Atte Päärni

PILVISOVELLUKSILLE SOPIVIA TIETOKANNAN HALLINTAJÄRJESTELMIÄ

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2012

TIIVISTELMÄ

Pilvisovelluksille sopivia tietokannan hallintajärjestelmiä
Jyväskylä: Jyväskylän yliopisto 2012, 35 s.
Tietojärjestelmätiede, kandidaatin-tutkielma
Ohjaaja: Mauri Leppänen

Tässä tutkielmassa on tarkoitus selvittää, millaisia vaatimuksia pilvisovelluksilla on tietokannan hallintajärjestelmille ja miten ehdotetut järjestelmät täyttävät nämä vaatimukset. Tutkimus suoritetaan kirjallisuuskatsauksena, joka perustuu tieteellisiin julkaisuihin ja tutkimusartikkeleihin.

Aluksi käsitellään pilvimaailman käsitteitä, ominaisuuksia ja haasteita tietokannan hallintajärjestelmille. Viime vuosina on kehitetty lukuisia uusia pilvisovelluksille suunnattuja tietokantatuotteita, joista suurin osa kuuluu termin NoSQL alle. NoSQL-tietokantatuotteet eivät perustu relaatioihin, eivätkä käytä pelkästään SQL-kieltä tiedon käsittelemiseen. Nämä ovat pilviympäristössä horisontaalisesti skaalautuvia tietokannan hallintajärjestelmiä. Tutkielmassa tarkastellaan tarkemmin kolmeen NoSQL-tietokannan hallintajärjestelmän (Google Bigtable, Amazon S3, Yahoo! PNUTS) tietomallia ja järjestelmän arkkitehtuuria. Tutkielmassa selvitetään minkälaisia ominaisuuksia nämä tuotteet tarjoavat ja miten ne vastaavat pilvimaailman haasteisiin. NoSQL-tietokantatuotteiden lisäksi tarkastellaan ElasTraS-tietokantakonseptia, joka pyrkii yhdistämään relationaalisen tietomallin monipuolisia ominaisuuksia, ACID-tapahtumanhallintaominaisuuksia, sekä horisontaalista skaalautuvuutta.

Tutkielmassa nähdään, miten NoSQL-tietokantatuotteen tukevat ACID-ominaisuuksien sijasta BASE-ominaisuuksia, jolloin tietokanta on aina saatavilla, mutta se ei ole jatkuvassa ristiriidattomassa tilassa. Ideana tässä on monien verkkosovelluspalveluiden tiukat vaatimukset vasteajoista, jolloin tiedon saatavuutta on jouduttu parantamaan tiedon ristiriidattomuuden kustannuksella. Yritysten liiketoiminnassa käytettäville tietokannoille jatkuva ristiriidattomuus on kuitenkin pakollista.

Avainsanat: pilvilaskenta, NoSQL, tietokannan hallintajärjestelmä, CAP-teoreema, BASE-ominaisuudet, ACID-ominaisuudet, Bigtable, S3, PNUTS, ElasTraS

ABSTRACT

Database Managment Systems for Cloud Applications
Jyväskylä: University of Jyväskylä 2012, 35 p.
Information Systems Science, Bachelor's Thesis
Supervisor: Mauri Leppänen

The purpose of this study is to describe requirements presented for database management systems to be used in cloud applications, and analyze how existing database management systems meet those requirements. Research is based on literature review.

First, concepts and characteristics of cloud computing are described, followed by the presentation of challenges of database management in cloud environment. In recent years, numerous new database management systems for cloud applications have emerged. Most of them belong under a suit called NoSQL database products. These are not based on relational data model and do not use only SQL language as the only language. These database management systems scale horizontally in cloud environment. In this study the properties of three NoSQL database management systems (Google Bigtable, Amazon S3, Yahoo! PNUTS) are described in terms of their data model and system architecture. We also examine how these products meet the requirements of cloud applications. Besides NoSQL database management systems, we describe the ElasTraS database management system that aims to combine relational data model, ACID properties and horizontal scalability.

This study shows that the NoSQL database management systems prefer the BASE properties over ACID properties. Through the BASE properties they aim to ensure that the database is always available, although not in continuous consistent state. The reason for this aim is that web-applications generally have more stringent requirements for response time, and not for consistency that is mandatory for databases in business operations.

Keywords: cloud computing, NoSQL, database management system, CAP theorem, BASE properties, ACID properties, Bigtable, S3, PNUTS, ElasTraS

KUVIOT

SISÄLLYS

1 JOHDANTO

Pilvilaskenta voi muuttaa merkittävästi informaatioteknologiateollisuutta. Pilvipalvelujen tuottajat tarjoavat tietokone-palveluja yleishyödykkeenä samaan tapaan kuin, esimerkiksi sähköä tai vettä on jo kauan tarjottu. Tällöin asiakas maksaa käyttämistään resursseista jatkuvan laskutuksen periaatteen mukaisesti. Internet-ohjelmistopalveluja kehittävät eivät näin tarvitse enää suurta alkupääomaa palvelujensa saattamiseksi saataville, eikä niiden tarvitse arvioida palvelun tarkkaa käyttöastetta etukäteen. Näin pystytään välttämään tilanne, jossa palvelua ei pystytä tarjoamaan kaikille palvelimen ylikuormittumisesta johtuen tai palvelinta käytetään alikuormitettuna käyttöasteen yliarvioimisesta johtuen. Lisäksi suuret osiin jaettavat prosessit voidaan suorittaa pilvessä niin nopeasti kuin ohjelma pystyy skaalautumaan. 1000 palvelimen tunnin käyttö maksaa saman verran kun yhden palvelimen käyttö 1000 tunnin verran. (Armbrust, Fox, Griffith, Joseph ym.,2009)

Pilvimaailma luo uusia haasteita ohjelmistoille. Toimiakseen nopeasti ja kustannustehokkaasti niiden pitää pystyä skaalautumaan nopeasti käyttöasteen kasvaessa ja laskiessa (Armbrust ym., 2009). Tämä skaalautuvuus alaspäin on uusi haaste myös tietokannan hallintajärjestelmille (Feuerlicht, 2010). Tietokannan hallintajärjestelmän pitää pystyä tarvittaessa skaalautumaan horisontaalisesti tuhansille eri palvelimille ja samalla pitämään huolta tiedon saatavuudesta ja yhteneväisyydestä (Feuerlicht & Pokorný, 2011). Vastaamaan näihin haasteisiin on kehitetty useita kaupallisia ja avoimeen lähdekoodiin perustuvia NoSQL–tietokantatuotteita (Feuerlicht ym., 2011).

NoSQL on sateenvarjo, jonka alle kuuluu useita tietokantatuotteita kuten esimerkiksi Googlen Bigtable (Chang, Dean, Ghemawait, Hsieh ym., 2008), Amazonin S3 (Brantner, Florescu, Graf, Kossmann ym., 2008), Yahoon PNUTS (Cooper, Ramakrishnan, Srivastava, Silberstein ym., 2008) ja Facebookin Cassandra (Lakshman & Malik, 2010). NoSQL - tietokannat eivät ole relaatioihin perustuvia, eivätkä käytä SQL-kieltä tiedon käsittelemiseen. Tästä johtuen NoSQL–tietokannoista puuttuu perinteisten relaatiotietokannan hallintajärjes-

telmien tapahtumanhallintaominaisuuksia. Tätä voidaan pitää vaihtoehtoiskustannuksena tiedon paremmalle saatavuudelle. (Feuerlicht & Pokorný, 2011)

Monille Internet-sovelluksille, kuten pankki- ja finanssialan sovelluksille, takeet tiedon eheydestä ovat pakollisia tietokannan hallintajärjestelmän ominaisuuksia. Tästä syystä kaivattaisiin tietokannan hallintajärjestelmää, joka yhdistäisi NoSQL–tietokantojen tiedon saatavuuden ja perinteisten relaatiotietokannan hallintajärjestelmien tapahtumanhallintaominaisuudet (Feuerlicht ym., 2011).

Armbrust, Fox, Griffith, Joseph ym. (2009) listasivat kymmenen tärkeintä pilvilaskennan haastetta. Näistä yksi oli skaalautuva tiedon varastointi, mikä esiteltiin avoimena tutkimusongelmana. Jaroslav Pokorný (2010) esitti kolmannen vuosituhannen tietokantojen trendejä ja tutkimusongelmia ja mainitsi ajankohtaisiksi tutkimusongelmiksi muun muassa ei-relaationaalisten datamallien kehittämisen, sekä ristiriidattomuuden vaihtamisen saatavuuteen tuhansille eri koneille skaalautuvuuden parantamiseksi. Pokorný myös esitti, että pilvilaskenta on yksi dataprosessoinnin muoto, joka on tuottanut viime aikoina uusia vaatimuksia ja käytännön tietokantasovelluksia, ja tarjoaa silti paljon tutkimusmahdollisuuksia. Pilvisovelluksille sopivan tietokannan hallintajärjestelmän löytäminen on siis ajankohtainen ja avoin tutkimusongelma (Armbrust ym., 2009; Pokorný, 2010).

Tämän tutkielman tarkoituksena on selvittää, millaisia pilvisovelluksille soveltuvia tietokantahallintajärjestelmiä on kehitetty. Tämä tutkielman pääkysymys voidaan jakaa seuraaviin alakysymyksiin:

- Mitä ominaisuuksia pilvisovelluksen tietokannan hallintajärjestelmältä kaivataan?
- Minkälaisia NoSQL-vaihtoehtoja on olemassa ja miten ne vastaavat pilvisovellusten vaatimuksiin?
- Onko mahdollista yhdistää NoSQL–tietokantatuotteiden tiedon saatavuutta ja relaationaalisten tietokannan hallintajärjestelmien tapahtumanhallintaominaisuuksia?

Tutkielman tarkoituksena on siis selvittää, millaisia vaatimuksia pilvisovelluksilla on tietokannan hallintajärjestelmälle, millaisia vaihtoehtoja NoSQL–tietokannat tarjoavat ja miten ne vastaavat pilvimaailman haasteisiin. Tarkoituksena on tutustua keskeisimpiin NoSQL–tietokantatuotteisiin ja selvittää miten ne sopivat pilvimaailman sovelluksille, minkälaisia ominaisuuksia ne tarjoavat ja mitä ominaisuuksia vielä kaivattaisiin. Tarkoitus on myös tehdä lyhyt katsaus mahdollisiin hybridi-järjestelmiin, jotka yhdistäisivät perinteisten relaatiotietokannan hallintajärjestelmien ja NoSQL–tietokantatuotteiden ominaisuuksia. Tutkimustuloksia voi hyödyntää verkkosovelluspalvelujen kehittäjät, jotka etsivät pilvisovellukselleen sopivaa tietokannan hallintajärjestelmää, tai ohjelmistokehittäjät, jotka harkitsevat pilvilaskennan käyttämistä ohjelmistonsa laitteistoalustana. Tutkimus suoritetaan kirjallisuuskatsauksena.

Tutkielma koostuu neljästä luvusta. Luvussa 2 käsitellään pilvimaailman käsitteitä, pilvilaskennan keskeisiä ominaisuuksia, sekä tietokannan hallintajär-

jestelmien haasteita pilvimaailmassa. Luvussa 3 kuvataan NoSQL-tietokantakonsepti, joka pyrkii vastaamaan pilvimaailman haasteisiin. Luvussa esitellään myös keskeisempiä NoSQL-tietokantasovelluksia ja pohditaan, miten ne vastaavat pilvimaailman haasteisiin. Luvussa 4 kerrotaan, miksi NoSQL-tietokannat eivät sovellu kaikille sovelluksille, ja esitellään ElasTraS-tietokantakonsepti, joka pyrkii tarjoamaan relaationaalisen tietokannan ominaisuuksia pilvimaailmassa. Tutkielman päätteeksi esitetään yhteenveto.

2 PILVILASKENTA

Tässä luvussa kuvataan mitä pilvilaskenta tarkoittaa, miten se on muuttanut informaatioteknologiateollisuutta ja minkälaisia haasteita se luo Internet-ohjelmistokehittäjille. Lopuksi esitellään vaatimuksia tietokannan hallinnalle pilvilaskennassa.

2.1 Pilvimaailman käsitteitä

Vaikka pilvilaskennasta on jo vuosia puhuttu, kirjoitettu blogeissa ja se on ollut useiden työpajojen, konferenssien ja lehtien otsikoissa, on ollut epäselvää mitä se tarkalleen tarkoittaa ja milloin se on hyödyllistä (Armbrust ym., 2009). Pilvilaskenta tuli terminä laajalti käyttöön vuosituhannen vaihteessa, jolloin siitä käytettiin aluksi IT-palvelukonseptin mukaista termiä SaaS (Software as a Service), eli verkkopalvelusovellus (Korpinen, 2011). Nykyään pilvi, pilvilaskenta ja pilvipalvelut termeinä ovat laajasti käytettyjä ja tarkoittavat monia eri asioita (Korpinen, 2011). Seuraavaksi määritellään keskeiset pilvimaailman käsitteet.

Pilven määritelmä pohjautuu tässä tutkielmassa yleisesti hyväksytyn ja viitattuun Vaqueron, Rodero-Merinon, Caceresin ja Lindnerin (2009) määritelmään (Korpinen, 2011). He kokosivat yhteen tieto- ja viestintätekniikan alan asiantuntijoiden määritelmiä pilvestä ja esittivät näiden pohjalta oman määritelmänsä pilvelle:

> ”Pilvet ovat suuri helposti käytettävä ja helppopääsyinen kokonaisuus virtualisoituja resursseja, kuten laitteistot, kehitysympäristöt ja palvelut. Nämä resurssit voivat dynaamisesti sopeutua muuttuneeseen kuormitukseen, mahdollistaen optimoidun resurssien käytön. Tästä resurssikokonaisuudesta maksetaan tavallisesti käytön mukaan. Käytön saatavuuden takaa pilvipalveluntarjoaja sovitun palvelutaso-sopimuksen mukaan.”(Vaquero ym., s. 51)

Määritelmä on varsin yleisluontoinen, ja kokoaa keskeisiä pilvilaskennan ominaisuuksia.

Pilvipalvelulla on viitattu sekä Internetin välityksellä palveluna tarjottuun ohjelmistoon että näiden palveluntarjoajien datakeskuksien laitteistoon ja järjestelmiin. Palveluna tarjottua ohjelmistoa kutsutaan verkkosovelluspalveluksi, ja siihen on jo pitkään viitattu termillä SaaS. Datakeskusten laitteistoa ja järjestelmiä kutsutaan pilveksi. Pilvipalvelujen tuottajat, kuten Amazon Web Services, Googlen AppEngine ja Microsoftin Azure, tarjoavat pilveä julkiseen käyttöön jatkuvan laskutuksen periaatteella. Tarjottua palvelua kutsutaan pilvilaskennaksi. (Armbrust ym.,2009)

Tässä tutkielmassa pilvisovelluksella ei viitata pelkästään verkkosovelluspalveluun vaan sellaiseen verkkosovelluspalveluun, joka käyttää pilvilaskentaa tietokoneresursseinaan. Toisin sanoen pilvisovellus on sovellus, joka sijaitsee pilvessä yksittäisen datakeskuksen sijasta.

2.2 Pilvilaskennan ominaisuuksia

Pilvilaskenta on uusi termi IT-alalla pitkään olleesta unelmasta tarjota tietokonelaskentaa yleishyödykkeen tapaan, mistä on nyt tullut todellista liiketoimintaa. Pilvilaskennalla on mahdollisuus muokata suurta osaa IT-alasta, koska se tekee ohjelmistojen toteuttamisesta verkkosovelluspalveluna entistä houkuttelevamman. Se vaikuttaa siihen, miten IT-laitteistoja suunnitellaan ja ostetaan. (Armbrust ym., 2009)

SaaS-verkkosovelluspalveluiden hyödyt niin loppukäyttäjälle kuin palveluntarjoajalle ymmärretään hyvin. Verkkosovelluspalveluntarjoajat pitävät keskitetystä ohjelmiston asennuksesta, ylläpidosta ja versionhallinnasta. Verkkosovelluspalvelun käyttäjät pääsevät käyttämään palvelua milloin tahansa ja mistä tahansa, sekä pystyvät helposti jakamaan dataansa ja varastoimaan sen turvallisesti palveluun. Pilvilaskenta mahdollistaa sen, ettei ohjelmistosuunnittelijoiden tarvitse enää sijoittaa suurta pääomaa palvelinlaitteistoon ja niiden ylläpitämiseen verkkosovelluspalveluidensa saataville saattamiseksi. Pilven resurssien skaalautuvuus tarpeen mukaan mahdollistaa myös sen, ettei pilvilaskennan asiakkaan tarvitse arvioida verkkosovelluspalvelunsa käyttöastetta etukäteen. Samaan tapaan kuin verkkosovelluspalveluiden käyttäjät voivat sysätä osan ongelmistaan verkkosovelluspalveluiden tarjoajille, voivat verkkosovelluspalvelun tarjoajat sysätä osan ongelmistaan pilvilaskennan tarjoajille, joita ovat esimerkiksi Amazon Web Services, Google AppEngine ja Microsoft Azure. (Armbrust ym., 2009)

Vaqueron ym. (2009) kokoamien ICT-alan ammattilaisten määritelmistä useimmin mainitut pilven ominaisuudet olivat skaalautuvuus, jatkuvan laskutuksen periaate ja virtualisointi. Samoja ominaisuuksia ilmenee myös Armbrustin ym. (2009 s. 4) raportissa, jossa listataan laitteiston kannalta kolme uutta näkökulmaa, jotka pilvilaskenta on tuonut:

1. Illuusio tietokoneresurssien ehtymättömyydestä tarvittaessa, mikä eliminoi pilvilaskennan asiakkaiden tarpeen varautua pitkälle etukäteen.

2. Pilven käyttäjien etukäteis-sitoutumisen poistuminen mikä mahdollistaa sen, että yritykset voivat aloittaa pienestä ja lisätä laitteistoresursseja tarpeen mukaan.

3. Mahdollisuus maksaa tietokoneresursseista käytön mukaan lyhyellä aikavälillä, ja mahdollisuus vapauttaa tarpeettomia resursseja, näin palkiten säästeliäisyydestä.

Armbrust ym. (2009) mukaan kaikki kolme ovat tärkeitä tekijöitä pilvilaskennan mahdollistamaan tekniseen ja taloudelliseen muutokseen. Aiemmissa epäonnistuneissa yrityksissä tehdä tietokonelaskennasta hyötypalvelua on joku näistä kolmesta ominaisuudesta puuttunut (Armbrust ym., 2009).

2.3 Tietokannan hallinta pilvessä

Pilvilaskennan tietokoneresurssien elastisuus ja jatkuvan laskutuksen periaate on luonut uusia haasteita ohjelmistoille. Toimiakseen nopeasti ja kannattavasti pilvisovellusten pitää pystyä skaalautumaan nopeasti käyttöasteen kasvaessa ja myös laskiessa (Armbrust ym., 2009). Tämä on uusi haaste myös tietokannan hallintajärjestelmille (Feuerlicht, 2010).

Perinteisesti organisaatioiden datakeskukset ovat koostuneet suhteellisen pienestä määrästä tehokkaita tietokanta-palvelimia (Feuerlicht ym., 2011). Relaationaalisille tietokannoille on voitu luoda skaalautuvuutta vertikaalisesti lisäämällä prosessoritehoa ja käyttämällä nopeampia muistilaitteita (Pokorný, 2011). Pilven infrastruktuuri koostuu taas sadoista tuhansista edullisista palvelimista ja muistilaitteista (Feuerlicht ym., 2011). Pilvessä on käytännöllisempää ja halvempaa skaalautua horisontaalisesti osittamalla tietokanta tuhansille dynaamisesti lisättäville koneille (Pokorný, 2011). Tietokannan hallintajärjestelmän pitää samalla pystyä hallinnoimaan tiedon saatavuutta ja ristiriidattomuutta (Feuerlicht ym., 2011).

Abadi (2009 s. 4) on maininnut kolme tiedonhallinnan kannalta relevanttia pilvilaskennan ominaisuutta:

- **Laskentateho on elastista, mutta ainoastaan jos työtehtävä voidaan tehdä rinnakkain**. Esimerkiksi kausittaisiin tai odottamattomiin verkkosovelluspalveluiden kysynnän piikkeihin voidaan allokoida lisää tietokoneresursseja minuuteissa. Tästä ei ole kuitenkaan mitään hyötyä, jos sovellus ei pysty jakamaan osaa työstään uudelle palvelininstanssille, joka toimii rinnakkain vanhojen palvelin instanssien kanssa. Tällaiseen ympäristöön

soveltuvat parhaiten Shared-nothing –arkkitehtuurin mukaiset sovellukset, joissa itsenäiset koneet suorittavat tehtävän resurssien minimaalisella limittäisyydellä.

- **Data on varastoitu epäluotettavalle isännälle.** Vaikka pilvilaskennan tarjoajan liiketoiminnan kannalta asiakkaan datan yksityisyyden loukkaaminen ei vaikuta järkevältä, tekee sen mahdollisuus osan potentiaalisista asiakkaista hermostuneiksi. Data on aina fyysisesti varastoituna johonkin maahan ja on alistettu kyseisen maan laille ja säädöksille. Esimerkiksi Yhdysvalloissa hallituksen on mahdollista vaatia pääsyä minkä tahansa tietokoneen dataan, jolloin pilvilaskennan asiakkaan dataa voidaan antaa eteenpäin sen tietämättä. Asiakas ei voi muuta kuin pelätä pahinta, että jos dataa ei ole salattu avaimella, joka ei sijaitse isännän koneella, dataan voidaan päästä käsiksi ilman asiakkaan tietämystä.

- **Dataa kopioidaan usein yli pitkien maantieteellisten välimatkojen.** Datan saatavuus ja pysyvyys on tärkeintä pilvivarastojen tarjoajille, koska datan häviäminen tai saavuttamattomuus on vahingollista niin palvelun käyttäjälle kuin tarjoajan liiketoiminnan maineelle. Datan saatavuus ja pysyvyys on tyypillisesti saavutettu kopioimalla tietoa automaattisesti ilman asiakkaan pyyntöä. Datakeskusten ollessa ympäri maailmaa on virheensietokykyä parannettu tiedon kopioimisella ja hajauttamalla yli pitkien välimatkojen. (Abadi, 2009)

Kaikki nämä ominaisuudet sopivat heikosti transaktioperusteiseen (transactional) tiedonhallintaan. Tällä termillä Abadi (2009) viittaa tietokantojen keskeisimpiin sovelluksiin, joita pankkien, lentoyhtiöiden varausjärjestelmien, verkkokauppojen ja toimitusketjujen tiedon hallinnassa käytetään. Nämä tietokannat luottavat ACID (atomicity, consistency, isolation, durability) – tapahtumanhallintaominaisuuksiin, eli tiedon atomisuuteen, ristiriidattomuuteen, eristyneisyyteen ja pysyvyyteen. Nämä tietokannat perustuvat tyypillisesti Shared-everything – arkkitehtuuriin, jossa skaalautuvuus on huonompaa verrattuna Shared-nothing – arkkitehtuurin sovelluksiin. Nämä tietokannat sisältävät usein myös kaiken operatiivisen datan liiketoiminta prosessien läpivientiin, ja ne voivat sisältää myös arkaluontoisia asiakastietoja, kuten luottokorttitietoja. Näiden tietojen säilyttämiseen ulkopuolisena palveluna voi liittyä riski. (Abadi, 2009)

ACID–ominaisuuksia on vaikea taata pilviympäristössä, jossa tietoa kopioidaan hajautetusti ympäri maailmaa (Abadi, 2009). Tilanteessa voidaan soveltaa Brewerin (2000) esittämää CAP-teoreema hajautetuille järjestelmille. CAP teoreema käsittelee kolmea ominaisuutta (Gilbert & Lynch, 2002 s 51):

- **Ristiriidattomuus** (Consistency) tarkoittaa, että tieto näkyy samana kaikille käyttäjille.
- **Saatavuus** (Availability) tarkoittaa, että data on aina saatavilla ja muokattavissa odotetussa vasteajassa.
- **Osittamistoleranssi** (Partition tolerance) tarkoittaa, että tietokantaan voidaan suorittaa kyselyitä ja päivityksiä, vaikka osa tietokannasta on saavuttamattomissa.

CAP-teoreeman mukaan hajautettu järjestelmä, joka käyttää jaettua dataa, voi ylläpitää täydellisesti korkeintaan kahta ominaisuutta (Brewer, 2000). Perinteiset tietokannan hallintajärjestelmät ovat yleensä suosineet ristiriidattomuutta (Pokorný, 2011). CAP-teoreeman pohjalta voidaan kuitenkin todeta, että pilven kaltaisessa hajautetussa järjestelmällä täydellistä ristiriidattomuutta voidaan ylläpitää ainoastaan tiedon saatavuuden tai osittamistoleranssin kustannuksella (Brewer, 2000). Pilven koostuessa sadoista tuhansista edullisista koneista on osittamistoleranssi pakollinen ominaisuus tietokannan hallintajärjestelmälle, joten CAP-teoreeman pohjalta joudutaan pilvimaailmassa tekemään valintoja ristiriidattomuuden ja saatavuuden väliltä. (Pritchett, 2008; Pokorný, 2011). Monet web-sovellukset sietävät tiedon väliaikaista ristiriitaisuutta, mutta kyselyiden kasvavat vasteajat tai tiedon saavuttamattomuus voi olla kallista (Feuerlicht ym., 2011; Abadi, 2009). Näille sovelluksille on kehitetty uusi tapahtumanhallintaominaisuuksien malli, jota kutsutaan BASE-ominaisuuksiksi (Pritchett, 2008). BASE-ominaisuuksia tukeva tietokanta on aina saatavilla (basically available), ei ole välttämättä yhtenevässä tilassa (soft state), mutta takaa että tietokanta saavuttaa lopulta ristiriidattoman tilan (eventual consistency) (Pritchett, 2008).

Pilvisovellukselle sopivalta tietokannan hallintajärjestelmältä vaaditaan siis elastista ja dynaamista skaalautuvuutta, tieturvan hallintaa ja osittamistoleranssia (Abadi, 2009; Pritchett, 2008; Pokorný, 2011). Tämän jälkeen joudutaan tekemään valintoja tietokannan ristiriidattomuuden ja saatavuuden väliltä (Pritchett, 2008).

3 NOSQL-TIETOKANNAN HALLINTAJÄRJESTELMÄT

Tässä luvussa esitellään lyhyesti, miten ja miksi NoSQL-tietokantakonsepti syntyi. Tämän jälkeen tutustutaan tärkeimpiin NoSQL-tietokantatuotteisiin. Tarkoituksena on esitellä NoSQL-tietokantojen toimintaperiaatteita ja niiden tarjoamia ominaisuuksia pilvimaailman sovelluksille. Luvussa otetaan myös kantaa siihen, miten nämä tietokantakonseptit vastaavat pilvimaailman haasteisiin.

3.1 NoSQL-tietokantakonsepti

Pilvimaailman tietokannanhallinnan haasteisiin on kehitetty useita kaupallisia ja avoimeen lähdekoodiin perustuvia NoSQL-tietokantatuotteita, joiden tarkoituksena on tarjota elastisuutta, automatisoitua tiedon saatavuuden hallintaa (automatic provisioning) ja virheensietokykyä. NoSQL -termin alle kuuluvat sellaiset tietokantatuotteet kuin Googlen Bigtable (Chang ym., 2008), Amazonin S3 (Brantner ym., 2008), Yahoon PNUTS (Cooper ym., 2008) ja Facebookin Cassandra (Lakshman ym., 2010).

Yhteinen tekijä kaikille NoSQL-tietokantatuotteille on se, etteivät ne perustu relaatioihin eivätkä käytä SQL-kieltä kyselykielenään (Feuerlicht & Pokorný, 2011). SQL-kieltä saatetaan kuitenkin käyttää osana tietokannan hallintajärjestelmää, joten NoSQL-termi on monesti tulkittu tarkoittavan, etteivät kyseisen termin alle kuuluvat tietokantasovellukset käytä pelkästään SQL-kieltä (not-only-SQL) (Pokorný, 2011).

Horisontaalisen skaalautuvuuden mahdollistaminen NoSQL-tuotteissa on johtanut relationaalisesta tietomallista luopumiseen, minkä tilalle tarjotaan yksinkertaisempia tietomalleja (Pokorný, 2011). Relationaaliset tietokannan hallintajärjestelmät ovat käytännössä aina tukeneet täydellisesti ACID-ominaisuuksia ja jatkuvaa ristiriidattomuutta (Pokorný, 2011). Monet web-sovellukset sietävät tiedon väliaikaista ristiriitaisuutta, mutta kyselyiden kasvavat vasteajat tai tiedon saavuttamattomuus voi olla kallista (Feuerlicht ym., 2011; Abadi, 2009).

Näille sovelluksille on kehitetty uusi tapahtumanhallintaominaisuuksien malli, jota kutsutaan BASE-ominaisuuksiksi (Pritchett, 2008). BASE-ominaisuuksia tukeva tietokanta on aina saatavilla (basically available), ei ole välttämättä yhtenevässä tilassa (soft state), mutta takaa että tietokanta saavuttaa lopulta ristiriidattoman tilan (eventual consistency) (Pritchett, 2008).

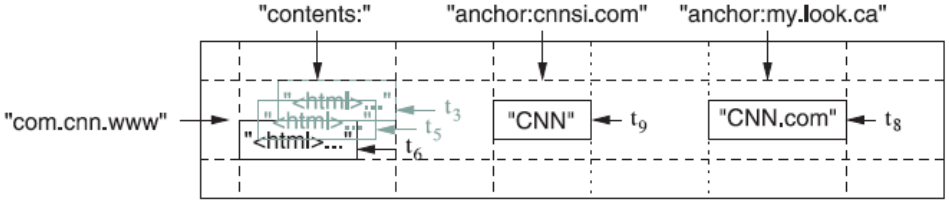
NoSQL on uusi tietokantakonsepti, joten lukuisista siihen kuuluvista tietokantatuotteista löytyy vähän kirjallisuutta. Esimerkiksi Amazon julkaisi uuden pilvisovelluksille suunnatun DynamoDB-tietokantasovelluksen kuluneen vuoden alussa (http://aws.amazon.com/dynamodb/, 2012). Kyseisestä tietokannasta on toistaiseksi kirjoitettu vain lyhyitä uutisia ja blogi-kirjoituksia, joten DynamoDB-tietokantasovelluksen sijasta käsitellään Amazonin toista merkittävää pilvisovelluksissa jo vuosia hyödynnettyä tietokantaratkaisua S3-tallennuspalvelua (Brantner ym., 2008). Tämän lisäksi tarkastellaan seuraavaksi Googlen Bigtable-tallennusjärjestelmää, jonka arkkitehtuuriin ja toimintaan perustuu myös esimerkiksi Facebookin Cassandra-tietokanta (Lakshman ym., 2010). Viimeisenä tarkastellaan Yahoo!:n PNUTS-tietokantapalvelua, joka tarjoaa mielenkiintoisen ohjelmistorajapinnan ristiriidattomuuden hallintaan (Cooper ym., 2008).

3.2 Bigtable

Bigtable on Googlen kehittämä hajautettu tallennusjärjestelmä, joka on suunniteltu skaalautumaan tuhansille koneille. Yli 60 Googlen tuotetta ja projektia, esimerkiksi Google Analytics ja Google Earth, käyttävät Bigtable-tietokantajärjestelmää. (Chang ym., 2008)

3.2.1 Tietomalli

Bigtable-tallennusjärjestelmässä tietokanta koostuu *Bigtable-tauluista* (Kuvio 1). Bigtable-taulu on harva (sparse), hajautettu, pysyvä, moniulotteinen ja lajiteltu kartta (map). Kartta on indeksoitu kolmiulotteisesti riviavaimen, sarakeavaimen ja aikaleiman mukaisesti. Yksittäistä alkiota kolmen ulottuvuuden risteyskohdassa kutsutaan *soluksi* (cell). Solujen arvot ovat tulkitsemattomia merkkijonoja. (Chang ym., 2008)



KUVIO 1 Bigtable-taulu (Chang ym., 2008 s. 3)

Bigtable-klusteri on joukko prosesseja, jotka ajavat Bigtable-ohjelmistoa. Kukin klusteri palvelee joukkoa Bigtable-tauluja.

Rivit on sanakirjamaisesti (lexicographic) järjestetty riviavaimen mukaan. Tämä mahdollistaa, että Bigtable-tietokannan käyttäjät voivat riviavaimen valinnalla järjestää toisiinsa liittyvät tiedot lähelle toisiaan, millä voidaan tehostaa rivijoukkoon kohdistuvia kyselyitä. Riviavain voi olla jopa 64 kilotavua pitkä. Kaikki yksittäiseen riviin kohdistuvat transaktiot ovat sarjallistuvia. Sen sijaan rivijoukon osalta sarjallistuvuus ei toteudu. Peräkkäisiä rivejä ryhmitellään *tableteiksi* (tablet), joita hallinnoimalla voidaan tasata järjestelmän kuormitusta. (Chang ym., 2008)

Sarakeavaimet ovat ryhmitelty *sarakeperheisiin* (columnfamily), jotka muodostavat tiedon saantiyksikön. Tiedon pakkaamisen helpottamiseksi samantyyppinen tieto tallennetaan yleensä samaan sarakeperheeseen. Datan käsittely tapahtuu sarakeperheittäin, mikä mahdollistaa, että eri sarakkeille voidaan antaa käyttäjästä riippuen erilaisia oikeuksia datan käsittelemiseen. (Chang ym., 2008)

Tiedoista voidaan tallentaa useita versioita, jotka tunnistetaan aikaleiman (vrt kuvion 1 t_3, t_5, t_6 -merkinnät) avulla. Versiot järjestetään päivityksen mukaan alkaen uusimmasta, jolloin uusin versio luetaan ensimmäisenä. Käyttäjä voi määrittää kuinka monta versioita säilytetään, tai kuinka vanhoja versioita säilytetään. (Chang ym., 2008)

3.2.2 Arkkitehtuuri

Bigtable on kehitetty hyödyntämällä useiden muiden Googlen infrastruktuuriin kuuluvia komponentteja. Bigtable käyttää lokin ja tiedon pysyvään varastointiin Google File System-tallennusjärjestelmää (GFS). GFS on hajautettu tallennusjärjestelmä, joka säilyttää tiedostoista useita kopioita paremman luotettavuuden ja saatavuuden ylläpitämiseksi. Googlen muuttumatonta SSTable-tiedostomuotoa käytetään tiedon sisäiseen tallentamiseen. Bigtable käyttää myös hajautettua Chubby-lukkopalvelua esimerkiksi tablettien sijaintitiedon ylläpitämiseen. Chubby-palveluun voi tallentaa hakemistoja ja pieniä tiedostoja, joita voidaan käyttää esimerkiksi lukkoina. (Chang ym., 2008)

Bigtable klusteri koostuu pääasiassa asiakkaan kirjastosta, isäntäpalvelimesta ja useista tablettipalvelimista. Kaikki komponentit luottavat toiminnassaan kommunikaation Chubby-palveluun. Tablettipalvelimia voidaan lisätä ja poistaa dynaamisesti kuormituksen muuttuessa. (Chang ym., 2008)

Isäntäpalvelin on vastuussa muun muassa tablettien määräämisestä tablettipalvelimille ja tablettipalvelimien kuormituksen tasapainottamisesta. Isäntäpalvelin valvoo tablettipalvelimien tilaa Chubby-palvelun välityksellä, ja mikäli tablettipalvelin menettää yhteyden Chubby-palveluun, isäntäpalvelin keskeyttää tablettipalvelimen toiminnan ja siirtää sen tabletit toimiville tablettipalvelimille. Isäntäpalvelin ei ole tiedon kulun välissä, ja kaaduttuaan se pystyy toipumaan ja päivittämään tilansa Chubby-palvelun avulla. Näin isäntäpalve-

limesta ei muodostu järjestelmän pullonkaulaa, eikä järjestelmän toiminta ole isäntäpalvelimen varassa. (Chang ym., 2008)

Tietoa tablettien sijainnista ylläpitää Chubby-lukkopalveluun tallennettu kolmitasoinen tiedostopolku. Juuritaulu pitää yllä sijaintia joukosta metatietotauluja, jotka pitävät yllä tablettien sijaintitietoa. Asiakaskone ohjaa kyselyt suoraan tablettipalvelimille paitsi tilanteissa, jolloin se ei löydä oikeaa tablettia. Tällöin asiakaskone hakee kyseisen tabletin ja sen ympärillä olevien tablettien sijainnin Chubby-palvelusta ja tallentaa ne kirjastoonsa. (Chang ym., 2008)

Tablettipalvelin hallinnoi tyypillisesti noin kymmenestä tuhanteen tablettia. Se hallinnoi luku- ja kirjoitusoperaatioita tabletteihin ja pilkkoo liian suureksi kasvaneita tabletteja. Käyttäjän luku- ja kirjoitusoperaatiot ohjataan suoraan tablettipalvelimelle, jotka ennen päivityksen suorittamista tarkistavat operaation oikeellisuuden lisäksi Chubby-palvelusta, että käyttäjällä on oikeudet kyseiseen päivitykseen. Operaatiot ohjataan tablettipalvelimen välimuistiin, joka täyttyessään muutetaan SStable-tiedostoksi, joka kirjoitetaan GFS-tallennusjärjestelmään. (Chang ym., 2008)

3.2.3 Pilvimaailman haasteet

Bigtable on osoittautunut suorituskykyiseksi ja skaalautuvaksi tietokannan hallintajärjestelmäksi. Klustereiden kapasiteettia voidaan helposti kasvattaa lisäämällä palvelimia, mikä on myös kokeellisesti osoitettu. Bigtable-tietokantasovellus on siis pilvisovelluksen tarpeisiin skaalautuva tietokannan hallintajärjestelmä. (Chang ym., 2008)

Bigtable tarjoaa myös ominaisuuksia pilvimaailmassa huolta aiheuttavaan tietoturvaan (Chang ym., 2008; Abadi, 2009). Bigtable pystyy tarjoamaan käyttäjille käyttöoikeuksia sarakepohjaisesti, joten arkaluontoiseen dataan voidaan päästää käsiksi vain oikeutetut käyttäjät (Chang ym., 2008). Tämä mahdollistaa, että osa taulusta voi olla yksittäiselle käyttäjälle kirjoituslukittuna ja osa myös lukulukittuna (Chang ym., 2008). Kaikissa luku- ja kirjoitusoperaatioissa tarkistetaan käyttäjän oikeudet datan käsittelemiseen (Chang ym., 2008). Kyselyiden tehostamiseksi tieto on pakattuna Bigtable-järjestelmässä algoritmilla, jossa esimerkiksi saralleessa vierekkäin oleva samankaltainen tieto voidaan pakata tehokkaasti (Chang ym., 2008). Tiedon tallentaminen salausalgoritmilla Bigtable-tietokantaan saattaa siis vaikuttaa pakkauksen tehokkuuteen, ja näin myös kyselyiden tehokkuuteen.

Bigtable on myös varautunut palvelimien kaatumisiin, mitä voidaan pitää pakollisena ominaisuutena pilvessä (Chang ym., 2008; Pritchett, 2008; Feuerlicht ym., 2011). Järjestelmän toiminnan kannalta tärkeät komponentit, isäntäpalvelin ja Chubby-palvelu pystyvät automaattisesti palautumaan palvelimen kaatumisesta. Chubby-palvelua ajetaan samanaikaisesti viisi, joista yksi toimii varsinaisena järjestelmän osana. Sen kaatuessa jonossa seuraava ottaa lennosta kaatuneen Chubby-palvelimen paikan. Isäntäpalvelimen kaatuessa tai sen huomatessa olevan järjestelmän saavuttamattomissa käynnistetään uusi isäntäpalvelin, joka päivittää tilansa Chubby-palvelun avulla. (Chang ym., 2008)

Bigtable tarjoaa sarjallistuvia ominaisuuksia klusterin sisällä tapahtuviin yhtä riviä koskeviin transaktioihin, mikä on osoittautunut riittäväksi ristiriidattomuudeksi monille Bigtable-tietokannan hallintajärjestelmää käyttäville sovellukselle. Maailmanlaajuisesti hyvän saatavuuden ylläpitäminen vaatii kuitenkin tietokannan kopioimista useammalle klusterille, mikä tekee ristiriidattomuuden ylläpitämisestä vaikeampaa. Tämä on mainittukin yhdeksi osa-alueeksi, joka vaatii kehittämistä Bigtable-tietokannan hallintajärjestelmässä. (Chang ym., 2008)

3.3 Amazon S3

Amazon S3 (Simple Storage System) on tiedon tallennuspalvelu, joka on yksi tuote Amazon Web Services –pilvipalvelutuoteryhmästä, johon kuuluu myös SQS –viestinvälityspalvelu (Simple Queueing System) ja EC2 pilvilaskentapalvelu (Elastic Computing Cloud). S3 ei ole tietokantatuote, mutta sen päälle voidaan rakentaa erilaisia tietokantasovelluksia. Esimerkiksi kuvapalvelusovellus Smugmug (www.smugmug.com) käyttää S3-tallennusjärjestelmän päälle rakennettua tietokantasovellusta. (Brantner ym., 2008)

3.3.1 Tietomalli

S3 tarjoaa skaalautuvuutta ja hyvää tiedon saatavuutta pienillä kustannuksilla. S3 voidaan ajatella olevan loputon tallennusmuisti, johon voi tallentaa erilaisia objekteja yhden tavun ja viiden gigatavun väliltä, ja jolla voi olla loputon määrä käyttäjiä. Objektit tallennetaan tunnisteen (URI) kanssa, minkä avulla ne voidaan eritellä. Objektit tallennetaan *astioihin* (bucket), jonka käyttäjä määrittää objektia tallentaessa. S3 tarjoaa erilaisia tapoja objektien hakemiseen astioista. Käyttäjä voi hakea tietyn astian kaikki objektit tai hakea tunnisteen avulla pelkästään tietyt objektit. Käyttäjä voi myös myöntää tietylle astialle luku- ja kirjoitusoikeuksia muille käyttäjille. S3-tallennuspalvelua käytetään yleisesti varmuuskopiointiin ja isojen multimediatiedostojen tallentamiseen. S3-tallennusjärjestelmän rajapinta muistuttaakin pitkälti levymuistin rajapintaa, mikä on toisaalta tehnyt siitä myös houkuttelevan alustan tietokantasovelluksen toteuttamiseen. (Brantner ym., 2008)

3.3.2 Arkkitehtuuri

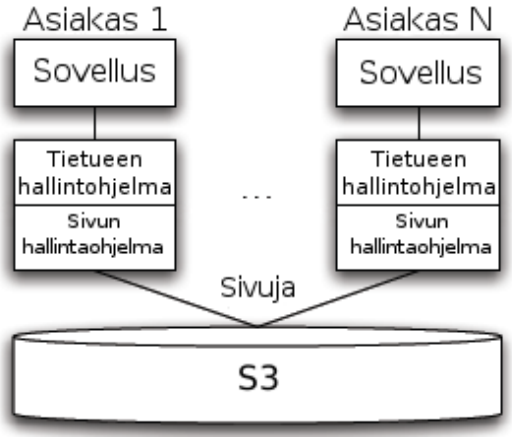
Amazon SQS (Simple Queueing System) on myös osa AWS-pilvipalvelutuoteryhmää. SQS:n avulla voidaan muodostaa lähes loputon määrä jono-tietorakenteita, jotka voivat olla lähes loputtoman pituisia. Jonot omaavat tunnisteet, jonka avulla ne voidaan eritellä. Näiden jonojen avulla voidaan välittää viestejä esimerkiksi S3-tallennusjärjestelmälle, mikä mahdollistaa eri-

laisten tietokannan hallintajärjestelmien kehittämisen S3-tallennusjärjestelmän alustalle. (Brantner ym., 2008)

S3 on suunniteltu tallennuspalveluksi, josta tietoa voi hakea kuka tahansa ja milloin tahansa, minkä takia tiedon ristiriidattomuudesta on jouduttu tinkimään. S3 takaa pelkästään, että päivitykset tulevat lopulta näkyville kaikille käyttäjille. Toisin sanoen voi kestää määrittämättömän ajan, ennen kuin tieto on päivittynyt kaikille käyttäjille, ja päivitysten lisäys ei välttämättä tapahdu samassa järjestyksessä. Mikäli halutaan saavuttaa parempia takeita tiedon eheydestä, joudutaan ne toteuttamaan osaksi tietokantasovellusta S3-tallennuspalvelun päälle. (Brantner ym., 2008)

Siirtonopeus S3-tallennuspalvelusta on huomattavasti hitaampi kuin paikalliselta levymuistista. S3-tallennuspalvelun etuna on kuitenkin se, että siirtonopeus pysyy suhteellisen samana kaikissa tilanteissa. Siirtonopeus pysyy samana käyttäjien määrästä riippumatta ja käyttäjän ei tarvitse huolehtia laitteistovirheistä, joista S3 toipuu automaattisesti. Amazon ei ole julkaissut tarkasti S3-tallennusjärjestelmänn toimintaperiaatteita, mutta oletettavasti se tallentaa (replikoi) tiedot useammalle palvelimille eri paikoissa. S3-tallennusjärjestelmän siirtonopeus kasvaa haettavan objektin koon kasvaessa, joten pienet tietueet kannattaa pakata *sivuiksi* (page), jolloin yksittäistä tietuetta haettaessa haetaan kokonaan se sivu, johon tietue kuuluu. (Brantner ym., 2008)

S3-tallennuspalvelun päälle rakennetun tietokantasovelluksen arkkitehtuuriksi on ehdotettu kuviosta 2 ilmenevää rakennetta.



KUVIO 2 S3-tietokannan hallintajärjestelmän arkkitehtuuri (Brantner ym., 2008 s. 254)

Tietueen hallintaohjelma (record manager) hallitsee tietueita, jotka kuuluvat johonkin ryhmään. Tietue sisältää varsinaisen tietonsa lisäksi avaimen, jonka mukaan ne on järjestetty binääripuuksi ryhmien sisällä. Tietueet on tallennettu sivuihin, jotka on tallennettu S3-tallennuspalveluun. *Sivun hallintaohjelma* (page manager) hallinnoi sivujen tallentamista ja hakemista S3-tallennuspalvelustä. Se ylläpitää myös välimuistia sivuista, joita se väliajoin päivittää S3-tallennuspalvelustä. Arkkitehtuuri mahdollistaa, että S3-tallennuspalvelun päälle rakennettu tietokannan hallintajärjestelmä voi sijaita EC2-pilvilaskentapalvelussa, erillisissä datakeskuksissa tai yksittäisillä tietokoneilla

tai mobiililaitteilla, mikä tekee siitä varsin houkuttelevan vaihtoehdon web-sovellusten kehittäjille. (Brantner ym., 2008)

Käyttämällä SQS-palvelua ja erilaisia algoritmeja transaktioiden välittämiseen voidaan rakentaa tietokannan hallintajärjestelmiä, jotka omaavat erilaisia takeita tiedon ristiriidattomuudesta. Käytännössä S3-tallennuspalvelun päällä voitaisiin tukea täydellistä transaktioiden sarjallistuvaa ajoitusta, mutta se vaatisi staattisen globaalin laskurin käyttöä. Tämä globaali laskuri olisi kuitenkin potentiaalinen pullonkaula järjestelmässä, ja tämän yksittäisen komponentin häiriö saattaisi järjestelmälle kohtalokasta. S3-tallennuspalvelun elastisuuden hyödyntäminen tietokantasovelluksessa vaatii eheyden takeiden höllentämistä. (Brantner ym., 2008)

Yksinkertaistettuna käyttäjän tekemät päivitykset koostuvat kahdesta askeleesta. Ensimmäisenä käyttäjä lisää SQS-jonoon kaikki transaktioon liittyvät päivitykset. Jokaisella sivulla ja binääripuulla on oma SQS-jononsa. Toinen askel on tarkistuspiste, jossa tietyn jonon päivitykset lisätään S3-tallennuspalveluun. Päivityksiä koskeva sivu haetaan S3-tallennuspalvelusta välimuistiin, ja siihen lisätään päivitykset, minkä jälkeen sivu tallennettaan tallennusjäsjestelmään. Jotta voitaisiin välttää, että kaksi käyttäjää suorittaisi tarkistuspisteen samanaikaisesti, on kullekin sivulle luotu lukkopoletti SQS-jono muodossa. Aina kun tarkistuspiste suoritetaan, varataan kyseinen poletti, jotta kukaan muu ei samanaikaisesti voi suorittaa tarkistuspistettä. Tarkistuspisteiden ajoitukselle voi olla erilaisia strategioita, missä nyrkkisääntönä voidaan sanoa, että lyhyempi tarkistuspisteiden aikaväli lisää tiedon tuoreutta, mutta lisää ylläpitokuluja, kun kommunikaation määrä käyttäjän ja S3-tallennuspalvelun välillä kasvaa. (Brantner ym., 2008)

3.3.3 Pilvimaailman haasteet

S3 on skaalautuva tallennuspalvelu, jonka päälle voidaan rakentaa skaalautuva tietokantasovellus. S3-tallennuspalvelun skaalautuvuuden sisällyttäminen tietokantasovellukseen johtaa kuitenkin siihen, ettei vahvaa ristiriidattomuutta voida toteuttaa. Ristiriidattomuutta voidaan lisätä, mutta se johtaa tiedon heikompaan saatavuuteen ja ylläpitokustannusten kasvuun. S3 takaa vain, että päivitys näkyy lopulta kaikilla käyttäjillä. Käyttämällä edellä ehdotettua arkkitehtuuria ja algoritmeja S3-tallennuspalveluun pohjautuvassa tietokantasovelluksessa voidaan vaikuttaa siihen, kuinka kauan päivityksen näkyminen kaikille tietokantasovelluksen käyttäjille kestää. Tuoreemman tiedon ylläpitäminen maksaa kuitenkin tiedonsiirtokustannusten kasvaessa. Riittävänä tarkistuspisteiden intervallina web-sovelluksille voidaan pitää noin 10–15 sekuntia, jolloin ylläpitokustannukset ovat siedettäviä S3-tietokantasovelluksille. On siis selvää, etteivät S3-tallennuspalvelun päälle kehitetyt tietokannat sovi sovelluksille, missä tietoa päivitetään intensiivisesti. (Brantner ym., 2008)

Tietoturva on tyypillinen haaste pilvisovelluksille (Abadi, 2009). Tieturvaa voidaan toteuttaa S3-tietokantasovelluksissa eri tavoilla. Astian omistaja voi myöntää muistipaikkansa luku- ja kirjoitusoikeuksia eri käyttäjille. Sivut voi-

daan myös tallentaa salattuna S3-tallennuspalveluun ja myöntää purkuavaimia käyttäjille, joilla on oikeus lukea tietoa. (Brantner ym., 2008)

Ehdotetusta arkkitehtuurista tekee web-sovellusten kehittäjille houkuttelevaksi se, että tietokannan hallintajärjestelmä voidaan asentaa mille tahansa laitteelle, joka on yhteydessä S3-tallennuspalveluun (Brantner ym., 2008). Esitetyssä arkkitehtuurissa ei kuitenkaan oteta kantaa, miten sivun- ja tietueen hallintaohjelmat varautuvat kaatumiseen, mikä on oleellinen huoli, mikäli tietokannan hallintajärjestelmää ajetaan esimerkiksi EC2-pilvilaskentapalvelussa, jolloin tietokannan hallintajärjestelmän pitää olla varautunut laitteistovirheisiin (Feuerlicht ym., 2011).

3.4 PNUTS

PNUTS on Yahoo!:n kehittämä tietokantatuote, jota se käyttää omissa sovelluksissaan ja tarjoaa tietokantapalveluna web-sovellusten kehittäjille. Keskeisimmät vaatimukset web-sovellukselle on skaalautuvuus, jatkuvasti hyvä vasteaika maantieteellisesti hajautuneille käyttäjille, hyvä saatavuus ja virheensietokyky, sekä höllennetyt takeet ristiriidattomuudesta. (Cooper ym., 2008)

Web-sovellusten tiedon hallintaan liittyvien tarpeiden pohjalta Yahoo! teki useamman perustavanlaatuisen päätöksen suunnitelleessaan PNUTS-tietokantatuotetta (Cooper ym., 2008 s. 1278):

- Arkkitehtuurin pitää perustua tietue-tason reaaliajattomaan replikaatioon yli maantieteellisten välimatkojen ja taattuun viestinvälitykseen pysyvän lokin ylläpitämisen sijasta.
- Ristiriidattomuuden hallinta pitää mahdollistaa transaktionaalisia ominaisuuksia, mutta ei kuitenkaan täydellistä sarjallistuvaa ajoitusta.
- Pitää valita tarkasti, mitä ominaisuuksia sisällyttää (esim. hajautustaulut, järjestetyt taulut, joustava tietokannan kaava) ja mitä jättää pois (esim. rajoituksia ad hoc-kyselyihin, viite-eheys, sarjallistuva ajoitus).
- Tietokannan hallinta tarjotaan palveluna.

3.4.1 Tietomalli ja rajapinta

PNUTS-tietokantapalvelu tarjoaa sen käyttäjälleen yksinkertaistetun relationaalisen tietomallin. Tieto on järjestetty tauluihin, jossa tietueet voivat sisältää perinteisten attribuuttien lisäksi mielivaltaisia tietotyyppejä. Taulujen kaavat ovat joustavia, joten uusia attribuutteja voidaan lisätä pysäyttämättä kyselyitä tai päivityksiä, ja tietueiden ei tarvitse sisältää jokaisen attribuutin arvoja. (Cooper ym., 2008)

PNUTS-tietokannan kyselykieli mahdollistaa valinnan ja projektion vain yhdestä taulusta, mitä voidaan pitää rajoitteena verrattuna relationaalisiin tietokannan hallintajärjestelmiin. PNUTS sisältää kuitenkin hajoitus-kokoamis –kone (scatter-gather engine) nimisen komponentin, joka hallinnoi useampaa taulua koskevia kyselyitä. Yahoo!:n kokemusten pohjalta web-sovellusten kyselyt koostuvat yleensä yhteen tai pieneen määrään tietueita kohdistuvista lukuja kirjoitusoperaatioista. Näin ollen järjestelmä on optimoitu sen mukaan, että useimmat kyselyt kohdistuvat vain muutamiin kymmeniin tai satoihin tietueisiin. Hajoitus-kokoamis –kone hajottaa useampaa taulua koskevat kyselyt useammaksi yksittäistä taulua koskevaksi kyselyksi ja tarkkailee näiden onnistumista. PNUTS ei myöskään tue viite-eheyttä tai monimutkaisia ad hoc-kyselyitä, kuten liitos- ja järjestysfunktioita. Näiden omaisuuksien ja toimintojen toteuttamista voidaan pitää tulevaisuuden haasteena. (Cooper ym., 2008)

PNUTS-tietokanta tarjoaa tietuekohtaisen *aikajanaristiriidattomuuden* (timeline consistency). Tämä malli pohjautuu ajatukseen, jonka mukaan web-sovellukset käsittelevät yhtä tietuetta kerralla. Tietuen päivitykset päivittyvät kaikkiin kopioihinsa samassa järjestyksessä. Jokaiselle tietueella on määrätty isäntätietue, ja kaikki kopioihin kohdistuvat päivitykset johdetaan isäntätietueelle. Isäntätietue vaihtelee sen mukaan, mihin kopioon kohdistuu eniten päivityksiä. Isäntätietue säilyttää versionumeroa ja jokainen päivitys kasvattaa sitä. Tietue päivittyy kopioihinsa versionumeroiden mukaisessa järjestyksessä. (Cooper ym., 2008)

Seuraavilla PNUTS-tietokannan rajapintakutsuilla voidaan hallinnoida ristiriidattomuutta eri transaktioille (Cooper ym., 2008 s. 1279):

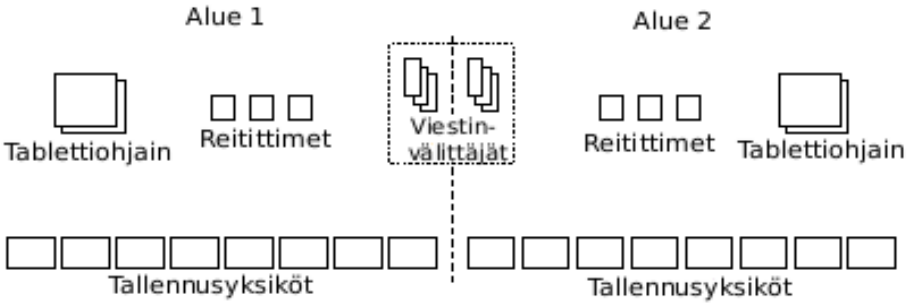
- **Read-any:** Palauttaa mahdollisesti vanhentuneen tietueen. Takaa ainoastaan, että palautettava tietue löytyy tietueen historiasta. Kutsun vasteaika on matala.
- **Read-critical(required_version):** Palauttaa halutun (required_version) tai uudemman version halutusta tietueesta.
- **Read-latest:** Palauttaa tietueesta uusimman version, joka on onnistuneesti kopioitu kaikkiin kopioihin.
- **Write:** Kirjoittaa tietueeseen.
- **Test-and-set-write(required_version):** Kirjoittaa tietueeseen ainoastaan jos tietueen versio vastaa haluttua versiota (required_version).

Näillä rajapintakutsuilla sovellus voi saavuttaa paremman suorituskyvyn tilanteissa, joissa ei tarvita tiukkaa ristiriidattomuutta. Myös yhtä riviä koskevat transaktiot voidaan suorittaa ilman lukkoja, mikä on hajautetuissa järjestelmissä erittäin toivottu ominaisuus. PNUTS ei kuitenkaan anna mitään takeita ristiriidattomuudesta transaktioille, jotka koskevat useampaa tietuetta. Sarjallistuvaa ajoitusta voidaan tarjota vain tietuekohtaisesti. (Cooper ym., 2008)

3.4.2 Arkkitehtuuri

PNUTS-tietokannan hallintajärjestelmän on jaettu *alueisiin* (region). Jokaiselle alueelle on kopioitu kaikki tietokannan taulut. Alueet on tavallisesti, mutta ei välttämättä, jaettu maantieteellisesti. Alueen komponentteja ovat *reitittimet* (router), *tablettiohjaimet* (tablet controller), *tallennusyksiköt* (storage unit) ja *viestinvälittäjät* (message broker). (Cooper ym., 2008)

Kuviosta 1 ilmenee järjestelmän rakenne. Data on horisontaalisesti jaettu tauluryhmiin, joita kutsutaan *tableteiksi* (tablet). Yksittäinen tabletti on tallennettu yhdelle palvelimelle alueen sisällä ja palvelin voi sisältää tuhansia tabletteja. Tabletteja voidaan joustavasti jakaa ja siirtää ylikuormitetuilta palvelimilta alikuormitetuille, ja palvelimen kaatuessa voidaan tabletit palauttaa hajautetusti muille käynnissä oleville palvelimille tai kokonaan uudelle palvelimelle. (Cooper ym., 2008)



KUVIO 3 PNUTS-tietokannan hallintajärjestelmän arkkitehtuuri (Cooper ym., 2008 s. 1280)

Tallennusyksiköt, reitittimet ja tablettiohjaimet ovat yhdessä vastuussa tablettien hallinnasta ja datan käsittelemiseen pääsystä. Tallennusyksiköt säilyttävät tabletteja joko järjestettynä tai hajautustauluina. Tietyn tietueen käsitteleminen vaatii tiedon siitä, missä tabletissa tietue sijaitsee ja missä tallennusyksikössä tabletti sijaitsee. Tablettiohjaimet ylläpitävät näitä tietoja samalla kun hallitsevat tablettien jakamista, siirtämistä palvelinkuormituksen tasaamiseksi ja toipumista palvelinvirheistä. Tablettiohjaimet eivät kuitenkaan ohjaa kyselyitä oikeisiin tietueisiin, vaan tästä pitävät huolen reitittimet. Reitittimet säilyttävät kopiota tietueiden sijainnista ja päivittävät sitä tietyn väliajoin tablettiohjaimelta. Reitittimet voivat sisältää väliaikaisesti vanhentuneen kartan, kun tablettiohjain siirtää tai jakaa tabletteja, jolloin väärin ohjattu kysely aiheuttaa virheilmoituksen. Tällöin reititin hakee tablettiohjaimelta uuden kopion kartasta. Kyseinen arkkitehtuuri mahdollistaa, etteivät tablettiohjaimet ole pullonkaulana tiedonkulussa ja kaatuneiden reitittimien tilalle voidaan käynnistää kokonaan uusia kaatuneiden reitittimien palauttamisen sijaan. (Cooper ym., 2008)

Jatkuvan lokin ylläpitämisen sijaan PNUTS käyttää Yahoo! Message Broker (YMB) –viestinvälittäjäkomponenttia ristiriidattomuuden hallintaan. Tämä on mahdollista, koska YMB pitää huolen, ettei viestejä kadoteta ennen kun ne

päivitetään tietokantaan, ja YMB on suunniteltu maantieteellisesti laajasti hajautettuun replikaatioon. YMB tallentaa viestit usealle palvelimelle ja poistaa ne vasta kun viesti on välittynyt kaikille kopioille. Näin se takaa, että julkaistut päivitykset saavuttavat kaikki kopiot yksittäisen viestivälittäjän kaatumisesta huolimatta. YMB-komponentit sijaitsevat PNUTS-alueista maantieteellisesti erillisissä datakeskuksissa, joista ne välittävät viestejä keskenään. Näin viestit voidaan hajauttaa keskitetysti, eikä päivityksiä tarvitse välittää alueesta toiselle. (Cooper ym., 2008)

YMB perustuu julkaise/tilaa–malliin, missä datan päivitykset julkaistaan YMB-järjestelmässä, kun ne saavuttavat vahvistetun (committed) tilansa. Mikäli päivitys ei kohdistu isäntätietueeseen, johdetaan päivitys isäntätietueelle. Isäntätietueen päivittyessä päivitys julkaistaan YMB-järjestelmässä, jossa viesti välitetään kaikille tietueen kopioille. Näin mahdollistetaan tietuekohtainen aikajanaristiriidattomuus, jossa tietueen päivitykset päivittyvät kaikkiin kopioihin päivitys-järjestyksessä. Viiteavainten rajoitteiden ylläpitämiseksi täytyy samalla viiteavaimella tulevat lisäykset ohjata samalle tallennusyksikölle, joten yksi tabletti joudutaan nimeämään isäntätabletiksi, johon kaikki lisäykset ohjataan. Tietue-tason ja tabletti-tason isäntiä hallinnoidaan erikseen, joten isäntätietueen ei tarvitse sijaita isäntätabletissa. (Cooper, Ramakrishnan, Srivastava, Silberstein ym., 2008)

PNUTS on palveluna tarjottu tietokantatuote. Palvelua käyttää samanaikaisesti useampi sovellus, jotka jakavat samoja resursseja. Käyttöasteen kasvaessa lisätään järjestelmään uusia palvelimia suorituskyvyn ylläpitämiseksi. Joillekin sovelluksille resurssien pullonkaula on levyhaut ja toisille välimuisti tai prosessorin laskentateho. Kaikissa tapauksissa palvelimien lisäys lisää pullonkaularesursseja, joita PNUTS allokoi automaattisesti käyttöönsä. (Cooper ym., 2008)

3.4.3 Pilvimaailman haasteet

PNUTS-tietokantatuotetta voidaan pitää pilvipalveluna tarjottuna tietokantana, joka käyttää pilvilaskentaa tietokoneresursseinaan. Se on suunniteltu automaattisesti skaalautuvaksi tietokannaksi maailmanlaajuisesti hajautetussa ympäristössä (Cooper ym., 2008). Kokeet ovat osoittaneet, että PNUTS-tietokannan käyttöasteen kasvaessa yli kaksinkertaisesti vasteaika kasvaa vain muutamia kymmeniä prosentteja, ja että uusien palvelimien lisäys pienentää vasteaikaa (Cooper, ym., 2008). Voidaan siis todeta, että PNUTS todella on skaalautuva tietokantasovellus, mikä on toivottu ominaisuus pilvisovellusten tietokannan hallintajärjestelmille (Abadi, 2009).

Yahoo!:n tavoitteena on ylläpitää yli kymmentä maailmanlaajuisesti hajautettua PNUTS-aluetta, joissa kussakin on palvelimia tuhannesta ylöspäin (Cooper ym., 2008). Ottaen huomioon, että pilvi koostuu edullisista laitteistokomponenteista, ja että tietokanta on hajautettu näin voimakkaasti ympäri maailmaa, täytyy järjestelmän olla varautunut erilaisiin laitteistovirheisiin (Feuerlicht ym., 2011, Cooper, ym., 2008) PNUTS tukee automaattista virheensietoky-

kyä niin varastoyksiköiden tasolla, kun viestinvälityksen tasolla (Cooper ym., 2008). Kaatuneiden palvelimien tilalle käynnistetään lennosta uusia palvelimia, joille kopioidaan kaatuneiden palvelimien tieto (Cooper ym., 2008).

Tietokannan ollessa hajautettuna ympäri maailmaa, on ristiriidattomuuden ylläpitäminen vaikeaa (Abadi, 2009). Tämä on otettu huomioon jo PNUTS-tietokantasovellusta suunnitellessa (Cooper, ym., 2008). PNUTS ei tue transaktioiden sarjallistuvaa ajoitusta, mutta se tarjoaa jonkin verran transaktionaalisia ominaisuuksia (Cooper, ym., 2008). BASE-ominaisuuksien lisäksi tuetaan tietuetason aikajanaristiriidattomuutta, mikä mahdollistaa, että sovellukset voivat tukea tietuekohtaisesti sarjallistuvaa ajoitusta silloin kun se on tarpeellista (Cooper, ym., 2008). Tätä voidaan pitää riittävänä ristiriidattomuutena esimerkiksi yhteisöpalveluille tai meta-datan ja liitetiedostojen tietokannoille (Cooper, ym., 2008).

Rajoitettujen eheyden takeiden lisäksi PNUTS ei tue myöskään liitos- ja järjestysfunktioita (Cooper, ym., 2008). Useampaa taulua koskevat kyselyt hajautetaan yksittäisiksi kyselyiksi, jolloin kyselyn tulos voi sisältää osittain erikäistä tietoa (Cooper, ym., 2008). Näistä perinteisiin relationaalisiin tietokantoihin liitetyistä ominaisuuksista on jouduttu luopumaan, jotta on voitu tarjota skaalautuvuutta, nopeaa vasteaikaa ja virheensietokykyä (Cooper, ym., 2008

4 HYBRIDI-TIETOKANTAKONSEPTI: ELASTRAS

Mahdollistaakseen pilvisovellukselle vaadittavaa skaalautuvuutta ja elastisuutta NoSQL-tietokantasovellukset on yleensä suunniteltu tukemaan ACID-ominaisuuksien sijasta BASE-tapahtumanhallintaominaisuuksia (Pokorný, 2011). BASE-ominaisuudet soveltuvat hyvin kuluttajille suunnatuille web-sovelluksille, mutta pankki- ja finanssialan sovellusten lisäksi myös yritysten liiketoiminnassa käytettäville sovelluksille ACID-ominaisuudet ovat pakollisia (Pritchett, 2008, Pokorný, 2011). Selvästikkin on tarvetta tietokannan hallintajärjestelmälle, joka yhdistäisi relationaalisten tietokannan hallintajärjestelmien monipuolisia toimintoja ja ACID-ominaisuuksia, sekä ei-relationaalisten tietokannan hallintajärjestelmien automatisoitua skaalautuvuutta, elastisuutta ja virheensietokykyä. (Feuerlicht & Pokorný, 2011, Das, Agarwal, Agrawal, Abbadi, 2010). Seuraavaksi esitellään ElasTraS – tietokantakonsepti, sen toimintaperiaate ja ominaisuudet. Lopussa otetaan myös kantaa siihen, miten ElasTraS vastaa sille asetettuihin haasteisiin pilvimaailmassa.

ElasTras on tietokantakonsepti, joka pyrkii tarjoamaan skaalautuvaa ja elastista relationaalista tietomallia pilviympäristössä. ElasTraS pyrkii tukemaan niin suuria ositettuja tietokantoja, kun paljon pieniä itsenäisiä tietokantojakin. (Das ym., 2010)

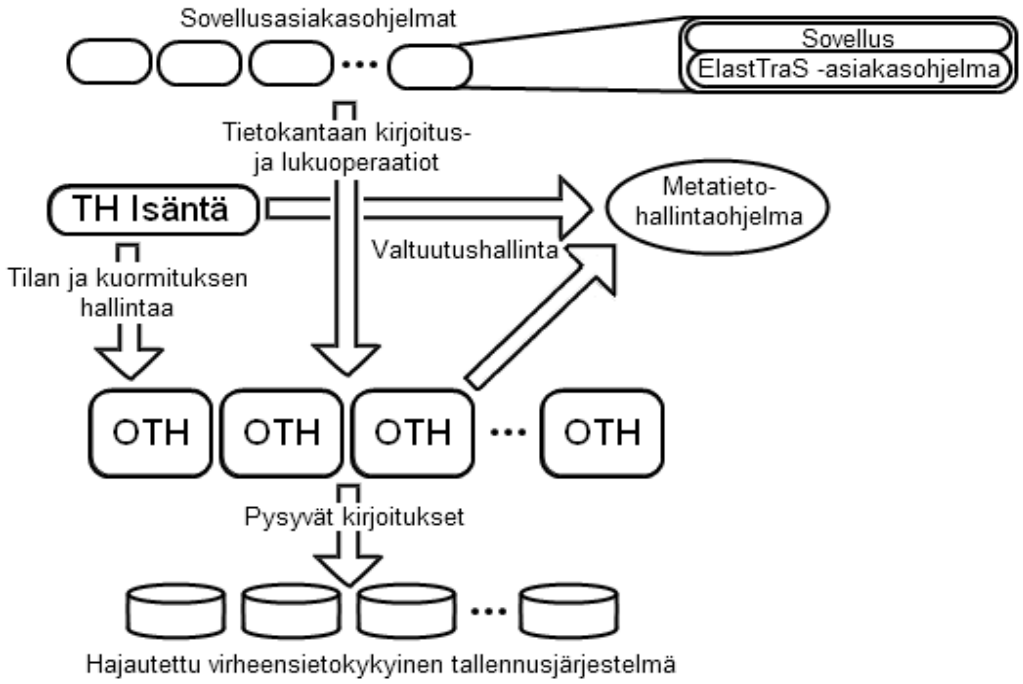
4.1 Tietomalli

ElasTraS-tietokannan hallintajärjestelmä perustuu perinteiseen relationaalisen tietomalliin. Tietokanta on jaettu osioihin niin, että jokainen osio on joko osa suurempaa kokonaista tietokantaa, tai yksi itsenäinen pienehkö tietokanta. Osittaminen on yleinen käytäntö suurten tietokantojen skaalaamisessa, mutta yleensä osittamista tehdään taulujen tasolla, jolloin yksittäinen taulu ositetaan. ElasTraS osittaa tietokantaa skeeman tasolla, jolloin toisiinsa liittyvät taulut liitetään samaan osioihin ja liittymätön tieto muihin osioihin. Skeeman tasolla osittamisen periaate pohjautuu ajatukseen, että hyvin useassa tapauksessa

transaktiot kohdistuvat vain muutamaan toisiinsa viittaaviin riveihin. Hyvän skaalautuvuuden ylläpitämiseksi sarjallistuvia transaktioita tuetaan osioiden sisällä ja osiota hallinnoimalla voidaan tasata järjestelmän kuormitusta. ElasTraS tukee dynaamista osittamista, jolloin osioita voidaan jakaa osion kasvaessa liian suureksi tai liittää yhteen niiden pienentyessä. (Das ym., 2010)

4.2 Arkkitehtuuri

ElasTraS-tietokannan hallintäjärjestelmän rakenne koostuu *hajaujutetusta virheensietokykyisestä tallennusjärjestelmästä* (distributed fault-tolerant storage), useista *omistustransaktiohallintaohjelmista* (owning transaction manager), yhdestä *isäntätransaktiohallintaohjelmasta* (transactional manager master), *metatietohallintaohjelmasta* ja useista *ElasTraS-asiakasohjelmista* (ElasTraS-client) (Kuvio 4). (Das ym., 2010)



KUVIO 4 ElasTraS-tietokannan hallintajärjestelmän arkkitehtuuri (Das ym., 2010 s. 3)

ElasTraS-tietokannan hallintajärjestelmän arkkitehtuuri ja toiminta muistuttaa paljon Googlen Bigtable-tietokantasovellusta. Kuten Bigtable-tietokannan hallintajärjestelmä luottaa GFS-tallennusjärjestelmän toimintaan, myös ElasTraS-tietokannan hallintajärjestelmän käyttää pysyvään tallennukseen hajautettua virheensietokykyistä tallennusjärjestelmää (HVT). Omistustransaktionhallintaohjelmat (OTH) hallinnoivat yhdestä useampaan tietokannan osiota samaan tapaan kuin Bigtable-tablettipalvelin hallinnoi tabletteja. Isäntätransaktiohallintaohjelma (TH Isäntä) valvoo omistustransaktiohallintaohjelmien tilaa ja tasaa

kuormitusta niiden välillä samaan tapaan kuin Bigtable-isäntäpalvelin hallinnoi Bigtable-tablettiohjaimia. Tietoa järjestelmän tilasta pitää yllä metatietohallintaohjelma, jonka toiminta vastaa Chubby-palvelua Bigtable-tietokannan hallintajärjestelmässä. (Das ym., 2010; Chang ym., 2008)

Hajautettu tallennusjärjestelmä pitää huolen, että tietoa voidaan pysyvästi kirjoittaa kaikissa tilanteissa ja pitää yllä tiedon ristiriidattomuutta. Omistustransaktiohallintaohjelmat pitävät huolen, ettei samaa tietuetta kirjoiteta samanaikaisesti hajautettuun tallennusjärjestelmään. Omistustransaktiohallintaohjelmat omistavat kymmenestä satoihin tietokannan osioita, joita ne hallinnoivat erikseen. ElasTraS-asiakasohjelmat suorittavat luku- ja kirjoitusoperaatioita suoraan omistustransaktiohallintaohjelmiin, jotka tarjoavat transaktioille ACID-ominaisuuksia yksittäisen omistustransaktiohallintaohjelman sisällä. Yksittäisen omistustransaktiohallintaohjelman sisältämiä osioita voidaankin pitää yksittäisenä relaationaalisena tietokantana. (Das ym., 2010)

Isäntätransaktiohallintaohjelma valvoo omistustransaktiohallintaohjelmien tilaa ja kuormitusta. Isäntätransaktiohallintaohjelma on vastuussa osioiden määräämisestä omistustransaktiohallintaohjelmille ja tarvittaessa jakaa niitä uudelleen kuormituksen tasaamiseksi. Isäntätransaktiohallintaohjelma valvoo myös, että kaikki omistustransaktiohallintaohjelmat ovat toimintakykyisiä ja tarvittaessa suorittavat palautumisen omistustransaktiohallintaohjelman kaatuessa. Kaatuneen omistustransaktiohallintaohjelman osioiden omistus siirtyy isäntätransaktiohallintaohjelmalle, joka myöntää osioiden omistuksen uudelle omistustransaktiohallintaohjelmalle, joka hakee kaatuneen omistustransaktiohallintaohjelman lokin tallennusjärjestelmästä ja päivittää osioiden tilan, minkä jälkeen osiot ovat taas käytettävissä. Virheensietokyvyn parantamiseksi on samanaikaisesti käynnissä ja synkronoituna useampia isäntätransaktiohallintaohjelmia, jolloin isäntätransaktiohallintaohjelman kaatuessa voidaan ottaa lennosta uusi käyttöön. (Das ym., 2010)

Metatietohallintaohjelma pitää huolen, että vain yksi isäntätransaktionhallintaohjelma on käytössä ja osiot ovat yksinomaan vain yhden omistustransaktiohallintaohjelman hallinnoitavana. Metatietohallintaohjelma pitää myös yllä karttaa omistustransaktiohallintaohjelmien osioista. Koska metatietohallintaohjelmalla on keskeinen osa järjestelmän toiminnassa, myös niitä on useampia käynnissä samanaikaisesti virheensietokyvyn parantamiseksi. (Das ym., 2010)

4.3 Pilvimaailman haasteet

ElasTras pyrkii yhdistämään seuraavat pilvisovelluksen tietokannan hallintajärjestelmälle ihanteelliset ominaisuudet (Das ym., 2010 s. 2):

- **Skaalautuvuus**. Pilven tietokoneresurssien ehtymättömyyden illuusion hyödyntämiseksi tietokannan hallintajärjestelmän pitää pystyä skaalautumaan sekä datan määrän, että käyttöasteen mukaan.

- **Joustavuus.** Pilven dynaamisesti kuormitukseen sopeutuvista resursseista maksetaan käytön mukaan jatkuvan laskutuksen periaatteella. Kustannustehokkaasti toimivan tietokannan hallintajärjestelmän pitää pystyä dynaamisesti lisäämään ja luovuttamaan resursseja kuormituksen mukaan.
- **Virheensietokyky.** Pilven koostuessa edullisista komponenteista, on tietokannan hallintajärjestelmän varauduttava säilyttämään toimintakykynsä laitevirheiden sattuessa.
- **Transaktionaaliset toiminnot.** Tietokannan transaktioiden atomisuus, eheys ja pysyvyys helpottavat datan käsittelemistä ja yksinkertaistavat sovellusten suunnittelemista. Transaktionaalisia ominaisuuksia kaivataan myös pilvisovellusten tietokannan hallintajärjestelmiltä.

ElasTraS tarjoaa relationaalisen tietomallin ja pilviympäristössä vaikeasti taattavia ACID-ominaisuuksia tietokannan osiojoukon sisällä (Abadi, 2009; Das ym., 2010). Huomioitavaa on, että tämän joukon koko riippuu siitä, kuinka monta osiota yksittäinen omistustransaktiohallintaohjelma hallinnoi (Das ym., 2010). Näin tietokannan osittamisella on skaalautuvuuden ja joustavuuden lisäksi vaikutus siihen, kuinka suureen osaan tietokannasta voidaan tietyn transaktion yhteydessä tukea ACID-ominaisuuksia (Das ym., 2010). Tietyn osion omistaa vain yksi omistustransaktiohallintohjelma, mikä tarkoittaa, että tietokanta on jatkuvassa ristiriidattomassa tilassa (Das ym., 2010).

Pilvimaailmassa, missä tieto on tallennettuna kolmannella osapuolella, on läsnä huoli tietoturvasta (Abadi, 2009). Oletettavasti tietoturvaa voidaan toteuttaa ElasTraS-tietokannan hallintajärjestelmässä samalla tavalla kuin perinteisissä relationaalisissa tietokannoissa myöntämällä luku- ja kirjoitusoikeuksia riippuen käyttäjistä ja tauluista, tai käyttämällä salausalgoritmeja.

Virheensietokykyä voidaan pitää pakollisena ominaisuutena pilven tietokannan hallintajärjestelmille (Pritchett, 2008). Pilvisovelluksen tietokannan pitää säilyttää toimintakykynsä, vaikka osa järjestelmän laitteista kaatuisi (Abadi, 2009). ElasTraS toteuttaa virheensietokykyä ajamalla samanaikaisesti useita järjestelmän toiminnan kannalta oleellisia komponentteja, kuten isäntätransaktiohallintaohjelmaa ja metatietohallintaohjelmaa (Das ym., 2010). Komponentin kaatuessa voidaan ottaa käyttöön seuraavassa jonossa oleva kopio, joka on jatkuvan synkronoinnin ansiosta valmiiksi samassa tilassa, kun kaatunut komponentti (Das ym., 2010). Kaatuneiden omistustransaktiohallintaohjelmien palauttaminen aiheuttaa kuitenkin hetkittäisen käyttökatkoksen niille osioille, joita kaatunut palvelin hallinnoi (Maia, Armendáriz-Iñigo, Ruiz-Fuertes & Oliveira, 2010).

Pilvisovelluksen tietokannan hallintajärjestelmältä vaaditaan myös skaalautuvuutta ja joustavuutta, mikä on ElasTraS-tietokannan hallintajärjestelmässä toteutettu osittamalla tietokantaa skeeman tasolla (Abadi, 2009; Das ym., 2010). Hyvin useat eri skeemat voidaan osittaa niin, että toisiinsa liittyvä tieto, jota yleensä käsitellään yhdessä, voidaan osittaa samaan osioon (Das ym., 2010).

Käytännössä tämä kuitenkin tarkoittaa, että tietokannan skaalautuvuus ElasTraS-tietokannan hallintajärjestelmässä on riippuvainen siitä, kuinka hyvin tietokannan skeemaa voidaan osittaa (Maia ym., 2010). Myös yksittäisen osion sisältäessä keskenään relevantit taulut, ja tämän osion kasvaessa liian suureksi, muodostuu tästä osiosta pullonkaula, kun osioita ei voida enää jakaa (Maia ym., 2010). ElasTraS onkin skaalautuva ja joustava tietokannan hallintajärjestelmä palvelimien määrän pysyessä kymmenissä ja tiedon määrä terabitti-tasolla (Das, 2010). Tämän ovat osoittaneet myös kokeet, joissa käyttöasteen kasvaessa palvelimien lisääminen samassa suhteessa pitää vasteajat lähes samana, ja käyttökustannukset joustavat palvelimien määrän mukaan (Das ym., 2010). Kuormituksen kasvaessa järjestelmä kuitenkin kohtaa skaalautuvuuden rajansa ja vasteajat kasvavat nopeasti kohtuuttoman suureksi (Das ym., 2010).

5 YHTEENVETO

Tutkielmassa on käsitelty sitä, miten pilvilaskenta on luonut uusia haasteita tietokannan hallintaan. Pilvisovelluksen tietokannan hallintajärjestelmän pitää käyttöasteen muuttuessa pystyä skaalautumaan horisontaalisesti osittamalla tietokanta lukuisille dynaamisesti lisääntyville koneille (Feuerlicht, 2010; Pokorný, 2011). Tämän lisäksi tekee tiedon hallinnan pilvessä ongelmalliseksi huoli tietoturvasta ja tiedon kopioiminen yli suurten välimatkojen, mikä tekee ristiriidattomuuden hallinnan vaikeaksi (Abadi, 2009). Brewerin (2000) CAP-teoreeman pohjalta on todettu, että pilven kaltaisessa hajautetussa järjestelmässä joudutaan tekemään valintoja ristiriidattomuuden, saatavuuden ja osittamistoleranssin väliltä. Pilven koostuessa edullisista komponenteista on varauduttava koneiden kaatumiseen, joten osittamistoleranssi on pakollinen ominaisuus pilvisovelluksen tietokannan hallintajärjestelmille (Pritchett, 2008; Pokorný, 2011). Tämän pohjalta on todettu, että pilvisovellukselle sopivalta tietokannan hallintajärjestelmältä vaaditaan siis dynaamista horisontaalista skaalautuvuutta, tieturvan hallintaa ja osittamistoleranssia (Abadi, 2009; Pritchett, 2008; Pokorný, 2011). Tämän jälkeen joudutaan tekemään valintoja tietokannan ristiriidattomuuden ja saatavuuden väliltä (Pritchett, 2008).

Monet verkkosovelluspalvelut voivat sietää väliaikaista ristiriidattomuutta, mutta vasteajoille on olemassa pakollisia rajoitteita ja pitkät vasteajat voivat tulla kalliiksi palveluntarjoajalle (Abadi, 2009). Pilvisovelluksille kehitetetyt NoSQL-tietokannan hallintajärjestelmät suosivatkin ACID-ominaisuuksien sijasta BASE-ominaisuuksia (Pokorný, 2011). ACID-ominaisuuksia tukevassa tietokannassa tieto on aina ristiriidattomassa tilassa, mutta se ei anna mitään takeita vasteajoista tiedon käsittelemiseen. BASE-ominaisuudet takaavat että tieto on aina saatavilla tietyssä vasteajassa, mutta tieto voi olla väliaikaisesti ristiriidattomassa tilassa (Pritchett, 2008).

Tutkielmassa käsiteltiin kolmea keskeistä NoSQL-tietokantatuotetta: Googlen Bigtable-, Amazonin S3- ja Yahoo!:n PNUTS-tietokannan hallintajärjestelmiä. Tämän lisäksi käsiteltiin myös hybridi-muotoista ElasTraS-tietokantatuotetta.

Kaikki kolme esiteltyä NoSQL-tietokantatuotetta on todettu pilvisovelluksen tarpeisiin skaalautuviksi tietokannan hallintajärjestelmiksi. Horistontaalisen skaalautuvuuden mahdollistamiseksi näiden tietomallit eivät perustu relaationaalisiin tauluihin, vaan joustavampiin rakenteisiin (Pokorný, 2011). Esimerkiksi Googlen Bigtable-taulut tarjoavat mielenkiintoisen kolmiulotteisen kartan, jolloin solusta voidaan säilyttää useita eri versioita (Chang ym.,2008). Bigtable-klustereiden välisestä kopioimisesta ei löytynyt dokumentaatiota, mutta esitelty PNUTS-tietokantatuote hyödyntää tietueiden versioita ristiriidattomuuden hallinnassa PNUTS-alueiden välillä (Cooper ym., 2008). Amazonin S3-tietokantatuote on hyvä esimerkki BASE-ominaisuuksia tukevasta tietokannan hallintajärjestelmästä, missä tieto voi olla väliaikaisesti ristiriitaisessa tilassa (Brantner ym., 2008). Tässä tietokannan hallintajärjestelmässä voidaan parantaa tiedon tuoreutta, mutta tämä lisää sen käyttökustannuksia (Brantner ym., 2008). Sen lisäksi, että ristiriidattomuus voidaan saavuttaa pilvessä ainoastaan saatavuuden kustannuksella, ristiriidattomuuden lisääminen pilvessä voi nostaa myös käyttökustannuksia. PNUTS tarjoaa mielenkiintoisen ohjelmistorajapinnan, jonka avulla käyttäjä voi itse vaikuttaa siihen, milloin tarvitaan tiukempia takeita ristiriidattomuudesta (Cooper ym., 2008). PNUTS-tietokanta takaa BASE-ominaisuuksien lisäksi myös sen, että tietueen päivitykset saapuvat kaikkiin kopioihinsa päivitysjärjestyksessä (Cooper ym., 2008). Ominaisuutta kutsutaan aikajanaristiriidattomuudeksi (Cooper ym., 2008).

Relaatioista luopuminen NoSQL-tietokantatuotteissa on tarkoittanut myös, että käyttäjälle jää suurempi vastuu tiedon oikeellisuuden hallinnassa ja, että esimerkiksi liitos- ja järjestysfunktioista on jouduttu luopumaan. Tärkeimpänä puutteena voidaan kuitenkin pitää ACID-ominaisuuksien puuttumista, jotka ovat yritysten liiketoiminnassa käytettäville tietokannoille pakollisia (Pokorný, 2011).

ElasTraS pyrkii tarjoamaan skaalautuvaa relationaalista tietomallia ja ACID-ominaisuuksia pilvisovelluksille (Das ym., 2010). ElasTraS-tietokannan skaalautuuvuus riippuu kuitenkin mahdollisuudesta osittaa tietokannan skeema ja parhaimmillaankin ElasTraS-tietokanta skaalautuu tehokkaasti vain kymmenien palvelimen ja terabitin tietokannoille, jolloin samanaikaisia käyttäjiä voi olla vain tuhansia (Das ym., 2010). Tämä voi olla tarpeeksi monien yritysten sisäiseen käyttöön, missä käyttöastetta voidaan arvioida hyvin etukäteen, mutta tämä kuitenkin rikkoo pilvilaskennan tietokoneresurssien ehtymättömyyden illuusion, mitä voidaan pitää yhtenä oleellisimpana pilvilaskennan ominaisuutena (Das ym., 2010; Armbrust ym., 2009).

Tutkielman tulosten pohjalta voidaan päätellä, että pilvisovelluksille kehitetyt tietokannan hallintajärjestelmät sopivat kuluttajille suunnatuille verkkosovelluspalveluille, kuten yhteisöpalveluille, joiden käyttöastetta on vaikea arvioida tai käyttöasteessa on suurta vaihtelua. Sen sijaan esimerkiksi kuluttajille suunnattujen verkkopankki-palveluiden toteuttaminen pilvisovelluksena ei ole mielekästä, johtuen jatkuvan ristiriidattomuuden vaatimuksesta. Jatkuvaa ristiriidattomuutta voidaan toteuttaa pilvisovelluksen tietokannan hallintajärjestelmässä, jos käyttöasteen tiedetään pysyvän kohtuullisena. Mikäli kuitenkin

käyttöastetta voidaan arvioida etukäteen, voi oman datakeskuksen käyttäminen olla käytännöllisempää, jolloin ongelma relaationaalisen tietokannan skeeman kyvystä osittua eri palvelimille sulkeutuu pois. Joissakin tapauksissa pilvilaskennan käyttäminen voi tosin olla oman datakeskuksen käyttämistä edullisempaa myös käyttöasteen pysyessä tasaisena johtuen pilven datakeskusten keskittämisestä alueille, missä on edulliset sähkö- ja tietoverkkokulut (Armbrust ym., 2009).

Tutkimuksen tuloksia voi käyttää pilvisovellukseensa sopivaa tietokannan hallintajärjestelmää etsivät verkkosovelluspalvelujen kehittäjät vertaillessaan tietokantatuotteiden sopivuutta heidän sovellukseensa tarpeisiin. Tämän lisäksi pilvilaskennan hyödyntämisestä kiinnostuneet ohjelmistokehittäjät voivat tutkielman perusteella arvioida ohjelmistojensa sopivuutta pilvimaailmaan.

Tutkielmassa käsiteltyjä pilvisovelluksille suunnattuja tietokantatuotteita tarkasteltiin hyvin suppeasti johtuen osaksi kandidaatintutkielman rajoitteista, kuin myös siitä, että pilvisovelluksen tietokannan hallintajärjestelmä on konseptina uusi, joten lukuisista pilvisovelluksille suunnatuista tietokannan hallintajärjestelmistä löytyy vain muutamia tieteellisiä artikkeleja. Tutkielman luonteen johdosta myöskään käytännön vertailu eri tietokantatuotteiden ominaisuuksien välillä ei ollut mielekästä. Luonteva jatkotutkimuskohde voisi ollakin eri pilvisovelluksille suunnattujen tietokantatuotteiden vertailu käytännön kokeilla, missä voitaisiin verrata esimerkiksi eri tietokantatuotteiden vasteaikoja ja käyttökustannuksia eri kuormituksilla

Tutkielmassa keskityttiin pelkästään transaktioperusteisiin tietokannan hallintajärjestelmiin, joten toisena jatkotutkimuskohteena voitaisiin tutkia analyysiperusteisia tietokannan hallintajärjestelmiä pilvimaailmassa. Mielenkiintoista olisi selvittää, miten tietovarastoja ja tiedon louhintaa voidaan toteuttaa pilvimaailmassa.

LÄHTEET

Abadi, D. J., (2009). Data management in the cloud: Limitations and opportunities. *Bulletin of the Technical Committee on Data Engineering, 32(1),* 3-12.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M. (2009). *Above the clouds: A berkeley view of cloud computing (Report UCB/EECS-2009-28), Electronic Engineering and Computer Sciences Department, University of California, Berkeley.*

Brantner, M., Florescu, D., Graf, D., Kossmann, D., & Kraska, T. (2008). Building a database on S3. Teoksessa *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, Canada (s. 251-264)*

Brewer, E. A., (2000). Towards Robust Distributed Systems. PODC 2000. Haettu 4.4.2012 osoitteesta http://openstorage.gunadarma.ac.id/~mwiryana/Kuliah/Database/PODC-keynote.pdf

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., et al. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems, 26(2)*, 4:1-4:26.

Cooper, B. F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H., et al. (2008). PNUTS: Yahoo!'s hosted data serving platform. Teoksessa *Proceedings of VLDB Endowment, Auckland, New Zealand, August 23-28 (s.1277-1288).*

Das, S., Agarwal, S., Agrawal, D., & Abbadi, A. E. (2010). *ElasTraS: An elastic, scalable, and self managing transactional database for the cloud (UCSB Computer Science Technical Report 2010-04) University of California, Santa Barbara, Department of Computer Science.*

Feuerlicht, G. (2010). Database trends and directions: current challenges and opportunities. Teoksessa J. Pokorný, V.Snásel & K. Richta (toim.) *DATESO 2010, (s. 163-174)*

Feuerlicht, G., Pokorný, J. (2011). Can relational DBMS scale-up to the cloud? *Esitetty ISD 2011-konferenssissa*, *Edinburgh, Scotland.*

Gilbert, S., Lynch, N. (2002). Brewer's conjecture and the feasibility consistent, available, partition-tolerant web services. *ACM SIGACT News*, *33(2)*, 51-59

Korpinen, S. (2011) *Sovelluksen siirtäminen pilviympäristöön, Tietojenkäsittelytieteiden laitos, Pro Gradu-tutkielma, Jyväskylän yliopisto, Jyväskylä*

Lakshman, A., & Malik, P. (2010). Cassandra: A decentralized structured storage system. *ACM SIGOPS Operating Systems Review, 4(2)*, 35-40.

Maia, F., Armendáriz-iñigo, J., Ruiz-Fuertes, M., & Oliveira, R. (2010). Scalable transactions in the cloud: Partitioning revisited. Teoksessa R. Meersman (toim.) *On the move to meaningful internet systems, OTM 2010 (s. 785-797) Springer Berlin / Heidelberg.*

Pritchett, D. (2008). BASE: An Acid Alternative. *ACM Queue, 6(3),* 48-55

Pokorný, J. (2010). Databases in the 3rd millennium: Trends and research directions. *Journal of Systems Integration, 1*, 1-2

Pokorný, J. (2011). NoSQL Databases: a step to database scalability in Web environment. Teoksessa *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, Ho Chi Minh City, Vietnam (s. 278-283)*

Vaquero, L. M., Rodero-Merino, L., Caceres, J. & Lindner, M. (2009). A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communications Review, 39(1)*, 50–55.