Pertti Mäki-Välkkilä

NoSQL tietokantojen skaalautuvuus

Tietotekniikan Kandidaatintutkielma

30. huhtikuuta 2021

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Pertti Mäki-Välkkilä

Yhteystiedot: pejama@student.jyu.fi

Työn nimi: NoSQL tietokantojen skaalautuvuus

Title in English: Scalability of NoSQL databases

Työ: Kandidaatintutkielma

Sivumäärä: 20+0

Tiivistelmä: NoSQL tietokantojen käyttö varsinkin verkkosovellusten tietokantaratkaisuina on yleistynyt suuresti. Yleistymistä on edesauttanut muun muassa NoSLQ tietokantojen tietomallien joustavuus verkkosovelluksen tarpeiden mukaan. Tallennettavan tiedon ja tietokantakyselyiden määrien kasvaessa tietokannan skaalautuvuus nousee myös olennaiseksi kriteeriksi tietokantaa valittaessa. Tutkielmassa tarkastellaan NoSQL tietokantojen skaalautuvuutta perehtyen aihetta koskeviin tutkimuksiin. NoSQL tietokantojen skaalautuvuuteen liittyvän yleisesti liittyvän tiedon lisäksi tarkastellaan tarkemmin MonoDB:n ja Couchbase Serverin skaalautuvuuden toteutustapoihin.

Avainsanat: tietokanta,NoSQL,skaalautuvuus,MongoDB, Couchbase Server

Abstract: The use of NoSQL databases has increased signicantly especially among the webapplication. There are several reasons for this increase such as the possibility to modify the data model flexibly according the needs of the applications. While the amount of saved data increases and the database transactions rise, the scalability of the database becomes a vital criteria in database selection. In this thesis the scalability of NoSQL databases is studied. The study is based on existing research concerning the topic. In addition to general information about the NoSQL scalability, the thesis will study more the methods of scalaling implementation of MongoDB and Couchbase Server.

Keywords: database, NoSQL, scalability, MongoDB, Couchbase Server

Sisällys

1 Johdanto

Verkkopohjaisten sovellusten suosion kasvun myötä on myös NoSQL tietokantojen suosio on kasvanut kiihtyvällä tahdilla. Kymmenen suosituimman tietokannan listalta löytyy jo neljä erityyppistä NoSQL tietokantaa (DB-Engines). Valittavissa on siis useita NoSQL-vaihtoehtoja, joiden välillä on kuitenkin erovaisuuksia muun muassa suorityskyvyssä, skaalautuvuudessa sekä viansietokyvyssä.

Tallennettavan tiedon määrä on kasvanut vuosittain, viimeisen kahden vuoden aikan jopa eksponentiaalisesti (*How Much Data Is Created Every Day in 2021?* 2021). Jatkuvasti lisääntyvä tietomäärä ja sitä myötä tietokantahakujen määrällinen kasvu ovat nostaneet tietokantojen skaalattavuuden helppouden tärkeäksi ominaisuudeksi.

Tämä tutkielman tarkoituksena on selvittää millaisia mekanismeja NoSQL tietokantojen skaalauksessa käytetään. Mekanismien selvittämisen lisäksi tarkastellaan miten MongoDB sekä Couchbase Server ovat skaalauksen toteuttaneet. Nämä ovat kaksi yleisimmin käytössä olevaa dokumentti-orientoitunutta tietokantaa. MongoDB on näistä kahdesta suositumpi ja samalla yleisin kaikista NoSQL tietokannoista. Couchbase Server on dokumenttiorientoituneista tietokannoista toiseksi yleisin ja kaikkien NoSQL tietokantojen vertailussa yhdestoista.

Tutkielmassa selvitetään aluksi mitä NoSQL tietokannalla tarkoitetaan sekä minkä tyyppisiä NoSQL tietokantoja on olemassa. Tämän jälkeen tarkastellaan skaalautuvuuden määritelmiä. Tähän aiheeseen kuuluvat muun muassa skaalautuvuuden suunnat sekä horisontaalisen skaalautuvuuden eri toteutustavat. Skaalautuvuuden toteutustapojen tarkempaan tarkasteluun on valittu kaksi dokumentti-oritentoitunutta tietokantaa, MongoDB ja Couchbase Server.

2 NoSQL tietokannat

Tässä luvussa tarkastellaan tutkielmaan liittyviä tietokantakäsitteitä. On tärkeää ymmärtää mitä NoSQL tietokannalla tarkoitetaan. Itse NoSQL käsitteen lisäksi käydään lyhyesti läpi erityyppiset NoSQL tietokannat, miten ne rakentuvat ja mitkä ovat niiden yleisimmät käyttökohteet.

2.1 NoSQL tietokannat

Termiä NoSQL käytti ensikerran Carlo Strozzi vuonna 1998. Strozzi nimesi kehittämänsä uuden avoimen lähdekoodin tietokannan NoSQL:ksi (Berg, Seymour ja Goel 2012). Kehitetty tietokanta ei käyttänyt tiedonhakuun tai -tallentamiseen lainkaan perinteistä SQL kyselykieltä. NoSQL nousi uudelleen esiin kun Eric Evans Rackspacelta esitteli termin Last.fm:n Johan Oskarssonin järjestämässä keskustelutilaisuudessa, jossa käsiteltiin avoimen lähdekoodin hajautettuja tietokantoja.

Terminologisesti NoSQL tietokannoiksi kutsutaan tietokantoja, jotka eivät noudata perinteistä relaatiotietokannoista tuttua taulukko-skeemaa (“Definition of NoSQL” 2021). Kaksiulotteisen taulukon sijaan tieto tallennetaan joustavampiin rakenteisiin kuten esimerkiksi avain-arvo -pareihin tai dokumentteihin (*NoSQL* 2021).

NoSQL tietokannat luokitellaan eri lähteiden mukaan kolmesta viiteen eri tyyppiin tietorakenteen mukaan. Lähteille yhteiset tyypit ovat avain-arvo -tietokannat (key-value store), sarakkeiset tietokannat (widecolumn/column-oriented database) ja dokumentti-orientoituneet tietokannat (document store database) (Jing Han ym. 2011). Näiden kolmen tyypin lisäksi, omiksi NoSQL tietokantatyypeiksi joissain lähteissä erotetaan myös graafitietokannat (graph database) (Moniruzzaman ja Hossain 2013) sekä objekti-orientoituneet tietokannat (object oriented database) (Nayak, Poriya ja Poojary 2013).

2.2 Avain-arvo tietokanta

Avain-arvo tietokanta perustuu kirjaimista ja numeroista koostuviin avainmerkkijonoihin sekä niihen liityviin arvoihin. Arvot voivat olla yksinkertaisimmillaan numeroita tai merkkejä. Avaimiin voidaan kuitenkin liittää myös monimutkaisempaakin tietoa kuten taulukoita tai objekteja.

Avain-arvo tietokanta on hakuja tehtäessä erittäin nopea ja sopii sitä kautta hyvin esimerkiksi käyttäjäprofiilien hallintaan ja tuotetietojen hakuun verkkokaupoissa. Heikkouksiin avain-arvo tietokantojen osalta voidaan lukea se, että haku voidaan pääsääntöisesti ulottaa ainoastaan avaimiin.

2.3 Dokumentti-orientoitunut tietokanta

Dokumentti-orientoituneessa tietokannassa tieto tallennetaan kantaan nimensä mukaisesti dokumenttimuotoisena. Dokumenttien formaattina voivat toimia esimerkiksi JSON, XML tai PDF.

Tietokannan jokaiselle dokumentille luodaan tallennettaessa tunniste, joka identifioi kyseessä olevan dokumentin eksplisiittiseti. Luotua tunnistetta voidaan käyttää hakukriteerinä muiden dokumentin kenttien ohella (Manoj 2014).

Tietokannan dokumentit muistuttavat relaatiotietokannan tietuita mutta skeemattomuudellaan ovat joustavampia verrattuna relaatiotietokantoihin. Käytännössä tämä tarkoittaa sitä, että dokumentti-orientoituneessa tietokannassa saman tietokantataulun tai -kokoelman dokumentit voivat poiketa rakenteeltaan toisistaan huomattavasti (Nayak, Poriya ja Poojary 2013).

Parhaiten dokumentti-orientointuneet tietokannat käyvät sellaisten sovellusten käyttöön, missä tallennettava tieto on monimuotoista ja jäsennettävissä dokumenttimuotoon. Hyvä esimerkki tällaisesta on esimerkiksi sisällönhallintajärjestelmä.

2.4 Sarakkeinen tietokanta

Sarakkeiset tietokannat muistuttavat etäisesti SQL tietokantoja siinä mielessä, että myös sarakkeisissa tietokannoissa tieto tallennetaan taulunomaisiin tietorakenteisiin. Taulurakenteiden välillä ei kuitenkaan ole riippuvuussuhteita SQL:n tavoin.

Sarakkeista tietokantaa voi ajatella suurena joukkona tietokantasarakkeita, jotka muodostuvat kontekstuaalisesti toisiinsa liittyvistä tiedoista. Tietokannan jokainen sarake pitää sisällään rivejä, joista jokainen koostuu jälleen sarakkeista. Rivin jokainen sarake voi sisältää joko varsinaisen tiedon avain-arvo parina tai vaihtoehtoisesti samaan kontekstiin liittyvää sarakkeista tietoa. Varsinainen tieto on tallennettuna avain-arvo pareihin.

Sarakkeisen tietokannan rakenne mahdollistaa tietojen keräämisen hyvin pienellä määrällä siirrettävää tietoa. Vasta haun ulottaminen sarakkeiden sisään lisää tiedon siirron määrää.

2.5 Graafitietokanta

Graafitietokanta rakentuu tieto-objekteista, solmuista (nodes), jotka kiinnittyvät toisiinsa relaatioiden, särmien (edges) kautta. Solmujen sisältämä tieto on tallennettu avain-arvo pareihin. Solmujen ja särmien muodostama relaatioiden verkko saa aikaan visuaalisesti hahmotettavan mallin tiedosta. Moniuloitteisesti ajateltuna tietokannan rakenne muistuttaa graafista muotoa, mistä tietokantatyypin nimi juontaakin juurensa.

Graafitietokantojen erikoisuutena voidaan pitää lähetymistapaa tallennettuun dataan. Varsinaisen tiedon sijaan graafitietokanta korostaa tietojen välisiä relaatioita. Tämän ominaisuuden ansiosta graafitietokantoja käytetään eritoten sosiaalisen median eri sovelluksissa.

2.6 Objekti-orientoitunut tietokanta

Objekti-orientoitunut tietokanta on yhdistelmä tietokantaa sekä olio- ohjelmointia. Olio-ohjelmointi tuo mukanaan tietokantaan sellaisia ominaisuuksia kuten tiedon kapselointi, polymorfismi sekä periytyvyys. Jokainen tallennettu tieto-objekti pitää sisällään myös siihen liittyvät toiminnallisuudet, mikä varmistaa tallennetun tiedon muuttumattomuuden.

Objekti-oritentoituneita tietokantojen hyödyt nousevat esiin esimerkiksi sovelluksissa missä tietoa sisältävät objektit muodostavat keskenään monimutkaisia riippuvuussuhteita tai sovelluksen sisällä objektien rakenteita muunnellaan (Saxena ym. 2012).

Yleisesti objekti-orientoituneita tietokantoja käytetään muun muassa telekommunikaatiossa esimerkiksi verkonhallinnan saralla (Raatikainen ja Porkka 1998) sekä tieteellisessä tutkimuksessa, joista mainittakoon molekyylibiologia sekä hiukkasfysiikan tutkimus. Tietokantatyyppi mahdollistaa nopean tiedon haun ja tiedon muuttumattoman tallentamisen. Varjopuolena näille ominaisuuksille voidaan kuitenkin lukea tietokannan riippuvuuden valitusta ohjelmointikielestä (Curator 2021).

3 Tietokannan skaalautuvuus

Termin 'skaalautuvuus' (scalability) tai tarkemmin sanan perusmuotona toimivan adjektiivin 'skaalautuva' (scalable) määritelmä on 'Kyky olla helposti laajennettavissa tai päivitettävissä tarpeen mukaan' (*Scalability* 2021). Järjestelmien ja tietokantojen osalta tämä tarkoittaa kykyä mukautua kasvavaan tarpeeseen prosessoida tietoa (Ali 2019).

Käytännössä kasvavaan tarpeeseen vaikuttaa kolme tekijää, jotka vuorovaikuttavat keskenään läheisesti. Nämä tekijät ovat yhtäaikaisten käyttäjien määrä, prosessointi kuorma sekä tietomäärä (Lake ja Crowther 2013). Yhtäaikaisten käyttäjien määrän nousu aiheuttaa lisäkuormaan prosessointiin kasvavien tietokantapyyntojen muodossa. Tietomäärän kasvu taas sekä lisää prosessointi tarvetta tietokantahakujen vaativuuden nousun kautta että todennäköisesti kasvattaa käyttäjämääriä. Lisäksi kasvava tietomäärä vaatii massamuistin koon kasvattamista. Joka tapauksessa päädytään tarpeeseen skaalata tietokantaa helposti.

3.1 Skaalautuvuussuunnat

Tietokannan skaalautuvuutta voidaan lähestyä kahdella eri tavalla. Tietokantoja voidaan skaalata vertikaalisesti (scale up) tai horisontaalisesti (scale out).

Perinteisesti SQL tietokantoja on skaalattu vertikaalisesti. Käytännössä vertikaalinen skaalaus tarkoittaa tietokannan osalta tietokantaa ajavan järjestelmän päivittämistä tehokkaammaksi. Tämä voi tarkoittaa esimerkiksi prosessorin päivittämistä tehokkaampaan, järjestelmän muistin määrän lisäämistä tai tallennus median päivittämistä nopeampaan kuten esimerkiksi kiintolevyn vaihtamista SSD-levyyn (Oussous ym. 2013).

Vertikaalinen skaalaus vaatii monasti suuriakin investointeja sekä korkean tason osaamista. Näiden reunaehtojen täyttyessäkin vertikaalinen skaalaus saattaa epäonnistua (Pokorny 2013).

Horisontaalisessa skaalauksessa lähestymiskulma kasvaneiden tietokantapyyntöjen tarpeiden tyydyttämiseen poikkeaa merkittävästi vertikaalisesta skaalaamiseta. Sen sijaan, että päivitettäisiin tietokantaa ajavaa järjestelmää tehokkaammaksi, pyritään jakamaan tarvittava

työkuorma useammalle palvelimelle. Jokainen laite toimii itsenäisenä yksikkönä parantaen tietokantakokonaisuuden suorituskykyä (Ali 2019).

Horisontaalisessa skaalaamisessa on sekä etuja että haittapuolia verrattuna vertikaalisen skaalamiseen. Selkeä etu on se, että tietokannan tehon lisääminen on selkeästi edullisempaa. Yhden järeän palvelimen sijaan voidaan käyttää useita pienempiä ja samalla edullisempia laitteita rinnakkain. Kääntöpuolena usean palvelimen klusterille on esimerkiksi tiedon eheyden mahdollinen vaarantuminen. Tietokannan hajauttaminen eri palvelimille kasvattaa myös kokonaisuuden monimutkaisuutta (Oussous ym. 2013).

3.2 Horisontaalisen skaalauksen toteutustavat

Tietokannan hajauttaminen ja sitä kautta tietokannan skaalaaminen horisontaalisesti, voidaan toteuttaa eri tavoilla. Tietokanta voidaan toisintaa eri laitteistoille isäntä-orja (Master-Slave) periaatteella, se voidaan sirpaloida (Sharding) osiin tai tietokanta voidaan hajauttaa Dynamo mallin (Dynamo Model) mukaisesti (Tauro, Aravindh ja Shreeharsha 2012).

3.2.1 Isäntä-orja toisintaminen

Isäntä-orja toisintamisessa tietokanta on monistettu usealle eri palvelimelle eli solmulle. Solmuista vain yksi voi toimia isäntänä, loput solmut toimivat orjina isäntäsolmulle. Isäntäsolmun tehtävänä on hoitaa yksinään kaikki tietokannan kirjoituspyynnöt, kun taas jokainen järjestelmään kuuluvan solmu vastaa tietokantaan tuleviin hakupyyntöihin. Isäntäsolmun tekemä muutos tietokantaan toisintuu jokaiselle orjasolmulle ja sitä kautta saatavaksi hakupyyntöjä varten (Kuznetsov ja Poskonin 2014).

Isäntä-orja toisintamisella pyritään skaalaamaan tietokannan vasteaikaa eli pienentämään tietokantahakuihin kuluvaa aikaa. Orjasolmuja lisäämällä yhtäaikaisesti palveltavien hakujen määrää voidaan nostaa. Isäntä-orja toisintaminen joutuu kuitenkin vaikeuksiin tilanteissa missä kirjoituspyyntöjen määrä kasvaa. Koska kaikki tietokannan päivitykseen liittyvät pyynnöt käsitellään yhdellä ainoalla solmulla, pyyntöjen määrä saattaa ylittää isäntäsolmun suorituskyvyn kapasiteetin.

3.2.2 Sirpalointi

Sirpaloitaessa tietokanta jaetaan täysin itsenäisesti toimiviin osatietokantoihin eli sirpaleisiin. Tietokanta voidaan jakaa osiin käyttäen sirpalointiavainta, jonka avulla jako voidaan tehdä esimerkiksi aakkosjärjestyksessä. Jos kanta sirpaloidaan esimerkiksi viiteen sirpaleeseen, ensimmäinen sirpale pitää sisällään sirpalointiavaimet A:sta E:hen, toinen sirpale sirpalointiavaimet F:stä J:hin ja niin edelleen (Tauro, Aravindh ja Shreeharsha 2012).

Tietokantahakua tehtäessä, sirpalointiavain määrittää sen sirpaleen mistä haettava tieto löytyy. Jokaisen sirpaleen toimiessa itsenäisesti, sekä tietokanta haut että muutokset tietokantaan tapahtuvat kyseisessä sirpaleessa. Tällä ratkaisulla tietokantatapahtumien työkuorma voidaan jakaa sirpaleiden kesken kuormittamatta yhtä yksittäistä sirpalette liikaa.

Vaikka työkuormaa voidaan edelleen pienentää lisäämällä palvelimia, yhden uuden sirpalepalvelimen lisääminen vaatii koko tietokannan alasajamisen ja tietojen uudelleen järjestelyn sirpalepalvelimien välillä.

3.2.3 Dynamo malli

Dynamo malli on Amazonin kehittämä ratkaisu, jolla vähennetään tietokannan sirpaloinnin ongelmia sirpalepalvelimien määrää muutettaessa. Dynamo toimii samalla periaatteella kuin tietokannan sirpalointi mutta se käyttää sirpaloinnin apuna johdonmukaista hajauttamista (consistent hashing) (Weintraub 2014).

Johdonmukaisessa hajauttamisessa jokaiselle tietoyksikölle lasketaan sen avaimesta hajautusalgoritmilla H(avain) oma yksilöllinen arvo. Näistä arvoista muodostetaan kehä, jonka solut numeroituvat myötäpäivään H(avain) hajautusalgoritmilla saatujen arvojen mukaisesti välille nolla ja M-1, missä M on tietoyksiköiden määrä. Muodostuvalle kehälle sijoitetaan käytettävissä olevat palvelimet laskemalle niille arvo käyttäen hajautusalgorimia H(palvelin) niin, että ne vastaavat yhtä hajautusalgoritmilla H(avain) saatua arvoa.

Järjestelmään tulevalle tietokantapyynnölle lasketaan arvo hajautusalgoritmilla H(avain), joka siis vastaa yhtä kaikista tietoyksiköistä muodostetun kehän arvoa. Järjestelmä etsii kyseisen kohdan arvokehältä ja lähtee kulkemaan kehää myötäpäivään numero kerrallaan kunnes

kohtaa kehälle sijoitetun palvelimen. Näin löydetty palvelin ottaa tietokantapyynnön vastaan ja suorittaa halutun pyynnön (Karger ym. 1999).

Kun Dynamo mallia skaalataan lisäämällä uusi palvelin, sille lasketaan oma sijainti kehälle hajautusalgorimilla H(palvelin). Johdonmukaisen hajauttamisen ansiosta muutos ei vaadi tietokannan alasajoa vaan lisäyksen vaikutus koskee ainoastaan kehällä myötäpäivään mentäessä seuraavaa palvelinta.

4 Dokumentti-orientoituneiden tietokantojen skaalautuvuus

Tässä kappaleessa tarkastellaan kahta dokumentti-orientoitunutta tietokantaa. Tietokantojen lyhyiden esittelyjen lisäksi käsitellään millä tavoin valitut tietokannat toteuttavat skaalamisen.

4.1 MongoDB

NoSQL-tietokantatyypeistä varteenotettavin haastaja SQL-tietokannoille, markkinaosuuksia tarkasteltaessa, on dokumentti-orientoitunut tietokanta MongoDB (DB-Engines). MongoDB kehitettiin vuonna 2007 Dwight Merrimanin, Eliot Horowitzin ja Kevin Ryanin toimesta. Ryhmä työskenteli yrityksessä nimeltä DoubleClick, joka toimi verkkomarkkinoinnin parissa. Yrityksen järjestelmät palvelivat jopa 400 000 verkkomainosta sekunnissa ja tämä aiheutti haasteita sen aikaisille tietokantaratkaisuille. Ongelman ratkaisuna ryhmä kehitti tietokannan nimeltä MongoDB (*About Us - Our Story* 2021).

MongoDB rakentuu muiden dokumentti-orientoituneiden tietokantojen tapaan kokoelmista, joita voidaan jollain tapaa verrata relaatiotietokantojen tauluihin. Kokoelmat pitävät sisällään BSON (Binary JSON) muotoisia dokumentteja, jotka voivat rakenteeltaan poiketa toisistaan. Dokumenttien tietokentät voivat koostua hyvin heterogeenisistä tietotyypeistä kuten esimerkiksi merkkjonoista, taulukoista, numeroista tai dokumenteista (Jose ja Abraham 2017).

4.1.1 MongoDB:n skaalaaminen

MongoDB hyödyntää skaalaamisessa sekä isäntä-orja toisintamista että tietokannan sirpalointia. Kahden eri skaalaustavan käyttö parantaa sekä kirjoitus- että lukunopeutta (Gu ym. 2015).

Isäntä-orja toisintaminen toteutetaan edellä esitellyn tavan mukaisesti. Tietokannasta luodaan yksi isäntäsolmu sekä n kappaletta orjasolmuja. Isäntäsolmu käsittelee tietokannan kirjoituspyynnöt ja tallentaa pyydetyt muutokset isäntäsolmun toisintoon. Orjasolmut huolehtivat toisintojensa tietojen ajantasaisuudesta synkronoimalle muuttuneet tiedot isäntäsolmun

toisinnosta asynkronisesti.

Tietokannan sirpalointi MongoDB:ssä toteutetaan kokoelma kohtaisesti ja sirpalointi perustuu yhteen tai useampaan sirpaleavaimeen. Yhden sirpaleen maksimikoko on oletusarvoisesti 200 megatavua. Jos uutta tietoa lisättäessä sirpaleen koko ylittää tuon raja-arvon, järjestelmä jakaa kyseisen sirpaleen tiedon kahteen osaan ja muodostaa niistä uudet sirpaleet.

Tieto sirpalepalvelimista ja niiden sisältämistä sirpaleista ylläpidetään kokoonpanopalvelimella. Sirpaleiden osalta tieto pitää sisällään muun muassa sirpaloidun kokoelman nimen sekä sirpaleavainten ensimmäisen ja viimeisen arvon. MongoDB järjestelmään tuleva tietokantapyyntö ohjautuu ensin kokoonpanopalvelimelle, joka määrittää pyynnön sirpaleavaimen perusteella millä sirpaleella käsiteltävä tieto sijaitsee. Pyyntö ohjautuu tämän jälkeen määritellylle sirpaleelle, joka itsenäisenä instanssina suorittaa pyynnön ja palauttaa tiedon MongoDB järjestelmälle. Järjestelmä koostaa mahdollisesti eri sirpaleilta tulevat tiedot yhdeksi vastaukseksi. MongoDB toimittaa koostetun vastauksen tietokantapyynnön lähettäneelle asiakkaalle ja tämän jälkeen tarvittaessa päivittää sirpaletiedot kokoonpanopalvelimelle.

4.2 Couchbase Server

Couchbase Serverin historiaa alkaa vuonna 2009, jolloin NorthScale niminen yritys käynnisti projektin uuden tietokannan kehittämiseksi. Projektin aikaansaannoksena syntyi Membase nimellä tunnettu NoSQL tietokanta, joka pohjautui Memcached tietokantaan (Tiwari 2011). Membase liittyi yhteen CouchOne nimisen yrityksen kanssa vuonna 2011 muodostaen Couchbase yhteisyrityksen (Rao 2011).

Couchbase Server on tietokantatyypiltään sekä dokumentti-orientoitunut että avain-arvo -tietokanta. Tieto tallennetaan JSON muotoisena (Amghar, Cherdal ja Mouline 2018) ja Memcached:sta periytyvä sisäänrakennettu välimuisti tekee tiedonhausta nopeaa (Băzăr ym. 2014).

Couchbase Serverissä tietokantoja kutsutaan bucketeiksi, joita on olemassa kolmea eri tyyppiä. Couchbase bucket voi olla käytössä sekä muistinvaraisena että levylle tallennettuna. Toistuvasti käyttössä olevat bucketit säilytetään muistissa mistä ne voidaan poistaa tarvittaessa jos käytössä oleva muistikapasiteetti ylitetään. Tällöin bucket säilyy levylle tallennet-

tuna josta se voidaan jälleen tarvittaessa palauttaa muistinvaraiseksi (*Buckets* 2021).

Ephemeral bucketit eli väliaikaiset bucketit säilytetään ainoastaan muistinvaraisina. Tämä mahdollistaa todella nopean tiedonkäsittelyn. Muistikapasiteetin täyttyessä väliaikaiset bucket kuitenkin joko jäädytetään niin, että niihin ei voi lisätä uutta tietoa tai ne poistetaan muistista. Koska väliaikainen bucket on tyypiltään ainoastaan muistinvaraista, tietoja ei tallenneta lainkaan levylle eikä sitä kautta muistista poistettuja tietoja voida enää palauttaa.

Kolmas bucket tyyppi on Memcached bucket ja sitä säilytetään väliaikaisen bucketin tapaan ainoastaan muistinvaraisena. Tämä bucket tyyppi on kuitenkin uusimissa Couchbase Server versioissa vanhentunut ja korvautunut väliaikaisilla bucketeilla.

4.2.1 Couchbase Serverin skaalaaminen

Couchbase Server on perusarkkitehtuuriltaan hajautettu tietojärjestelmä. Useita Couchbase Server instansseja eli solmuja voidaan yhdistää yhdeksi klusteriksi, jonka jokaista solmua voidaan hallita sen klusterimanagerin kautta.

Couchbase Server solmulla on klusterimanagerin lisäksi käytössä palveluita (services), jotka hoitava muun muassa tiedon tallennuksen, indeksoinnin, tietokantakyselyiden käsittelyn sekä hakujen käsittelyn. Näitä palveluja voidaan joko ajaa yhtäaikaisesti kaikissa klusterin solmuissa tai vaihtoehtoisesti osa solmuista voidaan dedikoida esimerkiksi ainoastaan tiedon tallentamiseen kun samalla jokin muu solmu hoitaa pelkästään tietokantakyselyiden käsittelyn (*Couchbase Server Overview* 2021).

Couchbase Serverin tietokannat eli bucketit ovat toteutettu vBucketeilla joita jokaisella jokaisella bucketilla on 1024 kappaletta. Poikkeuksena kuitenkin macOS käyttöjärjestelmä jossa vBucketeja on käytössä 64. vBucketit on hajautettu tasaisesti klusterissa oleville tiedon tallennusta hoitaville solmuille. Bucketin sisältö taas on hajautettu tasaisesti kaikille vBucketeille. Käytännössä tämä hajauttaminen siis toteuttaa sirpaloinnin periaatetta ja vBucketeja usein verrataankin sirpaleisiin (*vBuckets* 2021).

5 Pohdinta

Perusluonteeltaan kaikki NoSQL tietokantatyypit ovat horisontaalisesti skaalautuvia ratkaisuja. Tämä selittääkin niiden käytön yleistymisen. Osa NoSQL tietokannoista kuten Amazonin Dynamo sekä Googlen BigTable ovat kehitetty erityisesti suurten tietomassojen käsittelyyn. Tietomäärien kasvaessa skaalautuvuuden merkitys kasvaa entisestään.

Mahdollisuus sekoittaa horisontaalisen skaalaamisen eri tapoja antaa työkaluja tehostaa sekä tietokannan luku- että kirjoitusnopeutta. Painotuksen muuttaminen isäntä-orja toisintamisen ja sirpaloinnin välillä takaa tietokantojen käyttökelpoisuuden eri käyttöympäristöissä.

Skaalautuvuuden toteutustavat poikkeavat hieman yllättäen merkittävästi samaan NoSQL tietokantatyyppiin kuuluvien tietokantojen välillä. Vaikka MongoDB ja Couchbase Server ovat molemmat dokumentti-orientoituneita tietokantoja, niiden skaalaaminen tapahtuu hyvin eritavalla.

Siinä missä MongoDB noudattaa tiukasti isäntä-orja toisintamisen sekä sirpaloinnin periaatteita, Couchbase Server perustaa skaalautuvuutensa huomattavasti sofistikoituneempaan toimintatapaan. Vaikka Couchbase Serverin ratkaisusta voidaan löytää niin sirpaloinnin kuin toisintamisenkin ajatusmalleja, on se kuitenkin rakennettu arkkitehtuuriltaan kokonaisvaltaisemmin skaalautuvuuden näkökulma silmälläpitäen.

MongoDB:n tyypilliset skaalaustavat tekevät tietokannasta hyvin helposti lähestyttävän ratkaisun. Tietokannan käyttöönotto on nopeaa ja sen käsittelyyn on saatavilla helppokäyttöisiä ohjelmakirjastoja yleisimmille ohjelmointikielille. MongoDB:n skaalaustavat mahdollistavat tietokannan skaalaamisen kohtuullisen vaivattomasti tarpeiden kasvaessa. Kaiken kaikkiaan MongoDB jättää mielikuvan helposti lähestyttävästä tietokannasta, joka mahdollistaa skaalaamisen järkevästi tiettyyn rajaan saakka.

Siinä missä MongoDB:n skaalautumista voisi kuvailla ajatuksella 'pienestä suuremmaksi', Couchbase Serverin osalta se voisi olla 'suuresta vielä suuremmaksi'. Koko arkkitehtuuri suorastaan huokuu skaalautuvuutta. Couchbase Serverin klusterointiin ja yksittäisten Couchbase Server -instanssien heterogeeniseen palvelurakenteisiin perustuva kokonaisuus

antaa käyttöön todella voimakkaat työkalut skaalata järjestelmää monipuolisesti. Oikeastaan voisi todeta, että Couchbase Serverin käyttö yksittäisenä instanssina paljastaa vain pienen osan tietokannan mahdollisuuksista.

Tietokannan valinta on tärkeä osa sovelluskehitystä. Valintaan vaikuttaa useita tekijöitä, joista tietokannan skaalautuvuus on yksi olennainen. Pelkästään tieto siitä, että tietokannan ominaisuuksiin kuuluu skaalautuvuus ei kuitenkaan riitä valintaa tehtäessä. Olennaista on hahmottaa miten laajaksi kehitettävän sovelluksen oletetaan kasvavan. Jos sovelluksen tietomäärän ja käyttäjäkunnan arvioidaan kasvavan maltillisesti, MongoDB on erinomainen vaihtoehto skaalauksen näkökulmasta. Jos kuitenkin on odotettavissa, että tarve tietokannan suorituskyvyn parantamiseen on jatkuvaa ja alati kasvavaa, tarkastelluista tietokannoista Couchbase Server on selkeästi parempi vaihtoehto.

Lähteet

Ali, Ahmed Hussein. 2019. “A survey on vertical and horizontal scaling platforms for big data analytics”. *International Journal of Integrated Engineering* 11 (6): 138–150.

Amghar, Souad, Safae Cherdal ja Salma Mouline. 2018. “Which NoSQL database for IoT applications?” Teoksessa *2018 international conference on selected topics in mobile and wireless networking (mownet),* 131–137. IEEE.

Băzăr, C., ym. 2014. “The transition from rdbms to nosql. a comparative analysis of three popular non-relational solutions: Cassandra, mongodb and couchbase”. *Database Systems Journal* 5 (2): 49–59.

Berg, K.L., T. Seymour ja R. Goel. 2012. “History Of Databases”. *International Journal of Management Information Systems (IJMIS)* 17 (1): 29–36.

Buckets. 2021. Accessed: 2021-04-08, helmikuu. https://docs.couchbase.com/server/current/learn/buckets-memory-and-storage/buckets.html.

Couchbase Server Overview. 2021. Accessed: 2021-04-08, helmikuu. https://docs.couchbase.com/server/current/learn/architecture-overview.html.

vBuckets. 2021. Accessed: 2021-04-08, helmikuu. https://docs.couchbase.com/server/current/learn/buckets-memory-and-storage/vbuckets.html.

Curator, C. 2021. *What Are Object-Oriented Databases And Their Advantages.* Accessed: 2021-03-13. https://www.c-sharpcorner.com/article/what-are-object-oriented-databases-and-their-advantages2/.

DB-Engines. *DB-Engines Ranking.* https://db-engines.com/en/ranking. Accessed: 2021-02-28.

Gu, Yunhua, Xing Wang, Shu Shen, Sai Ji ja Jin Wang. 2015. “Analysis of data replication mechanism in NoSQL database MongoDB”. Teoksessa *2015 IEEE International Conference on Consumer Electronics-Taiwan,* 66–67. IEEE.

Jing Han, Haihong E, Guan Le ja Jian Du. 2011. “Survey on NoSQL database”. Teoksessa *2011 6th International Conference on Pervasive Computing and Applications,* 363–366. https://doi.org/10.1109/ICPCA.2011.6106531.

Jose, B., ja S. Abraham. 2017. “Exploring the merits of nosql: A study based on mongodb”. Teoksessa *2017 International Conference on Networks Advances in Computational Technologies (NetACT),* 266–271. https://doi.org/10.1109/NETACT.2017.8076778.

Karger, David, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins ja Yoav Yerushalmi. 1999. “Web caching with consistent hashing”. *Computer Networks* 31 (11-16): 1203–1213.

Kuznetsov, Sergey D, ja Andrey V Poskonin. 2014. “NoSQL data management systems”. *Programming and Computer Software* 40 (6): 323–332.

Lake, Peter, ja Paul Crowther. 2013. “Concise guide to databases”. *A History of Databases.*

Manoj, V. 2014. “Comparative study of nosql document, column store databases and evaluation of cassandra”. *International Journal of Database Management Systems* 6 (4): 11.

Scalability. 2021. Accessed: 2021-03-13, suomennos: Pertti Mäki-Välkkilä. https://www.merriam-webster.com/dictionary/scalability.

About Us - Our Story. 2021. Accessed: 2021-02-28. https://www.mongodb.com/company.

Moniruzzaman, A B M, ja Syed Akhter Hossain. 2013. *NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison.* arXiv: 1307.0191 [cs.DB].

Nayak, Ameya, Anil Poriya ja Dikshay Poojary. 2013. “Type of NOSQL databases and its comparison with relational databases”. *International Journal of Applied Information Systems* 5 (4): 16–19.

Oussous, Ahmed, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen ja Samir Belfkih. 2013. “Comparison and classification of nosql databases for big data”. *International Journal of Database Theory and Application* 6 (4.2013).

"Definition of NoSQL". 2021. Accessed: 2021-02-28. Viitattu 28. helmikuuta 2021. https://www.pcmag.com/encyclopedia/term/nosql.

Pokorny, Jaroslav. 2013. "NoSQL databases: a step to database scalability in web environment". *International Journal of Web Information Systems.*

Raatikainen, Kimmo, ja Pasi Porkka. 1998. "Database usage in telecommunications through CORBA". Teoksessa *IN'98. 7th IEEE Intelligent Network Workshop Proceedings (Cat. No. 98TH8364),* 291–301. IEEE.

Rao, Leena. 2011. *NoSQL Companies CouchOne And Membase Merge To Form Couchbase.* Accessed: 2021-02-28, helmikuu. https://techcrunch.com/2011/02/07/nosql-companies-couchone-and-membase-merge-to-form-couchbase/.

Saxena, V., ym. 2012. "Representation of Object-Oriented Database for the Development of Web Based Application Using Db4o". *Journal of Software Engineering and Applications* 5 (9): 687–694.

Tauro, Clarence JM, Shreeharsha Aravindh ja AB Shreeharsha. 2012. "Comparative study of the new generation, agile, scalable, high performance NOSQL databases". *International Journal of Computer Applications* 48 (20): 1–4.

How Much Data Is Created Every Day in 2021? 2021. Accessed: 2021-02-28. https://techjury.net/blog/how-much-data-is-created-every-day/.

Tiwari, Shashank. 2011. *Professional nosql.* John Wiley & Sons.

Weintraub, Grisha. 2014. "Dynamo and BigTable—Review and comparison". Teoksessa *2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI),* 1–5. IEEE.

NoSQL. 2021. Accessed: 2021-02-28. https://en.wikipedia.org/wiki/NoSQL.