hyväksymispäivä arvosana

arvostelija

Keskusmuistitietokannat – HyPer - OLTP&OLAP-yhdessä

Arto Kärki

Helsinki 9.3.2012

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Arto Kärki

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Seminaariraportti	9.3.2012	17 sivua

Tiivistelmä – Referat – Abstract

Tosiaikainen tapahtumien käsittely, OLTP (OnLine Transaction Processing) ja tosiaikainen tiedonjalostus, OLAP (OnLine Analytic Processing) asettavat erilaisen haasteen tietokanta-arkkitehtuurille. OLTP:n tulisi mahdollistaa nopea tapahtuminen käsittely. OLAP raporttien muodostaminen tarkoittaa monimutkaisten ja samalla hitaiden kyselyiden tekemistä. Nämä erilaiset tavoitteet on yleensä saavutettu siten, että tietokanta on jaettu kahdeksi erilliseksi osaksi: OLTP tietokannaksi ja niin sanotuksi datawarehouse tyyliseksi tietokannaksi (OLAP). Tämä tarkoittaa kuitenkin resurssien haaskausta: tietoa kopioidaan tietokannasta toiseen. Myös tiedon tuoreus kärsii: tietoa siirretään yleensä vain kerran päivässä. Tätä ongelmaa korjaamaan on kehitetty hybridijärjestelmiä jotka pystyvät käsittelemään sekä ajantasaiset tapahtumat että aikaa vievät raporttien muodostamiskyselyt. Seminaariraportissa tarkastellaan HyPer hybridjärjestelmää. Mistä HyPer koostuu ja kuinka se toimii.

ACM Computing Classification System (CCS):
H.2.4 Systems: Query Processing

Avainsanat – Nyckelord – Keywords

HyPer, hybrid, OLTP, OLAP, keskusmuistitietokannat

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Tämä seminaariraportti perustuu pääasiassa Alfons Kemperin ja Thomas Neumannin artikkeliin, HyPer: A Hybid OLTP&OLAP Main Memory Database System Based on Virtual Memory Snapshots [KN11].

Tietokantoja on perinteisesti käytetty ajantasaisissa tapahtumissa, OLTP (OnLine Transaction Processing). Tyypillisiä tapahtumia ovat kaupan tilaukset tai pankkitapahtumat. Nämä tapahtumat hakevat ja käsittelevät vain pientä tietokannan osaa ja voidaan näin ollen suorittaa nopeasti. Standardoidun TPC-C benchmark testitulosten mukaan tehokkaimmat järjestelmät voivat tänä päivänä käsitellä yli 100 000 myyntitapahtumaa sekunnissa [KN11]. Perinteiseen tietokantojen käsittelyyn kuuluvat myös eräajot, mutta niiden suoritus on pyritty ajoittamaan yöaikaan, jolloin ne eivät häiritse ajantasaisia tapahtumia.

Liiketoiminnan vaatimuksesta syntyi uusi tarve tietokantajärjestelmien käyttöön; liiketoimintatiedon hallinta eli business intelligence (BI). BI-sovellukset perustuvat pitkäkestoisiin ns. tosiaikaisiin tiedonjalostuskyselyihin, OLAP (OnLine Analytical Processing) , jotka käsittelevät suuria tieto määriä tuottaakseen raportteja liiketoiminnan analyytikoiden käyttöön [KN11]. Ensimmäiset OLAP-kyselyt suoritettiin käyttäen operatiivista OLTP tietokantaa. Tällöin huomattiin, että OLAP-kyselyt olivat liian raskaita ja hankaloittivat huomattavasti ajantasaisten tapahtumien käsittelyä. Siksi kehitettiin uusi OLAP-tietovarasto-arkkitehtuuri. Tapahtumatietokannan tiedot kopioidaan ja muunnetaan esim. kerran päivässä välivaraston kautta OLAP tietovarastoon. Tätä tiedon hakua, muunnosta ja lataamista kutsutaan ETL prosessiksi (Extract-Transform-Load). Koska ETL-prosessi voidaan suorittaa yleensä vain kerran yössä, aiheuttaa se tiedon vanhentumisongelman ja näin ollen liiketoiminnan raportit eivät ole ajan tasalla.

Jotta OLAP-kyselyt olisivat täysin ajantasaisia, tulee ne voida suorittaa samaan tietokantaan OLTP-kyselyiden kanssa. Tähän haasteeseen yrittää vastata HyPer keskusmuistitietokanta. Nykyinen valtava tietomäärän kasvu voi kylläkin olla ristiriidassa sen kanssa, että tapahtumatiedot voitaisiin pitää keskusmuistissa. Lähempi tarkastelu osoittaa kuitenkin että liiketoiminnan kannalta kriittisen tapahtumatietokannan koko on rajattu, mikä puoltaa keskusmuistin käyttöä. Tätä

olettamusta vahvistaa havainto, että erittäin suuren kaupallisen yrityksen, Amazonin (liikevaihto on noin 15 biljoonaa euroa), vuoden tilausrivit vievät tilaa noin 54 GB. Nämä tilausrivit ovat myynnin tärkein tietomassa. Tämä arvio ei sisällä muuta tietoa (asiakas- ja tuotetiedot), mutta kaikesta huolimatta voidaan turvallisesti olettaa että vuosittaisen myynnin tiedot mahtuvat suuren kokoluokan palvelimen keskusmuistiin [KN11].

2 Erilaisia ratkaisumalleja

Tämän luvun lähteet löytyvät pääosin Kemperin ja Neumannin artikkelista [KN11].

Keskusmuistitietokantajärjestelmien kehitys alkoi OLTP-tietokannoista. Ensimmäisten joukossa olivat TimesTen (Oraclen hankkima), P*TIME/Transact in Memory (SAP:n hankkima vuonna 2005). Solid Information Technologyn kehittämä SolidDB keskusmuistitietokanta on kehitetty Helsingissä (IBM:n hankkima). SolidDB:ssä rivitason [LW06] muistivedokset ovat peilattuja rivejä eivätkä peilattuja sivuja. SolidDB:ssä on havaittu tapahtumien läpimenon lisäyksestä (30%) ja pienentyneestä keskusmuistin käytöstä. Sivutason peilaus juontaa juurensa relaatiotietokantajärjestelmän varhaiseen kehitykseen.

Viimeaikainen keskusmuistien koon kasvu ja tarve tosiaikaiseen liiketoiminnan hallintaan on johtanut keskusmuistitietokantojen tutkinnan ja kaupallisen kehityksen elpymiseen. Viimeisimmät keskusmuistitietokannat voidaan jakaa kahteen sovellusalueeseen: OLAP ja OLTP. MonetDB on tietokantatutkimusprojekti liittyen sarakemuistiskeemaan perustuvaan OLAP keskusmuistitietokantaan. TREX on SAP:in tietokantaprojekti joka perustuu myös sarakemuistiskeemaan. Nyttemmin se tunnetaan Business Warehouse Accelerator nimellä ja on SAP:in omistuksessa. Hybridijärjestelmä on TREX:n ja P*TIME:n yhdistelmä ja pohjaa OLTP päivitysten liittämiseen ajoittain OLTP TREX tietokannan sarakemallin muistiin.

Aikaisempiin pankkitapahtumien tutkimuksiin perustuen H-Storen Kallman ja kumpp. [KKN+08] ovat analysoineet erilaisia perinteisen tietokantajärjestelmän käytöstä syntyviä ylimääräisiä kustannuksia (puskurin hallinta, lokitus, lukitus, jne.). He todistivat keskusmuistitietokannan toimivuuden joka käsittelee tapah-

tumat peräkkäisjärjestyksessä ilman synkronointi kustannuksia. VoltDB [VolO10]] on H-Storen kaupallistuma. VoltDB:n julkaisemat tehokkuustiedot pohjaavat tietokannan ositukseen koko tietokoneen klusterille.

Crescando on tutkimusprojekti Zürichissä, joka käsittelee kyselyt eränä selaamalla jaksoittain läpi koko datan vastaavalla tavalla kuin suorittamalla jatkuvia kyselyjä tietovirtaan. Lausannessa useilla Shore-tietokantajärjestelmää tutkivilla projekteilla on tavoitteena lukituksen optimointi ja lokituksen tehokkuus käytettäessä uusia moniytimisiä suorittimia. Blink ja sen kaupallistuma, IBM Smart Analytics Optimizer (ISAO), on IBM:n tuoreimpia kehitystöitä OLTP tietokantajärjestelmän laajentamiseksi muistitietokannalla OLAP kyselyitä varten.

Harizopoulos, Abadi, Madden ja Stonebraker [HAM+08] jättivät pois tietokantalukot ja sarjallistavat OLTP tapahtumat. Tämä arkkitehtuuri tekee tarpeettomaksi kalliin olioiden ja hakemistojen lukitsemis- ja salpaustarpeen, koska yksittäinen päivitystapahtuma omistaa koko tietokannan, tai oman osionsa tietokannasta. Keskusmuistiarkkitehtuurissa tyypillinen liiketapahtuma (esim. tilauksen tai maksun käsittely) kestää muutamasta korkeintaan kymmeneen mikrosekuntia. Sellaisen järjestelmän OLTP käsittelyn toimivuus todettiin tutkimus prototyypillä H-Store [KKN+08] , jossa yliopistojen tutkimusryhmää (MIT, Yale ja Brown) veti Mike Stonebaker. H-Store prototyyppi on kaupallistettu VoltDB [VolO10] nimisen yrityksen toimesta. H-Store arkkitehtuuri rajoittuu vain OLTP tapahtumiin. Jos monimutkaiset OLAP kyselyt sallittaisiin, tukkisivat ne järjestelmän, koska kaikki sen jälkeen tulevien OLTP tapahtumien tulisi odottaa pitkäkestoisen kyselyn valmistumista. Vaikka OLAP kysely valmistuisi 30 millisekunnissa, niin se lukitsisi järjestelmän niin pitkäksi aikaa, jonka aikana noin tuhat ellei useampikin OLTP kysely olisi valmistunut.

3 HyPer arkkitehtuuri

Tässä luvussa kerrotaan HyPer:n arkkitehtuurista, luku perustuu Kemperin ja Neumannin [KN11] artikkeliin.

HyPer on RISC-tyylinen tietokantajärjestelmä kuten RDF-3X (vaikkakin täysin eri tarkoitukseen tehty). HyPer:n arkkitehtuuri oli suunniteltu siten että OLTP tapahtumat ja OLAP kyselyt voidaan suorittaa samaan keskusmuistitietokantaan, ilman että kumpikaan siitä häiriintyy. Perinteisen levyperustaisen tietokantapalvelimen puskuriallas ja sivurakenne on jätetty pois. Tieto sijaitsee melko yksinkertaisessa optimoiduissa keskusmuistin tietorakenteissa virtuaalimuistissa. Näin on voitu hyödyntää käyttöjärjestelmän/suoritin tason osoitekäännöksen täysin ilman mitään lisäviittauksia. HyPer:n prototyyppi on määritelty toimimaan joko sarake- tai rivimoodissa. Vaikka virtuaalimuisti voi kasvaa suuremmaksi kuin fyysinen keskusmuisti, on tietokannan koko rajoitettu yhtä suureksi kuin on fyysinen keskusmuisti, näin on vältetty käyttöjärjestelmän hallinnoiman virtuaalimuistisivujen sivutukselta.

OLTP käsittelystä. Koska kaikki tieto on saatavilla keskusmuistista, niin tiedonsiirto-odotusta ei ole. Täten voimme luottaa yhden säikeen tekniikkaan jonka esitteli Harizopoulos, Abadi, Madden ja Stonebaker [HAM+08], siinä kaikki OLTP tapahtumat suoritetaan peräkkäin. Tämä arkkitehtuuri tekee kalliin olioiden lukitsemisen ja salpaaminen tarpeettomaksi, koska yksi päivitys tapahtuma omistaa koko tietokannan. Tämä peräkkäissuorittaminen on mahdollista vain puhtaassa keskusmuistitietokannassa jossa ei ole tarvetta levyoperaatioille. Keskusmuistiarkkitehtuurissa tyypillinen liiketapahtuma (eli tilaus tai maksutapahtuma) kestää vain noin 10 mikrosekuntia. Se tarkoittaa kymmeniä tuhansia tapahtumia sekunnissa, paljon enemmän kuin suuren kokoluokan liiketoimintasovellus vaatii.

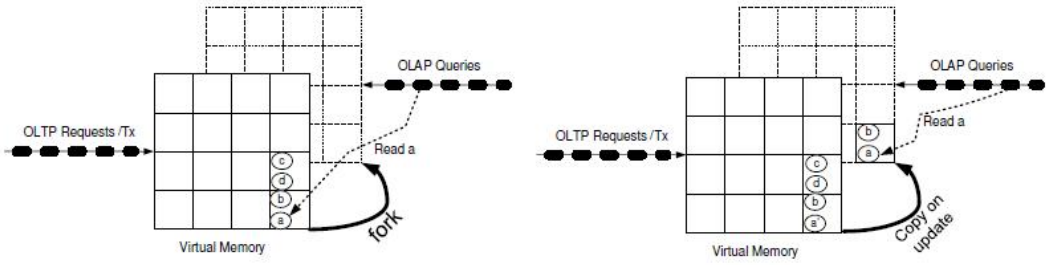
OLAP muistivedosten hallinnasta. Jos monimutkaiset OLAP kyselyt sallittaisiin OLTP:n kanssa samaan työkuormaan, niin ne tukkisivat järjestelmän, koska kaikkien seuraavien OLTP tapahtumien tulisi odottaa pitkäkestoisen kyselyn valmistumiseen saakka. Vaikka sellainen OLAP kysely valmistuisi 30 ms aikana, niin kysely lukitsisi järjestelmän niin pitkäksi ajaksi jossa tuhannet OLTP tapahtumat olisivat valmistuneet.
HyPer hyödyntää käyttöjärjestelmän toiminnallisuutta ja luo virtuaalisia muistivedoksia, kopioiduiksi prosesseiksi. Unixissa tämä tapahtuu luomalla lapsiprosessi OLTP:stä käyttäen fork () järjestelmäkutsua (jatkossa käytetään termiä fork). Jotta tapahtumien eheys säilyy, tulee fork suorittaa vain kahden sarjallistetun tapahtuman välissä, ei koskaan kesken tapahtumaa.

Kopioitu lapsiprosessi sisältää täydellisen kopion isäprosessin osoitetilasta (Kuva 1). Tätä virtuaalimuistivedosta, joka on luotu fork käskyllä, voidaan sitten käyttää OLAP kyselyissä. Muistivedos pysyy täsmälleen sellaisena kuin se ilmeni fork komennon tapahtuessa. Käyttöjärjestelmä ei kuitenkaan fyysisesti kopioi muistisegmenttejä heti. Se toteuttaa laiskaa, tarvehakuista toisinnusta (copy-on-update), (Kuva 1). Alun perin isäprosessi (OLTP) ja lapsiprosessi (OLAP) jakavat samat fyysiset muistisegmentit kääntämällä molemmat virtuaaliosoitteet osoittamaan samaan fyysiseen keskusmuistiosoitteeseen.

Toiminta voidaan kuvata myös näin: OLTP prosessi toimii koko tietokannan parissa, josta osa on jaettu OLAP modulin kanssa. Kaikki OLTP muutokset levitetään erilliseen kopioon (alueelle) nimeltä Delta, joka koostuu kopioiduista (peilatuista) tietokantasivuista. Täten OLTP prosessi luo päivitettyjen sivujen joukon tarpeen mukaan. Tämä on hyvin vastaavanlaista kuin puskurialtaan sivujen vaihto, kuitenkin, tarvepohjainen päivitettyjen sivujen toisinnus on kolme tai neljä kertaa nopeampaa, koska kestää vain 2 mikrosekuntia kopioida keskusmuistisivu, verrattuna puskurialtaan 10 millisekunnin sivukäsittelyyn. Aina ”silloin tällöin” Delta yhdistetään OLAP tietokantaan hakemalla fork komentoa käyttäen uusimmat prosessit ajantasaiseen OLAP istuntoon. Niinpä Delta on käsitteellisesti uudelleen integroitu (pää muistivedos) tietokantaan. Lisäksi on huomattava, että toisinnetut sivut ovat olemassa vain siihen saakka kunnes OLAP istunto loppuu, yleensä sekuntien tai minuuttien ajan. Perinteinen tietokantojen peilaus perustuu puhtaaseen ohjelmistomekanismiin joka säilyttää kopiot sivutasolla [Bai04] tai peilaa yksittäisiä olioita [LW06].

Muistivedokset aiheuttavat tiedon tallennuksen kuormittamisen joka on verrannollinen isäprosessin päivittämien sivujen lukumäärään. Se toisintaa Deltan (vastaamaan muutettuja sivuja) OLTP prosessin muistin tilan fork :n luoman muistivedoksen ja nykyisen OLTP prosessin muistitilan välillä. OLAP prosessit eivät koskaan muuta jaettuja sivuja, mikä tietysti olisi ongelmatonta tarvepohjaisen toisinnusmekanismin vuoksi. Kuitenkin tehokkuuden lisäämiseksi niiden tulisi allokoida väliaikaiset tietorakenteet jakamattomille keskusmuistialueille. Jos keskusmuistin määrä on vähäinen, OLAP kyselykone voi ottaa käyttöön toissijaisia tallennusvälineitä (esim. levyjä), siten vapauttamalla keskusmuistia muuhun

käyttöön. Relaation järjestäminen levyjen avulla on yksi esimerkki tästä. Kaikki OLAP kyselyt OLAP kyselyjonossa, käyttävät samaa yhtenäistä tietokannan muistivedoksen tilaa.



Kuva 1: fork komennon käyttö ja tarvepohjainen toisinnus.

Jos useita OLAP istuntoja. Koska OLAP kyselyt ovat lukevia, ne voitaisiin helposti suorittaa rinnakkain useissa säikeissä jotka jakavat saman osoitetilan. Kuitenkin synkronoinnin kuorma voidaan välttää (lukituksen ja salpauksen) kuormituksen koska OLAP kyselyt eivät jaa muuttuvia tietorakenteita. Nykyaikaiset moniytimiset tietokoneet, joilla tyypillisesti on enemmän kuin kymmenen ydintä, voivat varmasti yltää huomattavaan nopeuden lisäykseen rinnakkaisilla sisäisillä kyselyillä.

Toinen mahdollisuus hyödyntää moniytimisiä palvelimia on luoda useita muistivedoksia. HyPer:n arkkitehtuuri sallii sattumanvaraiset ajantasaiset muistivedokset. Tämä voidaan yksinkertaisesti saavuttaa ajoittaisilla (tai tarpeen mukaan) fork komennon avulla saatavilla uusilla muistivedoksilla ja näin käynnistämällä uuden OLAP kysely istunto prosessin. Selvästikin useimmat tietoalkiot eivät muutu eri muistivedosten välillä koska muistivedoksia luodaan muutaman sekunnin välein, ei minuuttien tai tuntien välein, kuten on tilanne nykyisissä tietovarasto ratkaisuissa, jotka käyttävät ETL-prosessia. Aktiivisten muistivedosten lukumäärä on rajoittamaton, koska jokainen niitä ”elää” omana prosessinaan. Prioriteettia säätämällä voimme varmistaa, että tehtäväkriittiset OLTP prosessit allokoidaan aina ytimessä, vaikka OLAP prosesseja olisi paljon ja/tai ne käyttävät monisäikeisyyttä ja näin ylittävät ydinten lukumäärän.

Muistivedos tuhotaan sen jälkeen kun istunnon viimeinen kysely on loppunut. Tämä tapahtuu yksinkertaisesti päättämällä prosessi, joka suoritti kyselyistuntoa. Ei ole tarpeellista tuhota muistivedoksia samassa järjestyksessä kuin ne luotiin. Jotkin muistivedokset voivat olla olemassa pitempään, esim. yksityiskohtaisen varaston haltuunoton vuoksi. Kuitenkin muistin kuormittaminen muistivedoksilla on verrannollinen muistivedoksen luonnista lähtien suoritettavien tapahtumien lukumäärään seuraavaan nuorempaan muistivedokseen saakka (jos sellainen on tai sitten tähän hetkeen).

Monisäikeisistä OLTP prosesseista. OLAP prosessit voidaan määritellä monisäikeisiksi ja näin paremmin hyödyntää nykyaikaiset moniytimiset tietokoneet. Laajennus voi sallia useat vain lukevat rinnakkaiset OLTP tapahtumat. Niin pian kuin luku/kirjoitustapahtuma on OLTP:n työjonon kärjessä, niin järjestelmän tilasta otetaan muistivedos ja siirretty taakse peräkkäismoodiin, kunnes jonon kärjessä ei ole enää päivittäviä tapahtumia. Todellisissa sovelluksissa on odotettavissa hyvin paljon enemmän lukevia kuin päivittäviä tapahtumia, siksi rinnakkaiskäsittely on mahdollinen, ja voi jopa lisääntyä OLTP työjonon järjestämisellä.

Nykyisessä HyPer-prototyypissä osituksen ylittävät tapahtumat vaativat poissulkevan pääsyn järjestelmään, juuri kuten HyPer:n peräkkäistoteutus. Tämä on riittävän tehokas keskitetyssä järjestelmässä, jossa kaikki partitiot sijaitsevat samassa solmussa. Kuitenkin, jos solmut on hajautettu eri klustereihin, mikä edellyttää kaksi-vaiheista sitoutumisprotokollaa moniositus tapahtumilta, kehittyneempi synkronointi on hyödyllisempää.

OLAP muistivedokset voidaan fork komentoa käyttäen tehdä kuten aiemmin, poikkeuksena se että kaikkien säikeitten tila tulee ottaa talteen ennenkuin tämä voidaan tehdä tapahtumien yhtenäisyys säilyttäen. Tätä vaatimusta voidaan höllentää muuntamalla toimintayhtenäiset muistivedokset tapahtumayhtenäisiksi peruutuslokin avulla. OLAP kyselyt voidaan laatia koskemaan kaikkia osioita ja jaettua tieto-osiota, mikä on tarpeellista monikäyttäjä sovelluksissa hallinnollisiin tarpeisiin.

Tietokannan osittaminen voidaan hyödyntää hajautetussa järjestelmässä mikä allokoi yksityisiä osioita eri solmuille tietokoneen klusterissa. Luetuimmat, jaetut

osiot voidaan toisintaa kaikissa solmuissa. Siten ositus-rajoitetut tapahtumat voidaan kuljettaa vastaavaan solmuun ja ajaa rinnakkain ilman mitään synkronointi kuormaa. Synkronointia tarvitaan osituksen ylittäviä tapahtumia varten ja synkronoitujen muistivedosten ottamiseksi kaikista solmuista.

4 Tapahtumien semantiikka ja toivutus

Luvun tiedot perustuvat Kemperin ja Neumannin artikkeliin [KN11].

Muistivedosten eristäminen OLAP kyselyistunnoissa. Muistivedoksen eristämisessä tapahtuma/kysely näkee jatkuvasti tapahtuman tietokantatilan yhtenäisenä, kuten se oli silloin ennen kuin tapahtuma alkoi. On olemassa erilaisia mahdollisuuksia toteuttaa kyseinen muistivedos, kun tietokannan muutokset tapahtuvat samaan aikaan: peruutus, versiointi, peilaus ja virtuaaliset muistivedokset.

Tapahtumayhtenäinen arkistointi. Virtuaalimuistin muistivedoksia voidaan hyödyntää luomalla varmistusarkistot koko tietokannasta vakaalle tallennusvälineelle. Tyypillisesti arkisto kirjoitetaan nopean varmistusverkon kautta (1 – 10 GB/s) sille omistetulle tallennuspalvelimelle samassa konekeskuksessa. Säilyttääkseen tämän siirtonopeuden, tallennuspalvelimen tulee käyttää useita (yli 10) levyjä vastaamaan siirtonopeutta.

OLTP tapahtumien synkronointi. Yksisäikeisessä moodissa OLTP tapahtumat eivät tarvitse synkronointi mekanismia koska ne omistavat koko tietokannan. Monisäikeisessä moodissa on kaksi tapahtumatyyppiä: ositus-rajoitetut tapahtumat voivat lukea ja päivittää tietoa niiden omissa osioissaan ja voivat myös lukea tietoa jaetuista osioista, osituksen ylittävät tapahtumat voivat päivittää jaettua tietoa ja hakea (lukea tai päivittää) toisen osion tietoja.
Osituksen ylittävät tapahtumat ovat harvinaisia koska jaetun tiedon päivitykset tapahtuvat harvoin ja ositus päätellään niin että tapahtumat normaalisti käsittelevät vain omia tietojaan. Talletettujen proseduurien luokittelu OLTP työkuormassa tehdään automaattisesti perustuen toteutuskoodiin ja niiden käynnistysparametreihin. Jos suorituksen aikana osoittautuu että tapahtuma oli virheellisesti luokiteltu ositus-rajoitetuksi, se peruutetaan ja lisätään uudestaan OLTP työkuormajonoon ”osituksen ylittäjänä”. HyPer järjestelmä sallii enintään yhden osi-

tus-rajoittuneen tapahtuman per ositus rinnakkain. Siksi ei ole tarvetta millekään lukitukselle tai salpaamiselle koska osituksilla ei ole ylivuotavia tietorakenteita ja jaettu tieto on vain lukukäytössä.

Tapahtumien kestävyys vaatii että kaikki sitoutuneiden tapahtumien vaikutukset tulee tallentaa virheen jälkeen. Tämän saavuttamiseksi HyPer käyttää klassista peruutuslokia lokittamalla talletettujen proseduurien parametrit jotka edustavat tapahtumia. Perinteisessä tietokantajärjestelmässä looginen lokitus on ongelmallista koska järjestelmän kaatumisen jälkeen tietokanta voi olla ei-yhtenäisessä tilassa. Näin ei voi käydä HyPer:ssä, koska uudelleenkäynnistys tehdään tapahtumayhtenäisestä arkistosta. On tärkeää kirjoittaa nämä loogiset lokitietueet siinä järjestyksessä kuin ne suoritettiin, jotta tietokanta voidaan palauttaa oikein. Yksisäikeisessä OLTP konfiguraatiossa tämä on helposti saavutettavissa. Monisäikeisessä järjestelmässä vain osituksen ylittävien tapahtumien lokitietueet tulee on järjestyksessä suhteessa kaikkiin tapahtumiin, kun taas ositus-rajoittuneet tapahtumalokitietueet voidaan kirjoittaa rinnakkain ja näin sarjallistetaan vain per osio.

Tapahtumien atomisuuden vuoksi kaikkien epäonnistuneiden tapahtumien aiheuttamat muutokset tulee poistaa tietokannasta. Vain eksplisiittisesti keskeytetyt tapahtumat tulee huomioda. Ns R3-palautus vaatii että ne tapahtumat jotka olivat aktiivisia kaatumisen aikana)perutaan, ei ole tarpeen HyPer:ssa, koska tietokanta on ei-pysyvässä muistissa ja loogiset peruutuslokit on kirjoitettu vain silloin kun tapahtuman onnistunut sitoutuminen on taattu. Lisäksi, arkistoitu tietokantavarmistus, joka toimii palautuksen lähtökohtana, on tapahtumine osalta yhtenäinen, siksipä, peruutusoperaatioita ei tarvita. Seurauksena tästä on että peruutuslokitusta tarvitaan vain aktiiveille tapahtumille (kaikille aktiiveille monisäikeisille tapahtumille) ja voidaan säilyttää vain ei-vakaassa muistissa.

Yhtenäisten muistivedosten siivoaminen. Peruutuslokia voidaan käyttää luomaan toimenpide yhtenäisestä virtuaalimuistivedoksesta joka oli luotu joidenkin tapahtumien ollessa vielä kesken, tapahtumayhtenäinen muistivedos. Tämä on hyödyllistä monisäikeisessä OLTP järjestelmässä koska silloin vältetään täysin tapahtumakäsittelyn tilan varmistaminen (quiesce). Kun OLAP prosessiin ja siihen liittyvään virtuaalimuistivedokseen on kohdistettu fork , niin peruutusloki tietueet

asetetaan muistivedoksen tilaan, ajallisesti päinvastaisessa järjestyksessä. Koska peruutuslokipuskuri kuvastaa kaikkia aktiivisten tapahtumien vaikutuksia (fork:n oton aikana), ja vain niitä, tuloksena saatu muistivedos on tapahtumayhtenäinen ja kuvastaa tietokannan tilaa ennen fork:n aikaisten tapahtumien käynnistymistä.

Toipuminen järjestelmävirheestä. Palautumisprosessi perustuu kestävään tietokanta-arkiston tallennusjärjestelmään ja peruutuslokiin. Palautus voidaan aloittaa tuoreimmasta täydellisestä arkistosta, joka palautetaan keskusmuistiin. Sitten asetetaan paluuloki aikajärjestyksessä, alkaen muistivedoksen tehneen fork:n jälkeisestä ensimmäisestä paluulokimerkinnästä. Koska arkisto voidaan palauttaa 10GB/s siirtonopeudella ja paluuloki voidaan levittää 100 000 tapahtuman sekuntivauhdilla, kestää noin 100 GB kokoisen tietokannan palautus vain muutamia minuutteja, jos varmistus arkistoitaan tunneittain. Jos tämä palautusaika on liian pitkä, voidaan silloin käyttää toisinnettuja HyPer-koneita. Virhetilanteessa tehdään vaihto toimivaan koneeseen hyvin nopeasti.

5 Tehokkuuden arviointi

Luku perustuu Kemperin ja Neumannin HyPer tehokkuusarviointiin [KN11].

HyPer-prototyypin tehokkuusarvio perustuu TPC-CH nimiseen suorituskykytestiin, joka on yhdistelmä kahdesta standardisoidusta TPC suorituskykytestistä. TPC-C testi on suunniteltu OLTP tietokantajärjestelmän tehokkuuden arviointiin ja TPC-H OLAP kyselyiden tehokkuuden arviointiin. Molemmat suorituskykytestit simuloivat kaupan tilausjärjestelmää (tilaus, maksu, toimitus).

TPC-CH testin tietokantaskeemassa on kohteiden kardinaliteetit sekä kardinaliteettien (min, max). Kardinaliteetit vastaavat tietokannan lähtötilaa TPC-C testin alussa ja kasvattaa (varsinkin tilausten ja tilausrivien määrää) testin aikana. Tietokannan lähtötila voidaan skaalata lisäämällä tavaratalojen lukumäärää, sekä lisäämällä asiakkaiden lukumäärää, tilausten ja tilausrivien lukumäärää. Alkuperäinen TPC-skeema, joka on säilytetty koskemattomana, koostuu yhdeksästä relaatiosta.

HyPer:n testiin on lisätty kolme relaatiota TPC-H testistä jotta kaikki kaikki 22 kyselyä voidaan toteuttaa järkevästi. Nämä relaatiot ovat: toimittaja (10000 toimittajaa), valtio ja alue (62 valtiota ja 5 aluetta). TPC-C OLTP tapahtumat sisältävät tilausten syötön ja toimitukset, maksujen tallennuksen, tilausten tilan tarkastuksen ja tavaratalojen varaston tarkkailun. Kaikki nämä tapahtumat, sisältäen myös lukevan tapahtuman Tilauksen-Tila ja Varaston-Tila suoritettiin sarjallisesti HyPerin OLTP työjonossa.

VoltDB:n testissä on käytetty VoltDB:n asetuksia, mitkä sisältävät joitakin muutoksia testiin. Testin tapahtumien joukko koostuu kolmesta päivittävästä tapahtumasta (uusi-tilaus, maksu ja toimitus) jotka kuvastavat tyypillisiä liiketoiminta-proseduureja. Järjestelmä ylläpitää tasapainoista tietokannan tilaa, eli jokainen tilaus on lopulta maksettu ja toimitettu. Järjestelmän tehokkuus on yleensä määritelty uusien tilaustapahtumien määrinä jotka käsitellään, samaan aikaan tietysti kaikki muut tapahtumat tulee käsitellä. Vertaillaksemme tuloksiamme muihin järjestelmiin ilmoitamme myös kaikkien viiden tapahtuman yhteenlasketut summat per sekunti (Kuva 2.).

OLAP-kyselyt. Täydellistä OLTP&OLAP testiä varten TPC-H testin 22 kyselyä on muokattu TPC-CH skeemaa varten. Uudelleenmuotoilulla on varmistettu että kyselyt säilyttävät niiden semantiikan (liiketoiminnan näkökulmasta nähtynä) sekä niiden syntaktisen rakenteen. OLAP kyselyt eivät hyödy tietokannan osituksesta koska ne kaikki vaativat tiedon selaamista yli kaikkien ositusrajojen.

Erilaisten HyPer kokoonpanojen tehokkuudesta. Kaikki testit on tehty TPC-C kokoonpanolla, jossa on 12 tavarataloa. Niinpä lähtötilanteen tietokanta sisältää 360 000 asiakasta, joilla on 3.6 miljoonaa tilausriviä, tämä tekee noin 1 GB verran tietoa. Uudelleenkäytettävyyden vuoksi kaikki kyselyistunnot aloitettiin (fork-ed) testin alussa (eli 12 tavaratalon moodissa) ja 22 kyselyä ajettiin sisään, muunnettuina, jotta niillä olisi vaikutus välimuistiin, viisi kertaa kussakin kyselyistunnossa. Niinpä jokainen OLAP istunto/prosessi suoritti 110 kyselyä peräkkäin. Ilmoitamme mediaanin kunkin kyselyn vasteajasta. Nämä kyselyt suoritettiin rinnakkain sekä yksi- että monisäikeisissä OLTP prosesseissa (Kuva 2.).
HyPer voidaan konfiguroida rivitason tai saraketason varastoksi. OLTP:lle emme havainneet mitään erityistä eroa tehokkuudessa, kuitenkin, OLAP kyselyt nopeu-

tuivat huomattavasti sarakemallin varastoskeemalla. Niinpä OLTP ja OLAP tehokkuus ilmoitetaan vain sarakemallista.

HyPer testi kuten myös MonetDB kyselytesti ajettiin tavallisella palvelimella, seuraavin määrityksin:

- Dual Intel X5570 Qaud-Core-CPU, 8MB Cache
- 64GB RAM
- 16 300GB SAS-HD (ei testissä)
- Linux käyttöjärjestelmä RHEL 5.4
- Hinta: 13.886 euroa (ale hinta yliopistolle)

Myös VoltDB:n on listalla OLTP tehokkuuden vertailun vuoksi, mutta sen arvot on otettu VoltDB:n tuote-esitteestä [VolO10] ja keskustelusivustolta [VolB10]. VoltDB:n testi oli suoritettu samantapaisella laitteistolla (dual- quad Xeon CPU Dell R610 servers). Suurin ero on, että HyPer:n testi on ajettu yksittäisellä palvelimella, kun taas VoltDB skaalautui kuudelle solmulle. Lisäksi HyPerin testi suoritettiin lokittamalla paluulokit toiselle tallennuskoneelle kun taas VoltDB oli ajettu ilman mitään lokitusta tai toisinnusta.
HyPerin yksittäisellä vakiopalvelimella saavutetut läpimeno tulokset vastaavat VoltDB:n 6-solmun klusterikoneella saavuttamia tuloksia. Kuten VoltDB:n julkaisut osoittavat [VolO10], nämä tulokset vastaavat parhaita julkaistuja suuren mittakaavan levyperustaisen tietokanta konfiguraation TPC-C tuloksia. HyPer OLTP:n läpimenot saavutettiin jopa silloin kun yksi, kahdeksan tai kolme rinnakkaista OLAP prosessia teki jatkuvasti OLAP kyselyitä rinnakkain OLTP työkuormaan (Kuva 2.).

HyPerin kyselyn vasteaikoja vertailtaessa MonetDB:hen, huomataan, että nämä kaksi kyselymoottoria ovat yhtä tehokkaita. Teknologinen kehitys sallii pian usean TB:n kokoiset keskusmuistit. Neljän kilon oletussivukokoa varten, 1 TB:n tietokannan tulee hallita sivutaulu, jossa on on 250 miljoonaa merkintää, sivutaulun viedessä tilaa noina 4GB.

Query No.	HyPer configurations: one query session (stream) single threaded OLTP — OLTP throughput	Query resp. times (ms)	HyPer configurations: 8 query sessions (streams) single threaded OLTP — OLTP throughput	Query resp. times (ms)	HyPer configurations: 3 query sessions (streams) 5 OLTP threads — OLTP throughput	Query resp. times (ms)	MonetDB no OLTP 1 query stream Query resp. times (ms)	VoltDB no OLAP only OLTP results from [18]
Q1		67		71		71	63	
Q2		163		233		212	210	
Q3		66		78		73	75	
Q4		194		257		226	6003	
Q5		1276		1768		1564	5930	
Q6		9		19		17	123	
Q7		1151		1611		1466	1713	
Q8		399		680		593	172	
Q9		206		269		249	208	
Q10		1871		2490		2260	6209	
Q11		33		38		35	35	
Q12		156		195		170	192	
Q13		185		272		229	284	
Q14		122		210		156	722	
Q15		528		1002		792	533	
Q16		1353		1584		1500	3562	
Q17		159		171		168	342	
Q18		108		133		119	2505	
Q19		103		219		183	1698	
Q20		114		230		197	750	
Q21		46		50		50	329	
Q22		7		9		9	141	
	new order: 56961 tps; total: **126576** tps		new order: 29359 tps; total: **65269** tps		new order: 171384 tps; total: **380868** tps			**55000** tps on single node; **300000** tps on 6 nodes

Kuva 2: tehokkuusvertailu.

6 Yhteenveto

HyPer arkkitehtuuri perustuu virtuaalimuistin tukemiin useiden kyselyistuntojen liiketoiminnallisen tiedon muistivedoksiin. OLTP tapahtumat ja OLAP kyselyt, käyttävät samaa tietokantaa ilman että ne häiritsevät toisiaan. Muistivedoksen ylläpito ja käsittelyn tehokkuus, joka ilmenee OLTP:n läpimenoaikoina ja OLAP kyselyiden vasteaikoina, saavutetaan laitteiston tukeman tarvepohjaisen toisinnuksen avulla ja samalla ylläpitäen muistivedoksen yhtenäisyys. Toisinnusta kaipaavien jaettujen sivujen havaitseminen tapahtuu tehokkaasti käyttöjärjestelmän toimesta muistinhallintayksikön avustuksella. Samanaikainen liiketoiminnallinen työkuorma ja liiketoimintatiedon hallinnan (BI) kyselyiden prosessointi käyttävät moniytimistä arkkitehtuuria tehokkaasti ilman rinnakkaisuuden ongelmia, koska ne on erotettu virtuaalimuistivedokseksi.

HyPer vaikuttaa lupaavalta arkkitehtuurilta. Sen testitulokset osoittavat sen pystyvän hyviin tuloksiin vertailussa muihin keskusmuistitietokanta ratkaisuihin. Nykyaikainen liiketoimintatiedon hallinta on varmasti kiinnostunut HyPer:n tarjoamasta potentiaalista.

Lähteet

Bai04 S. Bailey, “Us patent 7389308b2: Shadow Paging,” 17. Juni 2008, filed: 30. Mai 2004, granted to Microsoft.

HAM+08 Stavros Harizopoulos, Daniel J. Abadi, Samuel Madden, and Michael Stonebraker. 2008. OLTP through the looking glass, and what we found there. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (SIGMOD '08). ACM, New York, NY, USA, 981-992.

KKN+08 Robert Kallman, Hideaki Kimura, Jonathan Natkins, Andrew Pavlo, Alexander Rasin, Stanley Zdonik, Evan P. C. Jones, Samuel Madden, Michael Stonebraker, Yang Zhang, John Hugg, and Daniel J. Abadi. 2008. H-store: a high-performance, distributed main memory transaction processing system. *Proc. VLDB Endow.* 1, 2 (August 2008), 1496-1499.

KN11 Alfons Kemper and Thomas Neumann. 2011. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering* (ICDE '11). IEEE Computer Society, Washington, DC, USA, 195-206.

LW06 Antti-Pekka Liedes, Antoni Wolski, "SIREN: A Memory-Conserving, Snapshot-Consistent Checkpoint Algorithm for in-Memory Databases," Data Engineering, International Conference on, p. 99, 22nd International Conference on Data Engineering (ICDE'06), 2006

VolB10 VoltDB, VoltDB TPC-C-like Benchmark Comparison-Benchmark Description, https://community.voltdb.com/node/134, May 2010.

VolO10 VoltDB, “Overview,” http://www.voltdb.com/_pdf/VoltDBOverview.pdf, March 2010.