operaation P seurausten resetoinnin, ja jos näin on niin onko mahdollista suorittaa operaatio P uudelleen silloin kun se tulee vastaan logissa.
- Oletetaan että on olemassa testi $REDO(O)$, jolla voidaan testata pitäisikö toipumisprosessin suorittaa operaatio O uudelleen.

Yleinen kätkömuistin hallinta

- Kätkömuistin hoitaja jakaa epävakaan tilan likaiseen ja puhtaaseen (dirty and clean) osiin. Muuttuja tulee likaiseen epävakaaseen tilaan kun operaatio päivittää sitä ja lähtee likaisesta epävakaasta tilasta kun se on kirjoitettu stabiiliin tietokantaan.

- Muuttujia kirjoitetaan likaiseen epävakaaseen tilaan kahdesta syystä: Ensinnäkin epävakaa tila voi olla (miltei) täysi, jolloin siellä parhaillaan olevien muuttujien pitää poistua jotta uusille saadaan tilaa. Toiseksi voi olla suotuisaa lyhentää toipumista asettamalla tarkastuspisteitä stabiiliin logiin. Koska ainoastaan installoidut operaatiot voidaan poistaa logista voi olla tarpeellista installoida joitain niiden päivityksiä ennen kuin ne poistetaan logista. Järjestelmällinen installointi aikaansaa sen että login alkuosa lyhenee ja kuitenkin stabiilin tietokannan toipuvuus säilyy.

- Kätkömuistin hallinnan keskeinen ongelma on se, että installointigraafin solmut ovat päivityksiä kun taas kätkömuistin hoitaja kirjoittaa muuttujia. Kätkömuistin hoitajan täytyykin kirjoittaa muuttujajoukkoja siten, että päivityksen atomisuutta ja päivityksien talletusjärjestystä noudatetaan.

Käytännön toipumismenetelmät

- On olemassa kolme vakavaa ongelmaa, jotka käytännön toipumismenetelmän täytyy ratkaista:
  - Atomisuus, koska useita sivuja saatetaan joutua tallettamaan yhtä aikaa.
  - Kirjoitusjärjestys, koska kätkömuistin hoitajan täytyy tietää missä järjestyksessä muuttujat tulee kirjoittaa kätkömuistista levylle.
  - REDO -testi, koska tällä testillä määritellään mitkä logitetut operaatiot suoritetaan uudelleen toipumisen yhteydessä.
- Käytännön menetelmät yleensä selviävät näistä ongelmista rajoittamalla logitettavat operaatiot syntaktisesti yksinkertaiseen muotoon.
- Useimmat käytännön menetelmistä sallivat ainoastaan sivukohtaiset operaatiot, jotka kohdistuvat vain yhteen tietokantasivuun. Tällaisista operaatioista syntyy installointigraafeja, joissa ei ole yhtään kirjoitus-kirjoitus -polkua. Tästä seuraa että operaatiot ovat aina installoitavissa toipumisen aikana, atomisuusongelma ratkeaa sillä että kukin sivu voidaan tallettaa atomisesti ja kirjoitusjärjestysongelmaa ei ole, sillä sivut voidaan tallettaa missä järjestyksessä tahansa. Ainoastaan REDO -testi tuottaa ongelmia.

Puuoperaatiot

- Installointigraafien avulla voidaan sivuorientoituneista ratkaisuista siirtyä puuoperaatio -menetelmiin.

- Puuoperaatio O lukee sivun x ja kirjoittaa sitten x:n ja mahdollisesti muita sivuja NEW:hin. NEW on joukko sivuja, joita ei ole koskaan aikaisemmin kirjoitettu, ja sivun kirjoittaminen NEW:hin poistaa sen NEW:stä. Vaikka O voikin kirjoittaa useita sivuja, kirjoitusjoukko kullekin päivitykselle U on yksi sivu. Operaatioita kutsutaan puuoperaatioiksi, koska niiden konfliktigraafi on puu.

- Puuoperaatiot ratkaisevat atomisuusongelman samoin kuin sivuorientoituneet operaatiot. Eli määrittelevät kunkin päivityksen kirjoitusjoukon yhdeksi sivuksi.

- Samoin puuoperaatiot tuottavat installointigraafeja, joissa ei ole lainkaan kirjoitus-kirjoitus -polkuja, jolloin operaatiot ovat aina tallennettavissa.

- Kätkömuistin hallinta sen sijaan on vaikeampaa.

- Vertailu sivuorientoituneiden operaatioiden ja puuoperaatioiden välillä, B-link -puun solmun halkaisu:

  – Sivuorientoitunut: Tarvitaan kaksi logitettavaa operaatiota, yksi kummallekin solmulle. Yksi on sokea kirjoitus uudelle sivulle. Toinen on uudelleenkirjoitus alkuperäiselle sivulle, joka vaatii sen että halkaisun avain logitetaan ja poistetaan ne solmut jotka ovat isompia kuin avain.

  – Puuoperaatiot: Tarvitaan yksi logitettava operaatio päivittämään sekä uusi että alkuperäinen solmu. Tämä operaatio lukee originaalin solmun vanhan arvon ja käyttäen hyväksi luvun tulosta, kirjoittaa sekä uuden että vanhan solmun. Uuden solmun sisältöä ei tarvitse logittaa kokonaisuudessaan, koska se on johdettu alkuperäisen solmun vanhasta arvosta.