VÄLIMUISTITIETOISET PUSKURIT

Markus Montonen

Sisällys

1. Johdanto
2. L1-välimuistin puskuroinnin ongelma ja ratkaisuehdotus
3. L1-välimuistin puskurioperaation kokeellinen tutkiminen
4. L2-välimuistin puskuroinnin ongelma ja ratkaisuehdotus
5. L2-välimuistin puskuroinnin ratkaisuehdotuksen tutkiminen
6. Yhteenveto

1. Johdanto

- Miksi?
  - Keskusmuistit ovat tarpeeksi suuria sisältääkseen suuria tietokantaoperaatioita
  - Prosessorit ovat nopeita suorittamaan operaatioita
  - Pullonkaulaksi on muodostunut keskusmuistin ja prosessorien välinen tiedonkulku
  - Keskusmuistista haku nopeampaa kuin levyltä, mutta hitaampaa kuin prosessorin omasta välimuistista
  - Välimuistista haku kymmenenkertaa nopeampaa
  - Prosessorien välimuistit ovat pienempiä kuin keskusmuistit
    - Keskusmuisti 8GB, L1-välimuisti 16KB/ydin, 8KB/ydin, L2-välimuisti 3MB
- Ratkaisuna välimuistitietoiset operaatiot
  - Ne eivät aina paranna suorituskykyä tarpeeksi
  - Tarvitaan puskurointia

- Hutiosuma (cache miss) on tapahtuma, jolloin tietoa ei ole saatavilla prosessorin välimuisteissa
- Tieto pitää hakea keskusmuistista, mikä kasvattaa yleensä suoritusaikaa
- Prosessorilla on useita välimuisteja
  - Pienin ja lähimpänä prosessoria on L1-välimuisti
  - Se sisältää joko tietoa tai ohjeita
  - L2- ja L3-välimuistit ovat suurempia
- Prosessori hakee tietoa ensin L1-välimuistista, sitten L2-muistista ja L3-muistista (jos on). Jos tietoa ei löydy haetaan se keskusmuistista

2. L1-välimuistin puskuroinnin ongelma

- L1-välimuisti on pienin välimuisteista
- Siellä tapahtuvat hutiosumat eivät vaikuta suorituskykyyn niin suuresti kuin muissa välimuisteissa
- Silti hutiosumien vähentämistä on tutkittu
- Zhou ja Ross esittelevät artikkelissaan puskurointioperaation
  - Operaatio perustuu tietoon ettei tietokantaoperaatiot aina mahdu L1-välimuistiin
  - Puskurointioperaatio puskuroi tietokantaoperaation, ja tallentaa L1-välimuistiin vain osoittimen
  - Parantaa suorituskykyä, vain jos tietokantaoperaatiot eivät mahdu L1-välimuistiin

Esimerkki 1

- SELECT COUNT(*) FROM item;
- Kysely tarvitsee kaksi tietokantaoperaatiota; koostaminen (P) ja tauluselaus (C)
- Koostaminen on vanhempi- (parent) ja tauluselaus lapsioperaatio (child)
- Operaatioiden suoritusjärjestys on: PCPCPCPC
- Jos operaatioiden PC yhteenlaskettu koko ylittää välimuistin koon, syntyy hutiosumia
- Puskurioperaatio muuttaa suoritusjärjestystä: PCCCCPPP
  - Lapsioperaation tuloksista tallennetaan vain osoitin L1-välimuistiin
- Nyt sama operaatio suoritetaan peräkkäin, jolloin hutiosumien määrä vähenee

L1-välimuistin puskuroinnin ratkaisuehdotus

- Tsuji, Kawashima ja Takeuchi näyttävät artikkelissaan puskurioperaation ongelman, ja esittelevät parannusehdotuksen
- Koska operaatio parantaa suorituskykyä vain tietyissä tilanteissa, tulisi sitä käyttää vain silloin
- Tutkijoiden ehdotuksessa puskurioperaation lisää kyselynoptimoija, vain kun se on tehokasta
- Kyselynoptimoija tekee syväanalyysin puskurioperaation toiminnasta ja lisää sen vain tehokkuutta lisääviin operaatioväleihin

Puskurioperaation toteutus

- Toteutettu PostgreSQL-järjestelmälle
- Uusi operaatio, joten järjestelmän omiin operaatioihin ei tarvitse tehdä muutoksia
- Suorittajamoduuli (executor module) käynnistää operaation
- PostgreSQL:n open-next-close-rajapinta
  - Open- ja close-funktiot varaavat ja vapauttavat osoittimet
  - Next-funktio säilyttää niitä

Puskurointioperaation vaikutus

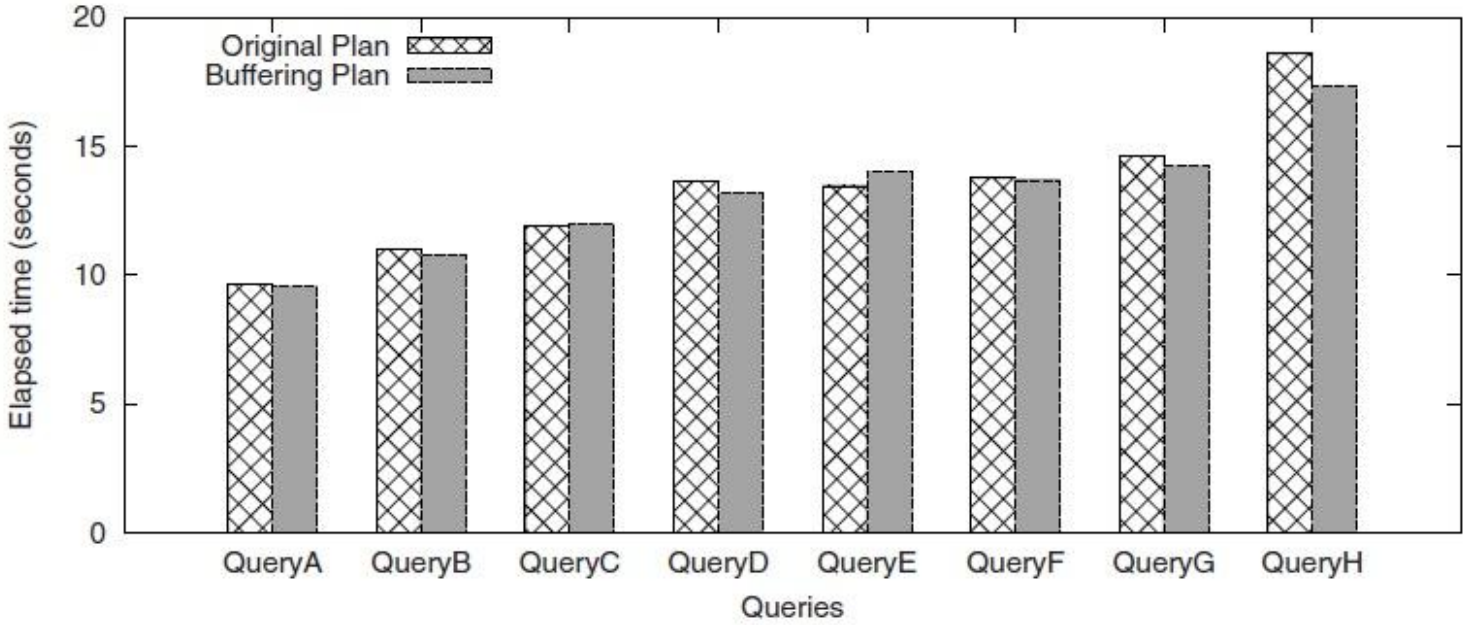
- Next-funktion suoritus
  - Jos taulukko on tyhjä, next-funktio suorittaa lapsioperaatioita kunnes se on täynnä
  - Sen jälkeen se tallentaa osoittimet taulukkoon
  - Jos taulukko ei ole tyhjä, se tarjoaa osoittimia vanhempioperaatiolle
- Puskurioperaation suoritus
  - Jos taulukko on tyhjä, lapsioperaatiota suoritetaan jatkuvasti
  - Jos taulukko ei ole tyhjä, vanhempioperaatiota suoritetaan jatkuvasti
  - Koska yhtä operaatiota suoritetaan putkeen, ei hutiosumia synny

3. L1-välimuistin puskurioperaation kokeellinen tutkiminen

- Tsuji ja kumppanit tutkivat puskurioperaation vaikutusta suoritukseen kolmella tavalla
  1. Vanhempi- ja lapsioperaatioiden kokonaiskoko
     - Kun operaatioiden kokonaiskoko on pienempi kuin L1-muisti, ei hutiosumia esiinny
  2. Puskurioperaatioiden määrä L2-välimuistissa
     - L2-välimuisti tallettaa puskuroidut puskuri- ja lapsioperaatiot
  3. Puskurioperaatioiden suoritusten määrä
     - Puskurioperaation suorituksesta aiheutuu kustannuksia

Vanhempi- ja lapsioperaatioiden koko

- Tutkijat kokeilivat suurilla ja pienillä operaatioilla
- Kooste- ja järjestämisoperaatiot
  - Select count(*), avg(point), sum(id)…
- Suoritusaika väheni melkein aina
- Järjestämisoperaatiolla 0.76% tehokkuusparannus
- Pienillä operaatioilla suoritusaika kasvoi



Puskurioperaatioiden määrä L2-muistissa

- Tarkoitus laskea sopiva puskurikoko operaatiolle
- Yksittäinen operaatio suunnittelupuussa (plan tree)
  - Neljä koosteoperaatiota, jotta välimuisti täyttyisi
  - Suorituskyky parani kun puskurinkoko x, 100 < x < 2500
  - Puskurikoon ollessa 1800 suorituskyky parani 10.86%
  - L2-välimuistin hutiosumat kasvoivat 2500 jälkeen
- Monta puskurointioperaatiota suunnittelupuussa
  - Kyselyssä kaksi liitosoperaatiota
  - Puskurikoko x > 100, suorituskyky parani huomattavasti
  - Puskurikoon ollessa 700, parannusta 19.7%
  - Alkuperäinen 14.25 sekuntia, puskuroidusti 13.75 sekuntia

Puskurioperaatioiden suoritusten määrä

- Alkuperäinen suunnitelma muuttuu heitteleväksi, jos lapsioperaation monikoita on yli 600
- Puskurointioperaatiolla L1-välimuistin hutiosumat vähenivät 300 monikon kohdalla, ja L2-välimuistin hutiosumat 500 monikon

Yhteenveto puskurioperaatiosta

- Kyselynoptimoija lisää puskurioperaation kun:
  1. Prosessoitavien monikoiden määrä on yli 500

  Ja jokin seuraavista

  1. Vanhempioperaatiossa on enemmän kuin kolme koosteoperaatiota ja lapsioperaatio on päämoduuli (main module) (paitsi järjestämisoperaatio)
  2. Vanhempi on liitosoperaatio ja lapsioperaatiot ovat päämoduuleja (paitsi järjestämisoperaatio)
  3. Vanhempi on järjestämisoperaatio ja lapset ovat päämoduuleja
- Optimoija etsii suunnittelupuusta paikat, jotka täyttävät ehdot
- Puskurille lasketaan sopiva koko (aluksi 100)
- Puskurioperaatio lisätään suunnittelupuuhun

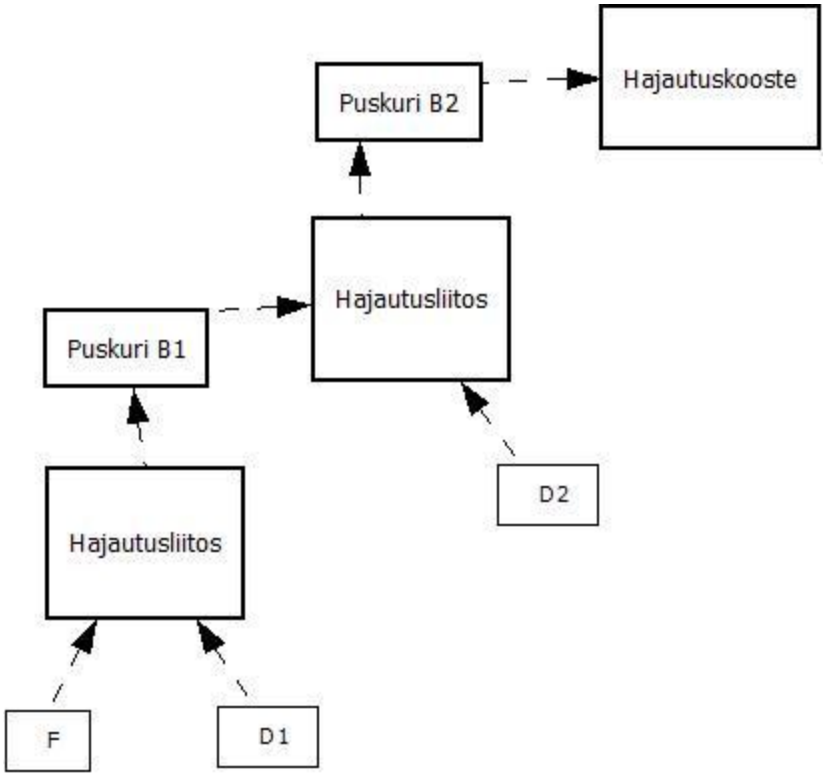
4. L2-välimuistin ongelma

- Tietokantaoperaatiot ovat usein suunniteltu käyttämään prosessorin välimuisteja hyvin
- Operaatiota suorittaessa sen tietoja jätetään välimuistiin hakuaikojen pienentämiseksi
- Useamman kuin yhden operaation samanaikainen suorittaminen aiheuttaa hutiosumia, koska toisen operaation tiedot korvaavat toisen tietoja välimuistissa
- Cieslwicz, Mee ja Ross ovat artikkelissaan esittäneet ratkaisun ongelmaan
  - Suoritetaan kaikilla prosessoreilla yhtä operaatiota kerrallaan
  - Aikataulutetaan tarpeeksi isoja viipaleita operaatioille

Ratkaisuehdotus

- Cieslewicz ja kumppanit esittävät ratkaisuksi tarpeeksi suuria suorituseriä (batch of work)
  - Kysely ehtii suorittamaan itsensä kokonaisuudessaan
- Kysely voi koostua useammasta operaatiosta
- Suorituksen nopeuttamiseksi kyselysuunnitelmaan lisätään puskureita, jotka toimivat operaation syötteenä tai tuloksena
  - Edellisen operaation tulostetta käytetään seuraavan syötteenä
  - Operaatio on valmis aikataulutukseen, jos sen syötepuskuri on ainakin puoliksi täynnä ja tulostepuskuri vähintään puoliksi tyhjä
  - Jokin operaatio on aina valmiina aikataulutukseen

Esimerkki 2

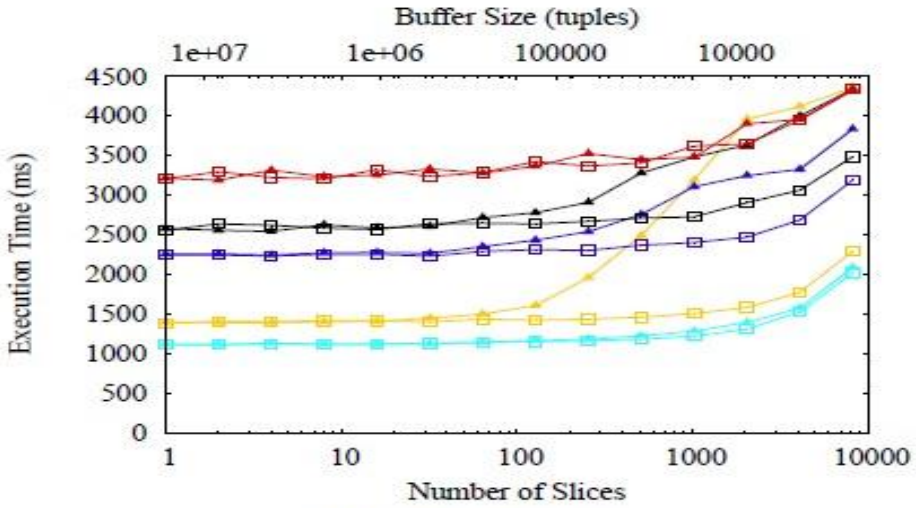

Puskuri B2
Hajautuskooste
Hajautusliitos
Puskuri B1
D2
Hajautusliitos
F
D1

5. L2-välimuistin ratkaisuehdotuksen kokeellinen tutkiminen

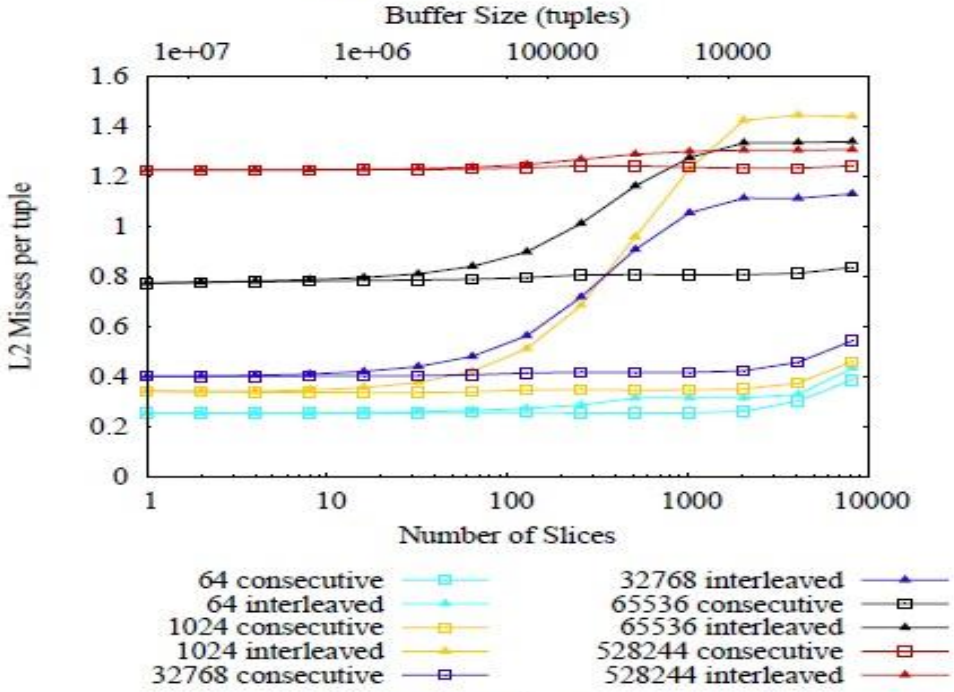
- Kokeessa käytettiin 32-säikeistä neljää prosessoria
- Yksi säie vastasi tilastoinnista, muut suorittivat määrättyä tehtävää
- Näin prosessorit voivat käyttää tehokkaasti jaettua välimuistia
- Tarkastelussa kooste- ja hajautusoperaatiot neljässä kokeessa
  1. Hutiosumien määrä kahden operaation suorituksessa samanaikaisesti
  2. Puskurinkoon vaikutus hutiosumien määrään
  3. Operaation syötteen välimuistiin jakautumisen vaikutusta suorituskykyyn
  4. Hajautusliitoksen hutiosumien määrä

Hutiosumien määrä kahden operaation suorituksessa

- Syötteenä noin 16 miljoonaa monikkoa
- Prosessointi jaettuna viipaleihin, kaikki 31 säiettä suorittivat yhtä viipaletta samanaikaisesti
- Ensimmäisessä kokeessa viipaleet suoritettiin alusta loppuun
- Toisessa niitä vaihdeltiin välillä
- Viipaleen monikoiden määrää muuteltiin
- Pienillä ja tosi suurilla monikkomäärillä ei vaikutusta
  - Pienet mahtuvat L1-välimuistiin
  - Suuret eivät mahdu L2-välimuistiin



(a) Execution time



(b) Cache Misses

Puskurikoon vaikutus hutiosumiin

- Kymmenen koostekyselyä 64- ja 1024-monikkomäärillä
- Pienet kyselyt tarvitsevat vähemmän välimuistia, joten antamalla määrätyn kokoisia puskurikokoja voidaan vaikuttaa hutiosumiin
- Jaottelulla parannusta suorituskykyyn oli 3.3% - 8.4% riippuen viipaleiden määrästä ja puskurien koosta
- Myös päällekkäin suoritettujen kyselyiden suorituskyky parani

Syötteen jakautumisen vaikutus

- Operaation vaatima välimuisti voi riippua syötteen jakelusta
- Koosteoperaation yleisemmät ryhmittelyarvot tarvitsevat enemmän paikallista muistia kuin tasaisesti jakautuneet
- Suurin osa hutiosumista tapahtuu tietyille samoille monikoille
- Vinouma (skew) kasvattaa koko ryhmittelyn hutiosumia
- Tutkijat ajoivat eri vaihtoehtoisia jakaumia koosteoperaatiolle
  - 80 % hauista kohdistui 20 %:iin tietoa
- Suurilla monikkomäärillä hutiosumia syntyi enemmän päällekkäin suoritetuista operaatioista

Hajautusliitoksen hutiosumien määrä

- Viiteavaimen mukainen liitos, missä kohdetaulun monikoiden määrää muutettiin
- Noin 150,000 monikon hajautustaulu mahtuu L2-välimuistiin
- Suoritus parani peräkkäin suoritetulla selvemmin keskisuurilla monikoilla (32768, 65536, 131072), jotka mahtuivat vielä välimuistiin
- 64 monikkomäärä mahtuu L1-välimuistiin eikä aiheuta L2-välimuistissa hutiosumia
- 524288 monikkomäärä ei mahdu L2-välimuistiin, ja tieto pitää hakea aina keskusmuistista

6. Yhteenveto

- Koska pullonkaulaksi on muodostunut keskusmuistin ja prosessorien tiedonkulku on prosessoreilla sijaitsevien välimuistien tehokkuutta parannettava
- L1- ja L2-välimuistien hutiosumia voidaan vähentää puskuroimalla tietokantaoperaatioita
- L1-välimuistissa puskurointi tapahtuu L2-välimuistiin
- L2-välimuistissa hutiosumia vähennetään suorittamalla kerralla mahdollisimman paljon samasta kyselystä, jolloin välimuistissa olevat tiedot ovat käytettäviä, ns. kuumia