Opinnäytetyö (AMK)

Tietojenkäsittely

2021

Jenni Heikkilä

Rakenteettoman datan tallennukseen soveltuvien tietokantaratkaisujen vertailu

– ARPA-projekti

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tietojenkäsittely

2021 | 34 sivua

Jenni Heikkilä

Rakenteettoman datan tallennukseen soveltuvien tietokantaratkaisujen vertailu

- ARPA-projekti

Turun ammattikorkeakoulun Applied Research Platform for Autonomous Systems -projektin tarkoituksena on luoda uusi kokeiluympäristö autonomisille kulkuneuvoille hyödyntäen muun muassa digitaalisia kaksosia. ARPA-projekti toimii case-studyna opinnäytetyön aiheelle, joka oli tietokantaratkaisujen vertailu. Projektiin oli tarve löytää tietokantaratkaisu väliaikaiseen video- ja sensoridatan tallennukseen. Lopullisen tietokantaratkaisun tulisi sopia myös projektin budjettiin ja käyttötarkoituksiin.

Avoimen lähdekoodin tietokantaratkaisuista vertailtavaksi valittiin kolme lupaavimman oloista vaihtoehtoa. Kaikki kolme asennettiin virtuaalikoneelle, jotta ne voitaisiin testata yksitellen ja verrata niiden soveltuvuutta käyttötarkoitukseen. Ensimmäinen vaihtoehto oli SeaweedFS, joka oli lupaavan oloinen, koska se oli kirjoitettu tehokkaalla ohjelmointikielellä. Odotuksista huolimatta se ei soveltunut käyttötarkoitukseen, sillä SeaweedFS menetti tehokkuutensa joutuessaan käsittelemään isomman määrän tietoa kerralla. Yhtenä vaihtoehtona oli Parse Server, joka oli mahdoton asentaa sen vanhentuneiden lisäosien takia. Viimeinen vaihtoehto oli Ambry, joka osoittautui parhaimmaksi. Se oli kaikin tavoin toimiva ja tehokas. Testeissä huomattiin, että mikäli Ambryn tietoväylä ruuhkautuisi liikaa, se loisi automaattisesti lisää tietoväyliä datamassan käsittelyyn.

Kaikkien testauksien, vertailujen ja tutkimusten jälkeen tutkimusryhmä päätyi Ambryyn. Sen tehokkuus ja helppokäyttöisyys täyttäisivät vaatimusmäärittelyn osoittamat rajat.

Asiasanat:

tietokannat, rakenteeton data, videotiedosto, suoratoisto, avoin lähdekoodi

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Business Information Technology

2021 | number of pages 34

Jenni Heikkilä

Comparison of database solutions for unstructured data

- ARPA-project

The objective of Applied Research Platform for Autonomous Systems project, run by Turku University of Applied Siences, was to create a testing environment for autonomous transportation using, among other technologies, digital twins. The subject of this thesis was the comparison of database solutions and the ARPA -project serves as a case study for the subject. The project needed a database solution for temporary storage for video and sensor data. The final solution for the database should also fit the budget and intended use of the project.

From the open source database solutions three options that seemed most promising were found. All three were installed into a virtual machine so they could be tested individually and compared for their suitability for the intended use. The first option was SeaweedFS which was promising since it was written in efficient programming language. Despite of all expectations it did not fit in the requirements as it could not properly cope with large sets of data. Another option was Parse Server which turned out to be impossible to install properly due to its outdated add-ons. The last option was Ambry which turned out to best meet the requirements. It was functional and effective in every aspect. During testing, it was observed that when there was a danger of Ambry’s data bus becoming bottlenecked, it automatically created more data buses for itself.

After extensive testing, comparisons and research, Ambry was selected because its efficiency and usability met the requirement specifications.

Keywords:

databases, unstructured data, video file, streaming, open source code

Sisältö

Kuvat

Sanasto

ARPA	Applied Research Plattform for Autonomous Systems (Turku AMK 2021)
Asiointibotti	Ohjelma, joka on suunniteltu keskustelemaan ihmisen kanssa useimmiten tekstin välityksellä
Data lake	Data lake eli datajärvi on keskitetty varasto, jonka avulla voidaan tallentaa kaikki rakenteellinen ja rakenteeton data missä tahansa mittakaavassa (AWS Azure 2021)
DBMS-järjestelmä	Database manage system eli tietokannan hallintajärjestelmä (Oracle 2021)
Go / Golang	Googlen kehittämä ja marraskuussa 2009 julkistettu ohjelmointikieli
IoT	Internet of Things eli esineiden internet, lyhyesti tarkoittaa kaikkien esineiden yhdistämistä nettiin
Localhost	Koneen oma verkko-osoite, jota voidaan käyttää joihinkin koneen omien toimintojen testaamiseen
OKM	Opetus- ja kulttuuriministeriö
SQL	Structured query language eli jäsennelty kyselykieli (IBM 2021)
TKI	Tutkimus-, kehitys- ja innovaatiotoiminta

1 Johdanto

Opinnäytetyön aiheena oli rakenteettoman datan tallennukseen soveltuvien tietokantaratkaisujen vertailu, joka suoritettiin toiminnallisena opinnäytetyönä ARPA-projektiin tapaustutkimuksena (case study) Turun Ammattikorkeakoululle alkukesästä 2021. Tavoitteena oli löytää toimivin ja kustannustehokkain tietokantaratkaisu kameroilta ja sensoreilta saapuvan datan hetkelliseen säilytykseen ja sen jatkokäsittelyyn. Ratkaisu oli tarpeellista löytää nopeasti, sillä dataa oli tarpeen pystyä ottamaan vastaan mahdollisesti jo heinäkuun aikana.

Tietokantaratkaisuja on paljon ja yleensä oikea vaihtehto löytyy nopeasti. Etsittäessä kustannustehokasta, avoimen lähdekoodin ratkaisua, joka pystyy käsittelemään suurtakin määrää rakenteetonta dataa, oikean löytämisestä tulee monimutkaisempaa. Opinnäytetyössä vastataan tutkimuskysymykseen, mikä avoimen lähdekoodin tietokantaratkaisu olisi soveltuva projektiin saapuvan datamäärän suuruuden ja sen jatkokäsittelyn huomioon ottaen. Yleisimmin käytetyistä avoimen lähdekoodin tietokantarataisuista on tehty listoja, mutta niistä ei suoraan selviä, mikä olisi sopiva tarvittavaan tilanteeseen. Aluksi valitsin pienen määrän tietokantaratkaisuja ja vertasin niitä toisiinsa samalla, kun testasin niiden kestävyyttä vaativassa käytössä.

Opinnäytetyön alussa käydään läpi, mitä tietokannat ovat, mitä eroa on rakenteellisella ja rakenteettomalla datalla, mitä tarkoittaa avoimen lähdekoodin tietokantaratkaisu ja minkälaisia tietokantaratkaisuja on. Tämän jälkeen käydään hieman läpi, miten eri tavoin näitä tietokantaratkaisuja voidaan testata, mitä työkaluja testaamiseen on ja millä tavoin tietokantoja pystytään rasitetestaamaan väliaikaisessa ympäristössä ennen lopullista käyttöönottoa. Tämän jälkeen kerrotaan, miten työ eteni vaihe vaiheelta alkaen vaihtoehtojen valinnasta ja niiden karsimisesta kolmeen vaihtoehtoon. Kerrotaan myös siitä, mitä tietokantaratkaisuista ilmeni jo asennusvaiheessa, mitä tapahtui yksinkertaisten testien aikana. Tämän jälkeen käydään läpi rasitetestaus

kahden viimeisen vaihtoehdon kohdalla, ja lopuksi kerrotaan, mihin päädyttiin ja miksi.

Johtopäätöksissä käydään lyhyesti läpi, mitkä olivat työn tavoitteet ja päästiinkö hyvään lopputulokseen. Lisäksi esitetään perusteet tutkimuksen luotettavuuteen ja oman ajatukset jatkokehitystä ajatellen.

2 Tietokannat ja data

Koska opinnäytetyön pääaiheena on sopivimman tietokantaratkaisun valinta käynnissä olevaan projektiin, on aiheellista käydä läpi perusasioita tietokannoista, niihin liittyvistä datatyypeistä ja mahdollisista datalähteistä.

Tietokanta on järjestetty kokoelma jäsenneltyä dataa, joka on tyypillisesti tallennettu sähköisesti tietokonejärjestelmään. Tietokantaa ohjaa yleensä tietokannan hallintajärjestelmä eli DBMS-järjestelmä. Tieto, DBMS-järjestelmä ja niihin liittyvät sovellukset luovat kokonaisuuden, jota kutsutaan tietokantajärjestelmäksi eli tietokannaksi. (Oracle 2021.)

2.1 Erilaisia tietokantoja

On olemassa erilaisia tietokantoja, joista jokainen tarjoaa käyttäjilleen erilaisia toimintoja ja soveltuvat erilaiseen käyttöön ja dataan. Jokaisen käyttäjän tulisi valita tietokantaratkaisunsa sen mukaan, joka soveltuu heidän vaatimuksiinsa ja tarpeisiinsa. Esimerkiksi relaatiotietokanta ei tarjoa juurikaan skaalautuvuutta ja tietokannan skeemat ovat luonteeltaan tiukempia, mutta lisäävät johdonmukaisuutta ja rakennetta. Ei-relaatiotietokannat käyttävät useita muotoja, kuten asiakirjoja, kaavioita, leveitä sarakkeita, jotka tarjoavat joustavuutta ja skaalautuvuutta tietokantasuunnitteluun. (Fatima 2019.)

2.1.1 Relaatiotietokannat ja ei-relaatiotietokannat

Relaatiotietokannat ovat usein kokoelma taulukoita, jotka pitävät sisällään tietyn setin rakenteellista dataa. Taulukko sisältää kokoelman rivejä ja sarakkeita. Näissä taulukoissa rivejä usein kutsutaan myös luetteloksi ja sarakkeita attribuuteiksi. Jokainen sarake taulukossa on suunniteltu säilyttämään vain tietynlaista informaatiota, esimerkiksi nimiä, rahasummia, numeroita tai päivämääriä.Tietokannan sisällä on yksi tai useampi objektien omistusryhmä, joita kutsutaan skeemoiksi. Jokaisessa skeemassa on tietokantaobjekteja,

kuten taulukoita, näkymiä ja tallennettuja proseduureja. Jotkut objektit, kuten sertifikaatit ja epäsymmetriset avaimet, sisältyvät tietokantaan, mutta eivät sisälly skeemaan. Relaatiotietokannat eli SQL-tietokannat on suunniteltu rakenteellista dataa varten. Rakenteettoman datan käsittelyyn ne soveltuvat huonosti. (Microsoft 2021.)

Ei-relaatiotietokannat eli NoSQL-tietokannat eivät ole taulukkomuotoisia tietokantoja. Ne tallentavat tiedot eri tavalla, kuin relaatiotietokantojen taulukot. NoSQL-tietokantoja on useita erilaisia tietomallinsa perusteella. Päätyypit ovat asiakirja, avainarvo, laaja sarake ja kaavio. Ne tarjoavat joustavia skeemoja ja skaalautuvat helposti suurilla tietomäärillä ja käyttäjäkuormilla. (MongoDB 2021.)

Esimerkkinä eroista voidaan käyttää yksinkertaisen kirjatietokannan skeeman mallintamista. Relaatiotietokannassa kirjatietue usein normalisoidaan ja tallennetaan erillisiin taulukoihin ja suhteet määritellään ensisijaisen ja vieraan avaimen rajoituksilla. Tässä esimerkissä Kirjat-taulukossa on sarakkeet ISBN:lle, kirjan nimelle ja painoksen numerolle. Tekijät-taulukossa on sarakkeet AuthorID:lle ja Author Nameille ja lopuksi Tekijä-ISBN-taulukossa on sarakkeet AuthorID- ja ISBN-numeroille. Relaatiomalli on suunniteltu mahdollistamaan tietokanta pakottamaan viite-eheys tietokannan taulukoiden välillä, normalisoitu redundanssin vähentämiseksi ja yleensä optimoitu tallennusta varten. NoSQL-tietokannassa kirjatietue tallennetaan yleensä JSON-dokumenttina. Jokaisen kirjan nimike, ISBN, kirjan nimi, painoksen numero, tekijän nimi ja tekijän tunnus tallennetaan attribuutteina yhteen asiakirjaan. Tässä mallissa tiedot on optimoitu intuitiivista kehitystä ja horisontaalista skaalautuvuutta varten.

2.1.2 BLOB

BLOB eli binaarinen suuri objekti on kokoelma binääritietoja, jotka on tallennettu yhtenä kokonaisuutena. Nämä ovat tyypillisesti ääntä, kuvias tai muita multimediaobjekteja. Blobeja voi tallentaa myös esimerkiksi relaatiotietokantaan, mutta niiden käsittely saattaa olla hankalaa muualla, kuin

niille soveltuvilla tietokantatyypeillä. Sovellussuunnittelijoiden on päätettävä tallennetaanko blobit tiedostojärjestelmään eli niin kutsuttuun varastoon vai tietokantaan, jolloin vaihtoehdot ovat rajatumpia. (Sears, Van Ingen and Gray 2007.)

2.2 Rakenteellinen ja rakenteeton data

Data on liiketoimintaa koskevien päätösten perusta. Yrityksen kyky kerätä oikeaa dataa, tulkita sitä ja toimia näiden oivallusten perusteella on usein se, joka ratkaisee yrityksen menestyksen. Yritysten saatavilla olevan tiedon määrä kasvaa jatkuvasti, mutta samalla kasvaa myös erilaisen saatavilla olevan datan määrä. Yritystietoja on monenlaisissa muodoissa tiukasti muodostetuista relaatiotietokannoista jonkun viimeiseen Twitter-viestiin. Kaikki tämä data eri muodoissaan voidaan jakaa kahteen pääluokkaan. Kyseessä olevat pääluokat ovat rakenteellinen ja rakenteeton data. (Smallcombe 2020.)

2.2.1 Rakenteellinen

Rakenteellinen data on erittäin järjestelmällistä ja koneoppimisen algoritmien kautta katsottuna helposti luettavaa. Tyypillisesti se kategorisoidaan kvantitatiivisena datana. Tällaisen datan käsittelyyn luotiin ohjelmointikieli SQL vuonna 1974, jonka avulla pystytään tänäkin päivänä tehokkaasti lisäämään, etsimään ja manipuloimaan rakenteellista dataa. (IBM 2021.)

Rakenteellista dataa ovat esimerkiksi yksittäiset henkilötiedot, kuten nimet, osoitteet ja henkilötunnukset, mutta myös muut yksittäiset, helposti järjestettävät tiedot kuten pankkitilinumerot, korttinumero tai osallistujamäärä. Rakenteellisen datan eduiksi voi laskea helppokäyttöisyyden ja saatavuuden, mutta datalla on alttius tietojen joustamattomuuteen. Rakenteellista dataa käytetään yleisimmin asiakashallintojärjestelmissä, jossa käytetään koneoppimista tunnistamaan asiakkaiden käytös- ja ostomalleja ja luomaan

personifikoituja tarjouksia, online-varauksissa esimerkiksi hotellien ja lentojen osalta sekä kirjanpidossa. (IBM 2021.)

2.2.2 Rakenteeton

Rakenteeton data tyypillisesti kategorisoidaan kvalitatiivisena datana, eikä sitä voida prosessoida tai analysoida yleisillä data työkaluilla ja metodeilla. Koska rakenteettomalla datalla ei ole ennalta määritettyä tietomallia, sen hallinta onnistuu parhaiten ei-relaatiotietokannoissa, niin kutsuttu NoSQL-tietokannoissa. Toinen tapa on säilyttää ne raakamuodossa datajärvien avulla. (IBM 2021.)

Rakenteeton data on tietoa, jota ei ole järjestetty ennalta määritetyn tietomallin tai skeeman mukaan. Tästä syystä sitä ei voida tallentaa perinteiseen relaatiotietokantaan. Teksti ja multimedia ovat kaksi yleistä jäsentämättömän sisällön tyyppiä. Monet yritysasiakirjat ovat jäsentämättömiä, kuten myös sähköpostiviestit, videot, valokuvat, verkkosivut ja äänitiedostot. (MongoDB 2021.)

Rakenteettoman datan tärkeys maailmalla on kasvanut ja kasvaa edelleen maailmalla nopeasti. Kaikesta yritysdatasta yli 80 % on rakenteetonta dataa ja yrityksistä 95 % priorisoi rakenteettoman datan hallintaa. Esimerkkeinä rakenteettomasta datasta ovat sosiaalisen median viestit ja postaukset, IoT sensoridata ja kameradata. Etuina rakenteettomassa datassa selkeästi ovat datan pitäminen alkuperäisessä formaatissaan, datan käsittelyn nopeudessa, kun sitä ei tarvitse esimäärittellä ja tallennustilassa esimerkiksi datajärvien avulla. (IBM 2021.)

Rakenteettomalla datalla on myös huonoja puolia. Määrittelemättömän luonteensa vuoksi datatieteen asiantuntemusta tarvitaan rakenteettoman datan valmisteluun ja analysointiin. Tämä on hyödyllistä data-analyytikoille, mutta vieraannuttaa erikoistumattomat yrityskäyttäjät, jotka eivät täysin ymmärrä spesialioituja data-aiheita tai kuinka hyödyntää omaa dataansa. Lisäksi tuotevaihtoehtoja rajoittaa huomattavasti, että rakenteeton data vaatii omat,

rakenteettoman datan manipulointiin ja käsittelyyn erikoistuneet datankäsittelytyökalut. (MongoDB 2021.)

Rakenteetonta dataa käytetään asiointiboteissa, joiden tarkoituksena on reitittää asiakkaat kysymysten perusteella oikeisiin vastauslähteisiin, datan louhinnassa, joka mahdollistaa yrityksille jäsentämättömien tietojen käyttämisen kuluttajien käyttäytymisen ja ostotottumusten tunnistamiseen, ja ennakoivaan data-analytiikkaan, joka ilmoittaa yrityksille etukäteen tärkeästä toiminnasta, jotta voivat suunnitella ja sopeutua merkittäviin markkinamuutoksiin. (IBM 2021.)

Binääridata

Joissain tapauksissa rakenteeton data voi olla myös binääridataa. Binääridata on datatyyppi, joka esitetään tai näytetään binäärilukujärjestelmässä. Binääridata on ainoa tietoluokka, jonka tietokone voi ymmärtää ja suorittaa suoraan. Se esitetään numeerisesti nollien ja ykkösten yhdistelmällä. (Technopedia 2021.)

Tämän tyyppistä dataa tuotetaan aina, kun prosessi suoritetaan tietokoneella. Prosessia pyytävä sovellus lähettää ohjeet korkean tason kielellä, joka lopulta muunnetaan binääridataksi suoritettaviksi tai lähetettäväksi prosessorille. Kaikki prosessit niiden tyypistä riippumatta muunnetaan binäärimuotoon ennen suorittamista. Tästä huolimatta binääridata ei hyödy relaatiotietokantojen tuomista työkaluista, joten se käsitellään rakenteettomana datana. (Technopedia 2021.)

Multimediadata

Opinnäytetyössä on käytetty kameroita ja sensoreita datalähteenä, joten on hyvä käydä läpi myös multimediadata. Multimediadatalla tarkoitetaan dataa, joka koostuu erilaisista mediatyypeistä, kuten tekstistä, äänestä, videosta ja animaatiosta. Erilaiset perustietotyypit: teksti, kuvat, ääni, video jne. ovat tyypillisesti elementtejä yleistettyjen multimediaympäristöjen, alustojen tai integrointityökalujen rakennuspalikoista. Nämä perustietotyypit sisältävät ASCII-pohjaista tekstidataa, joka on tyypillisesti tallennettu prosessoritiedostoihin,

laskentataulukoihin, tietokantoihin ja yleisempien multimediaobjektien huomautuksiin. (IGI Global 2009.)

GUI:iden, tekstifonttien, jotka mahdollistavat erikoistehosteet, kuten värit ja sävyt, saatavuuden ja lisääntymisen myötä tekstin tallentaminen on monimutkaista. Kuvien tai kuvatietojen osalta still-kuvien tallennustilan laadussa ja koossa on suuria eroja. Digitalisoidut kuvat ovat pikseleitä, jotka edustavat käyttäjän graafisen näytön aluetta. Still-kuvien yläpuolella oleva tila vaihtelee resoluution, koon, monimutkaisuuden ja kuvan tallentamiseen käytetyn pakkaustavan mukaan. Suosittuja kuvamuotoja ovat jpg, png, bmp, tiff. (IGI Global 2009.)

Yhä suositumpi useimpiin sovelluksiin integroitu tietotyyppi on ääni, koska se vaatii melko paljon tilaa. Yksi minuutti ääntä voi viedä jopa 2-3 Mbs tilaa. Sen pakkaamiseen sopivaan muotoon käytetään useita tekniikoita. (IGI Global 2009.)

Yksi eniten tilaa vievistä multimediatietotyypeistä on digitalisoitu video. Digitalisoidut videot tallennetaan kehysjaksoina. Resoluutiosta ja koosta riippuen yksi kehys voi kuluttaa jopa 1 Mt. Myös realistisen videon toisto edellyttää digitalisoidun sisällön siirtoa, pakkausta ja purkamista. Nykyään 3D-elokuvien aikakaudella meillä on myös graafisia objekteja tietotyyppinä. Tämä koostuu erityisistä tietorakenteista, joita käytetään määrittelemään 2D- ja 3D-muotoja, joiden avulla voimme määritellä multimediaobjekteja. Näitä ovat erilaiset kuva- ja videoeditointisovellusten käyttämät tiedostomuodot. Esimerkkejä ovat CAD/CAM-objektit. (IGI Global 2009.)

2.3 Sensorit ja kamerat datalähteenä

Sensoreilta ja kameroilta tuleva data on usein rakenteetonta dataa. Sensori on laite, joka lähettää dataa fyysisestä ympäristöstään havaitsemista asioista. Näitä tietoja usein käytetään antamaan informaatiota jollekin toiselle järjestelmälle tai ohjaamaan jotain prosessia. Sensoreilla pystytään havaitsemaan lähes kaikkia fyysisiä elementtejä ympäristöstä sensorimallista

riippuen. Esimerkiksi kiihtyvyysmittari havaitsee muutokset painovoiman kiihtyvyydessä siitä laitteesta, johon se on asennettu, kuten peliohjaimesta tai älykellosta. Toisena esimerkkinä lidar sen sijaan on laserpohjainen menetelmä tunnistukseen ja kartoitukseen, joka tyypillisesti käyttää pienitehoista ja silmille turvallista pulssilaseria, joka toimii yhteistyössä kameran kanssa. (Wigmore 2015.)

Langattomat anturiverkot yhdistävät erikoistuneita muuntimia viestintäinfrastruktuuriin olosuhteiden tarkkailua ja tallennusta varten eri paikoissa. Yleisesti valvottuja parametreja ovat lämpötila, kosteus, paine, tuulen suunta ja nopeus, valaistuksen voimakkuus, tärinän voimakkuus, äänen voimakkuus, sähkölinjan jännite, kemikaalien pitoisuudet, saastetasot ja kehon elintärkeät toiminnot. IoT-ympäristössä sensoridata on olennaisena osana, sillä kaikki data voidaan varustaa yksilöllisellä tunnisteella eli UID-tunnisteella ja siirtää verkon yli mihin tahansa. Suurin osa maailman lähetetystä datasta on sensoridataa. (Wigmore 2015.)

Erilaisten sensorien lisäksi myös kameroita käytetään datan keräämiseen. Kamerat ovat nykypäivänä enimmäkseen digitaalisia sekä erittäin korkearesoluutioisia. Perusvideokuvaamisen lisäksi monet kamerat kykenevät myös itsenäisesti analysoimaan ja välittämään tietoa siitä mitä videokuvassa tapahtuu. Verrataan esimerkiksi fyysisiä kauppoja verkkokauppoihin. Kameroilta saatavan datan ja analyysin avulla fyysisen kaupan kauppias pystyy keräämään konkreettista tietoa kaupan toiminnasta, kuten kuinka monta ihmistä vierailee heidän myymälöissään, mihin vuorokauden aikaan ja viikonpäivänä nämä vierailut tapahtuvat, missä vierailevat ihmiset liikkuvat heidän kaupoissaan ja millä osastolla he viettävät eniten aikaa. Tällaista dataa voidaan kerätä esimerkiksi käyttämällä polku- tai lämpökarttoja. (Cambridge Innovation Institute 2021.)

2.4 Tietokantojen testaus

Tietokanta on tärkeä osa sovelluksen kokonaisuutta, sen johdonmukaisuutta ja eheyttä tulee valvoa ja ylläpitää. Tietokannan testaaminen on tärkeää jo sovelluskehityksen aikana, jotta voidaan minimoida myöhempien ongelmien mahdollisuus. Tietokannan testaus tulisi suorittaa valvotussa ja hallitussa ympäristössä ja sen yhteydessä käydään läpi että kaikki toimii moitteetta ja virheitä ei tule. Testaukseen kuuluu muun muassa taulukoiden tai ohjelmarutiinien käynnistävien liipaisimien tarkistaminen. Samalla testausprosessi tarkistaa tietojen eheyden ja johdonmukaisuuden. Testauksessa luodaan monimutkaisia tietokantakyselyitä, joiden on tarkoitus kuormittaa ja stressitestata tietokantaa. Tämän tarkoituksena on testata tietokannan responsiivisuutta ja kykyä kestää erilaisia skenaarioita. (ReQtest 2020.)

2.4.1 Testaustyökalut

Tietokannan testaustyökaluilla tarkoitetaan ohjelmia, jotka on tarkoitettu suorittamaan aiemmin mainitut testausrutiinit tietokantoihin joko automaattisesti tai ohjattuna. Pääajatus on kuitenkin vähentää käsin tehtävää työtä isompia projekteja testattaessa. Useista testaustyökaluista jokaiseen tietokantatyyppiin ja erilaiseen tilanteeseen on löydettävissä sopiva työkalu. Esimerkkejä näistä on Db stress, Database Rider ja SeLite, mutta tässä opinnäytetyössä käytössä oli Siege.

Siege on avoimen lähdekoodin regressiotesti ja vertailuapuohjelma tietokantojen testaukseen. Siegella voi testata yksittäisen URL-osoitteen, jossa on käyttäjän määrittämä määrä simuloituja käyttäjiä tai se voi lukea muistiin useita URL-osoitteita ja testata niitä samanaikaisesti. Ohjelma raportoi tallennettujen osumien kokonaismäärän, siirretyt tavut, vasteajan, samanaikaisuuden ja paluutilan. Siege tukee HTTP/1.0- ja 1.1 -protokollia, GET- ja POST -direktiivejä, evästeitä, tapahtumakirjausta ja perustodennusta. Sen ominaisuudet ovat konfiguroitavissa käyttäjäkohtaisesti. (JoeDog 2021.)

Useimmat ominaisuudet ovat konfiguroitavissa komentorivivalinnoilla, jotka sisältävät myös oletusarvoja. Oletusarvot minimoivat ohjelman kutsumisen monimutkaisuuden. Siege antaa stressitestata verkkopalvelinta käyttäjän valitsemilla käyttäjämääräluvuilla ja latauskerroilla. Se tallentaa testin kokonaiskeston, sekä jokaisen yksittäisen tapahtuman keston. Siege raportoi myös tapahtumien määrän, kuluneen ajan, siirretyt tavut, vasteajan, tapahtumien nopeuden, samanaikaisuuden ja kuinka monta kertaa palvelin vastasi OK, eli tilakoodin 200. (JoeDog 2021.)

3 ARPA-projektin tietokantaratkaisun valinta

ARPA-projekti eli Applied Research Plattform for Autonomous Systems -projektin on tarkoitus vahvistaa, yhdistää ja parantaa jo olemassa olevia alueellisia yhteistöitä tehtaiden ja merenkäynnin alueilla virtuaalitodellisuussimulaatioiden, digitaalisten kaksosten ja tekoälyn avulla. Nämä muodostavat kasvavan alan, johon kohdistuu paljon odotuksia, jotka liittyvät paljon tuottavuuteen, ekologisuuteen ja turvallisuuteen. Tavoitteena on mahdollistaa etänä hallittavien, automaattisten ja autonomisten järjestelmien pilotointi ja testaus, käyttäen hyväkseen tietokonelaskentaa, erilaisia datalähteitä, kameroita ja sensoreita. (Turku AMK 2021.)

ARPA-hanke on Turun ammattikorkeakoulun ja Yrkeshögskolan Novian yhteistyö ja projekti lähti virallisesti käyntiin marraskuussa 2020. Aikeena on kolmen vuoden aikana perustaa yhdessä alan yritysten kanssa Turkuun kokeiluympäristö autonomisille kulkuneuvoille, jossa monipuolinen tutkimusympäristö tarjoaa mahdollisuuden yrityksille ja tutkijoille testata tuotteitaan sekä simuloiduissa virtuaaliympäristöissä että fyysisessä testausympäristöissä. Projekti sai OKM hakemusten perusteella erityisrahoituksena ammattikorkeakouluille myöntämää valtionrahoitusta tutkimus-, kehitys- ja innovaatiotoimintaan. (Turku AMK 2021.)

Testeistä ja piloteista kerätyn datan taltiointi ja soveltava tutkimus on mahdollista toteuttaa ilman kilpailuasetelmaa alan yritysten kanssa, kun kyseessä on korkekouluvetoinen hanke. Tarjoamalla käsiteltyä ja visualisoitua tutkimusdataa kaikkien käyttöön hyödyttää systemaattisesti kerätty, analysoitu ja tallennettu data alan tutkimusta ja kehitystä. (Turku AMK 2021.)

Turun ammattikorkean verkkosivulla ARPA-projektista kirjoitetaan: ”Uuden ARPA-tutkimusympäristön luomiseksi asiantuntemuksensa yhdistävät Yrkeshögskolan Novian Aboa Mare -koulutuskeskus ja Turun ammattikorkeakoulun langattoman tietoliikenteen, kyberturvallisuuden ja konetekniikan laboratoriot sekä vuorovaikutteisten teknologioiden osaamiskeskus FIT Turku.”

Hankkeella oli tarve toimivalle ja tehokkaalle tietokantaratkaisulle kesän 2021 alkuun, joten opinnäytetyön soveltava osuus alkoi toukokuussa 2021. Kameroilta ja sensoreilta tuleva jatkuva datavirta tuli saada tallennettua väliaikaisesti tietokantaan. Kamera- ja sensoridatalle tultaisiin tekemään esikäsittelyä ennen siirtoa lopulliseen tallennuspaikkaansa. Tästä syystä datan tulisi olla helposti saavutettavissa ja käsiteltävissä myös väliaikaisessa tallennuspaikassaan.

3.1 Tietokantaan kohdistuvat vaatimukset

Koska projektin tapauksessa oli kyse kameroiden ja sensoreiden tuottamasta datasta eli aikaisemmin opinnäytetyössä esitellystä rakenteettomasta datasta, tarvittiin tietokantaratkaisu, joka tuki juuri kyseistä rakenteetonta dataa. Tästä syystä tutkimusryhmä sulki pois aikaisemmin teoriaosuudessa esitellyt relaatio- ja ei-relaatiotietokannat. Kameroiden ja sensoreiden oli tarkoitus tuottaa dataa suoratoistona, joten rakenteettoman datan määrä kasvaa nopeasti suureksi. Tästä syystä tietokantaratkaisun vaatimusmäärittelyyn lisättiin, että tietokantaratkaisun tuli kestää hidastumatta ja kaatumatta suurikin datamäärä ja -liikenne. Vaikka dataa oli paljon ja sitä tulisi jatkuvasti lisää, tuli yksittäiset datapaketit olla tehokkaasti ja nopeasti löydettävissä datamassan keskeltä.

Esimerkiksi, jos haluttiin käsittelyyn kameralta tuleva videodata maanantailta lokakuun 4. päivä, sen piti olla löydettävissä isostakin määrästä videodataa. Tästä johtuen ”data lake” vaihtoehdot suljettiin alusta asti pois. Data lake on tallennusvarasto, joka säilyttää kyllä valtavan määrän raakadataa alkuperäisessä muodossaan, kunnes sitä tarvitaan analytiikkasovelluksiin. Perinteinen tietovarasto tallentaa tiedot hierarkkisissa mitoissa ja taulukoissa, kun taas datajärvi käyttää tasaista arkkitehtuuria tietojen tallentamiseen, ensisijaisesti tiedostoihin tai objektien tallennustilaan. Tästä johtuen yhden tietyn datapaketin löytäminen nopeasti tai edes helposti saattoi olla hankalaa.

Tietokantaratkaisun tuli olla myös edullinen. Kun kyseessä oli väliaikainen datavarasto ei ollut tarpeen tutkimusryhmän mukaan käyttää hankkeen budjettia

tietokantaratkaisuun. Koska datamäärä kasvaisi nopeasti suureksi tulisi tietokantaratkaisun ylläpidosta nopeasti kallista ja koska tietokantaratkaisu ei ollut datan viimeinen tallennuspaikka tuli ajatella budjetin kannalta loogisesti. Tästä syystä yleisesti käytetyt ja kenties jopa tehokkaammat vaihtoehdot jäivät pois, kuten Azure BLOB storage.

Azure Blob storage on Microsoftin pilveen tarkoitettu objektitallennusratkaisu. Blob storage eli Blob varasto on optimoitu valtavien määrien rakenteettoman datan tallentamiseen. (Sears, Van Ingen and Gray 2007.)

Projektilla oli tarve tietokantaratkaisulle, jonka vaatimusmäärittely sisälsi kaksi kokonaisuutta vaatimuksista. Tietokantaratkaisun tuli olla avoimen lähdekoodin ratkaisu budjetillisista syistä. Sen tulisi toimia rakenteettoman datan kanssa ilman että tietokannasta tietyn datapaketin etsiminen olisi hidasta tai hankalaa.

3.2 Avoimen lähdekoodin tietokantaratkaisujen vertailu ja valinta

Avoimen lähdekoodin tietokantaratkaisuja on useita ja niiden löytäminen itsessään on kohtuullisen vaivatonta. Projektin asettama vaatimusmäärittely kuitenkin laittoi tarkat rajat hyväksyttävälle tietokantaratkaisulle. Koska tavoitteena oli löytää tietokantaratkaisu, joka soveltui ison datamäärän käsittelyyn ja tallentamiseen ilman, että tietokanta alkaisi jäämään jälkeen datan tallentamisessa, osoittautui tarpeeseen sopivan tietokannan löytäminen kuitenkin yllättävän hankalaksi ja aikaavieväksi.

Vaatimusmäärittelyn rajoituksiin sopivien avoimen lähdekoodin tietokantaratkaisuvaihtoehdoista löytyi kolme vaihtoehtoa. Tietokantaratkaisut Parse Server, SeaweedFS ja Ambry vaikuttivat monin tavoin samankaltaisilta ja niiden dokumentaation pohjalta oli syytä uskoa, että ne olisivat helposti käyttöönotettavia ja soveltuisivat suurten datamäärien käsittelyyn. Valittaessa testiin otettavia tietokantaratkaisuja kiinnitettiin huomiota erityisesti mahdolliseen tehokkuuteen ja kuinka yleisiä kyseiset tietokantaratkaisut ovat käytössä sekä yksityishenkilöiden projekteissa että isommissa hankkeissa.

3.2.1 Parse Server

Parse Server on avoimen lähdekoodin mobile Backend As a Service (mBAAS) -alusta. Se voidaan ottaa käyttöön missä tahansa infrastruktuurissa, joka pystyy suorittamaan Node.js:ää ja Parse Server toimii Express-verkkosovelluskehyksen kanssa. Vuodesta 2016 Parse Serverin on ollut avoimen lähdekoodin tietokantaratkaisu. Parse Server on myös laajennettavissa webhookeilla, töillä ja konfiguraatioilla. Lisäksi siinä on useita yhteisön tekemiä laajennuksia. (Parse Server 2020.)

Parse Serverin dokumentaation mukaan tietojen tallennus on helppoa. Kaikkien tietojen perusyksikkö on objektien API. Esimerkiksi, jos määritetään ajoneuvoluokan avainten valmistajan ja mallin avulla, CRUD REST -toiminto voidaan suorittaa yksinkertaisella curl-pyynnöllä. CRUD on lyhenne sanoista Create, Read, Update ja Delete, jotka ovat tietokantojen perustoimintoja. Curl-pyynnöstä saadaan vastauksena objektin ”objectId” ja ”created_date”. Muita toimintoja objektille voidaan suorittaa käyttämällä ”objectId”. (Parse Server 2020.)

Jos luokkaa ei ole ennalta määritetty, palvelin luo sen itse. Parse Server tukee merkkijonoa, numeroa, loogista arvoa, taulukoita, JSON-objekteja, päivämäärä-aikaa, tiedostoa ja null-arvoa. Lisäksi jäsennyksessä on kaksi mukautettua tietotyyppiä, osoitin toiseen jäsennysobjektiin ja suhde toiseen jäsennysluokkaan. Parse Serverin tietotyypit on lukittu. Kun tietotyyppi on asetettu, se palauttaa virheilmoituksen, jos yritetään tallentaa jotain muuta. (Parse Server 2020.)

3.2.2 SeaweedFS

SeaweedFS koostuu kolmesta pääkomponentista. Master Service ja Volume Service tarjoavat yhdessä hajautetun objektivaraston, jossa on käyttäjän määritettävä replikointi ja redundanssi. Valinnainen arkistointi ja S3-palvelu ovat

lisätasoja objektivaraston päällä. Jokainen näistä palveluista voi toimia yhtenä tai erillisenä esiintymänä useilla todellisilla palvelimilla. (Cottlehuber 2020.)

Master Service edustaa yhden tai useamman palvelimen klusteria, joka omistaa yhtenäisen näkymän koko SeaweedFS-klusteriin. Palvelinklusterien määrä on aina oltava pariton, jotta enemmistön konsesus voidaan toteuttaa. Palvelinklusterien määrä pyritään myös pitämään pienenä. Hyvänä lukumääränä pidetään yhtä tai kolmea. Master Servicen tehtävänä on kommunikoida kaikille osallistuville nodeille Raft-protokollan kautta valitun johtajan kautta. (Cottlehuber 2020.)

SeaweedFS:n Volume Servicet lähettävät sykettä valitulle johtajalle, joka päätää sykkeiden perusteella mihin liikenne reititetään ja miten replikointi käsitellään. Jos johtava ei ole tavoitettavissa, Raft-protokollan enemmistön konsesus varmistaa, että uusi johtaja nimitetään koko klusterin suostumuksella ja olemassa oleva tavoittamaton johtaja alennetaan, kunnes se pystyy taas toimimaan oikein. (Cottlehuber 2020.)

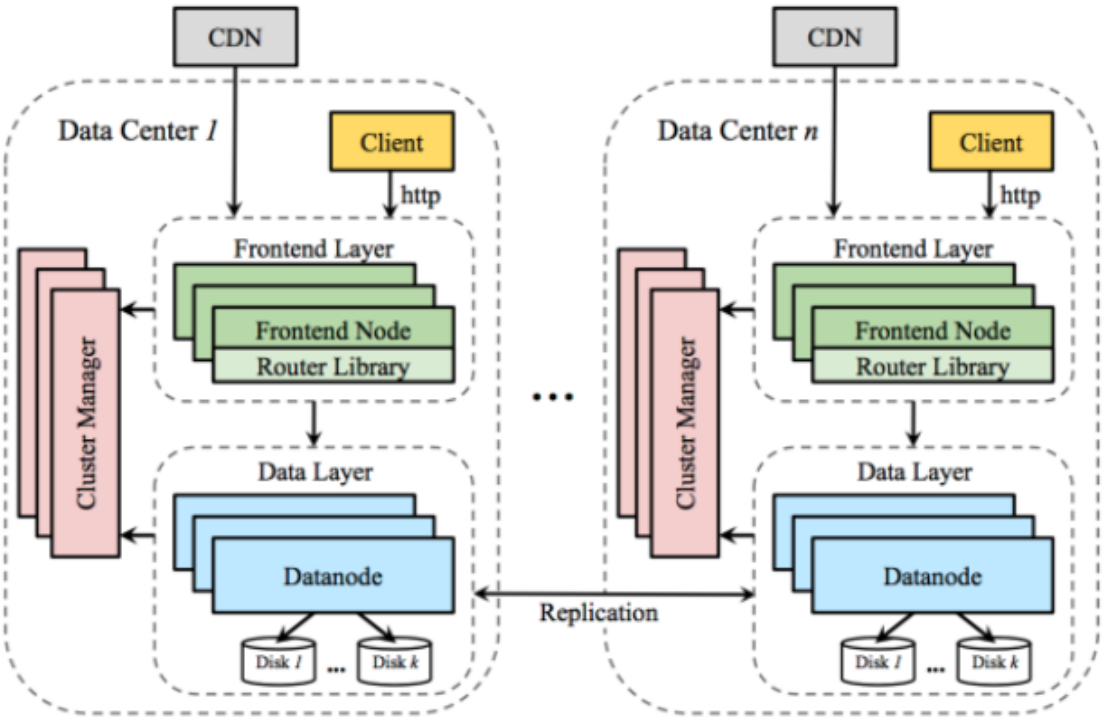
3.2.3 Ambry

Ambry on LinkedInin käyttämä hajautettu object storage, joka tukee biljoonien pienten muuttumattomien objektien sekä miljardien suurten objektien tallennusta. Se on erityisesti suunniteltu tallentamaan ja palvelemaan mediaobjekteja verkkoyrityksissä. Sitä voidaan kuitenkin käyttää yleiskäyttöisenä tallennusjärjestelmänä DB-varmuuskopioiden, hakuindeksien tai liiketoimintaraporttien tallentamiseen. Järjestelmän luvataan olevan erittäin saatavilla, vaakasuunnassa skaalautuva. Sillä luvataan myös olevan alhainen latenssi ja korkea suorituskyky. Ambry on optimoitu sekä pienille että suurille objekteille. Dokumentaatio lupaa sen myös olevan helppokäyttöinen ja kustannustehokas. (Xia 2016.)

Ambry koostuu joukosta data nodeja, jotka vastaavat tietojen tallentamisesta ja noutamisesta, etupään koneista, jotka reitittävät pyynnöt jonkin esikäsittelyn jälkeen data nodeihin, ja klusterinhallinnasta, joka koordinoi ja ylläpitää

klusteria. Data nodet replikoivat tietoja keskenään ja tukevat datakeskusten replikointia. Käyttöliittymä on vuorovaikutuksessa etäpalvelinkeskuksen data nodejen kanssa, kun vaaditaan ”read-after-write”-yhdenmukaisuutta. Käyttöliittymä tarjoaa HTTP-sovellusliittymän POST-, GET- ja DELETE-objekteille. Vaihtoehtoisesti voidaan käyttää suoraan käyttöliittymän käyttämää reititinkirjastoa suorituskyvyn parantamiseksi. LinkedInissä nämä käyttöliittymä nodet toimivat CDN:n alkuperäpalvelimina. (Xia 2016.)

Kuvassa 1 kuvataan Ambryn tapaa replikoida datakeskuksia, mikäli ohjelma sattuisi ruuhkautumaan käytössä. Jokainen datakeskus on toisensa kopio, jotta kaikki toimivat saumattomasti yhteen ja tietojen tallennuspisteet pysyvät samana. Ainut asia, joka datakeskusten välillä on erilaista on datakeskuksen oma numero, jota ainoastaan Ambry itse käyttää pitämään kirjaa datakeskusten määrästä. (Xia 2016.)

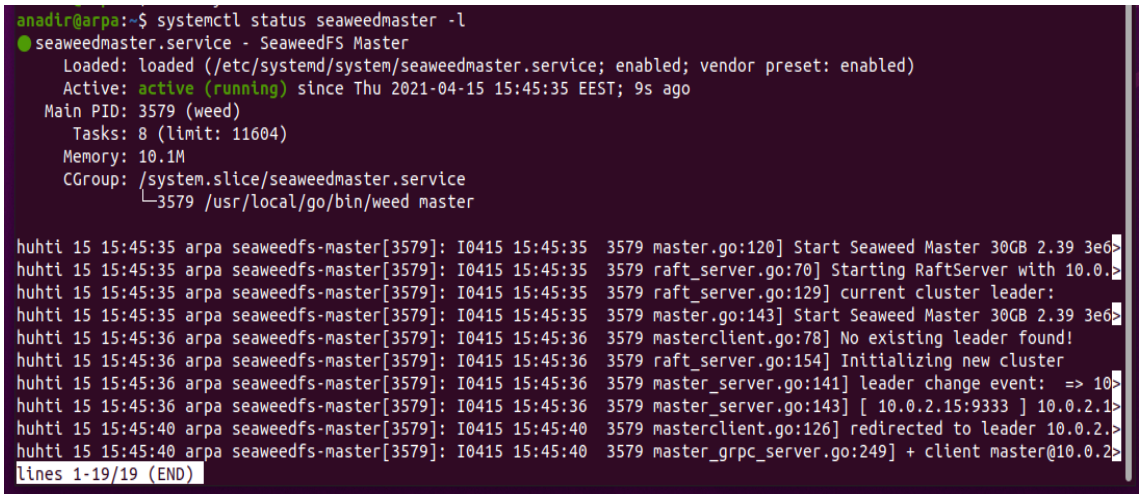


Kuva 1. Ambryn toimintamalli design

3.3 Avoimen lähdekoodin tietokannan asennus ja testaus

Kun tietokantaratkaisuvaihtoehdot oli valittu, valitut tietokantaratkaisut asennettiin virtuaalikoneelle. Asennuksissa ei SeaweedFS:n ja Ambryn kohdalla tullut vastaan ongelmia, mutta ensimmäisenä olleen Parse Serverin kohdalla tuli virheviesti. Virheviestin alkuperää selvitellessä tuli huomattua, että kyseinen tietokantaratkaisu ei voinut toimia alunperinkään. Se vaati toimiakseen kirjaston osan, jota ei oltu päivitetty pitkään aikaan, eikä vaadittu kirjaston osa edes ollut enää ladattavissa. Ongelmaa yritettiin kiertää monin tavoin, mutta ratkaisua löytämättä oli todettava, että Parse Serveriä ei voitu ottaa testaukseen. Ensimmäinen tietokantaratkaisuvaihtoehto jäi jo tässä vaiheessa pois lopullisesta vertailusta.

SeaweedFS:n dokumentaatiosta tarkemmin tutkiessa löytyi huomautus siitä, että mikäli tiedostot olisivat liian suuria tietokantaratkaisulle, se saattaisi kaatua. Tästä syystä SeaweedFS:lle tulisi pätkiä datapaketit 10Mb kokoon, jotta voitaisiin varmistua sen kestävyydestä suuremman datamassan alla. Kuvassa 2 esitetään onnistunutta SeaweedFS:n asennusta ja Master Servicen käynnistystä systemctl:n avulla.



Kuva 2 SeaweedFS:n käynnissä oleva Master Service

Jo aikaisemmin tässä opinnäytetyössä mainittuna Ambryn dokumentaatio lupasi suuren skaalautuvuuden ja tietokantaratkaisulta vahvaa kestokykyä suurempiinkin datamäärään ja datapakettien kokoihin. Tietokantaratkaisu asentui virtuaalikoneelle dokumentaation ohjeita seuratessa helposti ja vaivattomasti. Testatessa onnistunutta asennusta, kuvassa 3 esitetään komento, jonka avulla käynnistetään Ambryn pääserveri, joka käsittelee liikennettä taustalla tietokantaa käytettäessä.

```
$ nohup java -Dlog4j.configuration=file:../config/log4j.properties -jar ambry.jar --serverPropsFilePath
../config/server.properties --hardwareLayoutFilePath ../config/HardwareLayout.json --partitionLayoutFilePath
../config/PartitionLayout.json > logs/server.log &
```

Kuva 3 Ambryn pääserverin käynnistyskomento

Kuvassa 4 käynnistetään Ambryn käyttöliittymä, jonka kautta pystyy antamaan tietokannalle komentoja. Käyttöliittymä toimii oletuksena suoraan terminaalista. Kun käyttöliittymään kirjataan komento, se antaa suoraan vastauksen terminaaliin. Vastaus kuvastaa onko annettu komento tai toiminto onnistunut.

```
$ nohup java -Dlog4j.configuration=file:../config/log4j.properties -cp "*"
com.github.ambry.frontend.AmbryFrontendMain --serverPropsFilePath ../config/frontend.properties --
hardwareLayoutFilePath ../config/HardwareLayout.json --partitionLayoutFilePath ../config/PartitionLayout.json >
logs/frontend.log &
```

Kuva 4 Ambryn käyttöliittymän käynnistyskomento

Kuvassa 5 on käyttöliittymälle annettu niin kutsuttu healthCheck-komento. Komennon tarkoitus on lähettää sydämenlyönti sekä käyttöliittymälle, pääserverille että mahdollisille data nodeille tarkistaakseen niiden vastauskyvyn. Mikäli kaikki vastaavat vastaus on kuvassa näkyvä ”GOOD”. Jos jokin edellä mainituista ei vastaa sydämenlyöntiin komentoriville tulee ilmoitus virheestä ja mahdollisuuksien mukaan myös virheen sijainti.

```
$ curl http://localhost:1174/healthCheck
GOOD
```

Kuva 5 Onnistuneesti käynnistetyn Ambryn testauskomento ja vastaus

3.3.1 Testausmenetelmät

Kun tietokantaratkaisuvaihtoehdot oli valittu, siirryttiin testaukseen. Alkuperäisenä ajatuksena oli testata tietokantaratkaisuvaihtoehtoja mahdollisimman samalla tavoin, jotta testauksesta saatu data olisi verrannollista keskenään. Näin olisi helppo verrata tietokantaratkaisuja toisiinsa ja saada selville parhaiten projektiin sopiva tietokantaratkaisu.

Asennuksen jälkeen tietokantaratkaisuista käydään läpi niiden testausesimerkit. Jokaisella tietokantaratkaisulla tulisi dokumentaatiosta löytyä esimerkki tai useampi siitä, miten tietokantaratkaisua käytetään. Opinnäytetyössä oli käsittelyssä tässä kohtaa enää kaksi tietokantaratkaisuvaihtoehtoa ja sekä Ambrysta että Seaweedistä löytyi testausesimerkit.

Dokumentaatiosta löytyneiden esimerkkien jälkeen siirryttäisiin raskaampaan testaukseen, johon käytettäisi tietokannantestaustyökalua avuksi. Projektissa testaustyökaluksi vakiintui aikaisemmin tässä opinnäytetyössä esitelty Siege niminen ohjelma. Siegen avulla testattaisiin tietokantaratkaisun sietokykyä suurempien datapakettien ja runsaamman datamäärän kanssa.

3.3.2 Testaus

Asennuksen jälkeen tuli tietokantaratkaisut testata virtuaalikoneessa, johon ne oli asennettu. Testaus oli suunniteltu kulkemaan molempien vaihtoehtojen kanssa samaa linjaa, jotta tulokset olisivat mahdollisimman verrannoliset keskenään.

SeaweedFS testattiin sen omilla esimerkeillä, joita oli kaksi. Ensimmäinen esimerkki (Kuva 6) oli Seaweedin pääkäyttökansion asettaminen localhost-osoitteeseen. Tällä annetaan Seaweedille lupa käyttää asetettua kansiota tiedostojen ja master servicen hallintaan.

```
anadir@arpa:~$ curl http://localhost:9333/dir/assign
{"fid":"4,01d889bd9d","url":"10.0.2.15:8081","publicUrl":"10.0.2.15:8081","count":1}
```

Kuva 6 SeaweedFS:n pääkäyttökansion asettaminen localhost-osoitteeseen

Kuvassa 7 on näkyvissä komento, jolla katsotaan kuvan 6 komennon avulla saadun verkko-osoitteen tila. Tämä on volumen service. Kuvassa 7 näkyvä volumeId saadaan kuvan 6 avulla selville katsomalla count osaa. Tämä komento kertoo volume service 1 tilan ja lokaation.

```
anadir@arpa:~$ curl http://10.0.2.15:9333/dir/lookup?volumeId=1
{"volumeId":"1","locations":[{"url":"10.0.2.15:8080","publicUrl":"10.0.2.15:8080"}]}
```

Kuva 7 Volume servicen tilatarkastuskomento

Kuvassa 8 näytetään taas alkuun Seaweedin assign-komento, jonka jälkeen tietokantaan ladataan kuvatiedosto. Kuvatiedosto ladataan tietokannasta erillään olevasta kansiosta tietokantaan curl-komennolla. Komentoon annetaan ensin tiedoston osoite ja sen jälkeen tietokannan osoite kokonaisena. Tämän jälkeen kuvassa on vielä kolmas komento. Tämä komento on tarkoitettu jo olemassa olevien tiedostojen päivittämiseen. Kuvan 8 viimeisessä komennossa päivitetään toisessa komennossa tietokantaan ladattu kuva uuteen. Päivittäessä tiedoston osoite ei muutu vaikka koko, nimi tai muu metadata muuttuisi.

Kuva 8 Kuvatiedoston lataaminen ja päivittäminen SeaweedFS tietokantaan

Esimerkkien kautta tehdyissä testauksissa tietokantaratkaisu näytti tehokkaalta ja toimivalta. Tietokantaratkaisun nopeus vaikutti lupaavalta, joten siirryttiin esimerkeistä raskaampaan testaukseen. Testaustyökalu Siege otettiin käyttöön ja se valjastettiin lataamaan tietokantaan useampi videotiedosto tietokantaan. SeaweedFS kesti rasitustestauksen pienillä luvuilla ja määrillä. Ensimmäiset kierrokset Siegellä olivat vain yksi videotiedosto kerrallaan ja kymmenen kertaa. Kun luvut nostettiin 100 kappaleeseen useampi tiedosto kerrallaan Seaweedin teho ei riittänyt suorittamaan pyydettyä testauskomentoa. Seaweedin sietokyky ylittyi noin 40 kappaleen kohdalla ja ohjelma kaatui. Tässä kohtaa testausta myös virtuaalikoneen terminaali kaatui, joka aiheutti koko virtuaalikoneen kaatumisen.

SeaweedFS on ollut useissa projekteissa käytössä ja sen käytöstä ja käyttöönotosta löytyy konkreettista tietoa internetistä, joskin kevyemmistä käyttötarkoituksista. Eniten Seaweed näytti olleen käytössä projekteissa, joissa tarvitsi tallentaa paljon pieniä datapaketteja nopeasti. Huolimatta siitä, että Seaweediä ei suositeltu suurikokoisten tiedostojen käsittelyyn, sitä päätettiin kuitenkin testata rasitustestaustyökaluilla. Se ei kuitenkaan yltänyt projektin vaatimusmäärittelyn antamiin rajoihin.

Seaweedin jälkeen testaukseen otettiin Ambry. Ambry on käytössä useissa dataraskaissa palveluissa, joista kenties tunnetuin on LinkedIn. Ambry on alun perin kehitetty LinkedIniä varten ja on rakennettu kestämään suurta määrää rakenteetonta dataa eri lähteistä ladattuna ja uudelleen katsottavaksi suurelle yleisölle. Tästä johtuen sen oletettiin kestävän vähintään yhtä hyvin raskasta käyttöä kuin SeaweedFS. Ambrya verrattaessa Seaweediin huomataan, että se

ei ole ollut yhtä kovassa lataussuosiossa, mutta sitä kokeilleet olivat olleet erittäin tyytyväisiä kyseiseen tietokantaratkaisuun.

Ambry asentui nopeasti ohjeiden mukaan. Tämä mahdollisti nopean pääsyn testaukseen. Kuten aikaisemmin opinnäytetyössä mainittiin, testaus toistettiin samalla tavoin, kuin Seaweedin kohdalla. Näin Ambryn ja Seaweedin eroista saataisiin mahdollisimman hyvä kuva toisiinsa verrattuna. Tästä syystä testaus aloitettiin Ambryn dokumentaatiosta löytyneestä esimerkistä. Esimerkit toimivat yhtä hyvin kuin Seaweedin kohdalla. Niiden varaan ei kuitenkaan testausta voinut jättää, sillä esimerkit on luotu niin, että tietokantaratkaisu näyttäisi parhaintaan.

Seuraava vaihe testauksessa oli videotiedoston yksittäistallennukset sekä Ambryn omilla komennoilla että Siege ohjelmalla (Kuva 9). Kun Ambryyn latasi tiedoston tietokantaan riippumatta siitä käyttikö Siegeä vai Ambryn komentoja, Ambry tarjoaa suoraan terminaalissa linkkiä tallennettuun tiedostoon pelkän tiedoston uniikin tunnisteen sijaan. Tämä ratkaisisi suoraan ongelman tiettyjen datapakettien löytymisen tietokannasta, mikäli ohjelma tekisi niin jokaiselle tallentamalleen tiedostolle.

```
anadir@arpa:~$ siege --concurrent=5 --reps=1 --verbose --header="x-ambry-service-id:CUrlUpload" --header="x-ambry-owner
-id:`whoami`" --header="x-ambry-content-type:video/mp4" --header="x-ambry-um-description:Demonstration Image" --content
-type="video/mp4" --get 'http://localhost:1174 POST < SoT2021.mp4'
IS ASCII: TRUE
POST / HTTP/1.0
Host: localhost:1174
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/4.0.4
x-ambry-service-id:CUrlUpload
x-ambry-owner-id:anadir
x-ambry-content-type:video/mp4
x-ambry-um-description:Demonstration Image
Connection: close
Content-type: video/mp4
Content-length: 0



HTTP/1.1 201 Created
connection: close
x-ambry-non-compliance-warning: TTL < 2592000 will be required for future uploads
x-ambry-blob-size: 0
Location: /AAYQAf__AAAAAQAAAAAAAAAAtROCc9I3SSCbrs4xvuoogw
Date: Tue, 01 Jun 2021 10:43:39 GMT
Content-Length: 0
x-ambry-creation-time: Tue, 01 Jun 2021 10:43:39 GMT
x-ambry-request-cost: WRITE_CAPACITY_UNIT=1.0; STORAGE_IN_GB=0.0
x-ambry-datacenter: Datacenter
x-ambry-frontend: localhost


Transactions:                       1 hits
Availability:                  100.00 %
Elapsed time:                    0.02 secs
Data transferred:                0.00 MB
Response time:                   0.02 secs
Transaction rate:               50.00 trans/sec
Throughput:                      0.00 MB/sec
Concurrency:                     1.00
Successful transactions:            1
Failed transactions:                0
Longest transaction:             0.02
Shortest transaction:            0.02
```

Kuva 9 Videotiedoston lataus Siegellä Ambryyn

Siegen luvut nostettiin samoihin kuin mitä Seaweedille oli asetettu. 100 kappaletta videotiedostoja ladattiin tietokannalle samanaikaisesti, jotta pystyttiin testaamaan tietokantaratkaisun rasituksensietokykyä. Rasitustestauksessa kävi nopeasti ilmi, että Ambry oli kevyt, mutta äärimmäisen tehokas. Koska tietokantaratkaisu ei hidastunut testissä, eikä sen teho näyttänyt vähenevän isomman datamassan alla nostettiin videotiedostojen määrä 1 000 kappaleeseen samanaikaisesti (Kuva 10 ja Kuva 11). Tämäkään ei kaatanut ohjelmaa ja kaikki tiedostot löytyivät helposti Ambryn tarjoamilla osoitteilla. Kuvissa 10 ja 11 näkyvät tavumäärät näyttävät nollaa, koska Siege ei osaa laskea videotiedostojen kokoja oikein.

```
anadir@arpa:~$ siege --concurrent=5 --reps=1000 --verbose --header="x-ambry-service-id:CUrlUpload" --header="x-ambry-owner-id:`whoami`" --header="x-ambry-content-type:video/mp4" --header="x-ambry-um-description:Demonstration Image" --content-type="video/mp4" 'http://localhost:1174 POST < SoT2021.mp4'
IS ASCII: TRUE
** SIEGE 4.0.4
** Preparing 5 concurrent users for battle.
The server is now under siege...
HTTP/1.1 201     0.06 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.08 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.14 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.10 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.37 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.02 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.02 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.53 secs:       0 bytes ==> POST http://localhost:1174
```

Kuva 10 Ambryn rasitustestauskomento Siegellä

```
HTTP/1.1 201     0.00 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.01 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.01 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.00 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.00 secs:       0 bytes ==> POST http://localhost:1174
HTTP/1.1 201     0.01 secs:       0 bytes ==> POST http://localhost:1174

Transactions:                   5000 hits
Availability:                 100.00 %
Elapsed time:                  27.01 secs
Data transferred:               0.00 MB
Response time:                  0.03 secs
Transaction rate:             185.12 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    4.98
Successful transactions:        5000
Failed transactions:               0
Longest transaction:            0.55
Shortest transaction:           0.00
```

Kuva 11 Siegen ajon tulos

3.3.3 Tulokset

Kahden tietokantaratkaisuvaihtoehdon vertailu oli helppoa projektin kannalta. SeaweedFS ei täyttäisi vaatimusmäärittelyn asettamia rajoja, eikä tietokanta kestäisi niin raskaassa käytössä, kuin mihin tarkoitukseen sitä oltiin kaavailtu. Seaweeding kyky käsitellä suuria datapaketteja oli rajallinen ja sen sietokyky suurissa datamäärissä ei kestäisi jatkuvana suoratoistona tulevaa sensori- ja videodataa.

Ambry vaikutti sopivan vaatimusmäärittelyihin ja vaikka se oli ammattitaitoisesti ohjelmoitu tietokantaratkaisu, se oli avoimen lähdekoodin tietokantaratkaisu. Ambry kesti testauksissa paremmin kuin SeaweedFS, eikä se menettänyt tehoaan raskaammassakaan rasitustestauksessa.

Projektiin valittiin tietokantaratkaisuksi Ambry. Valinnan jälkeen Ambryn dokumentaatio tuli käydä uudestaan läpi ja kirjoittaa siitä tarpeellisimmat huomiokohdat ylös. Näitä listattuja huomioita tultaisiin myöhemmin käyttämään projektin omassa Ambry dokumentaatiossa, jossa oli myös ohjeet, kuinka tietokantaratkaisu asennettaisiin juuri ARPA-hankkeelle käyttöön. Tämän lisäksi virtuaalikoneesta puhdistettiin muut asennetut tietokantaratkaisut, tässä tapauksessa vain SeaweedFS, ja jaettiin se tutkimusryhmälle mahdollisia jatkotestauksia varten.

4 Johtopäätökset

Opinnäytetyössä oli tavoitteena löytää paras mahdollinen ratkaisu ARPA-projektin väliaikaiseen säilytykseen soveltuva tietokantaratkaisu. Kyseisen tietokantaratkaisun tuli myös pystyä suodattamaan tehokkaasti läpi suuriakin datapaketteja ja suurta määrää datamassaa. Tietokantaratkaisun tuli olla budetillisista syistä myös avoimen lähdekoodin ratkaisu.

Opinnäytetyön alussa käytiin tarkasti läpi ja kirjattiin ylös vaatimusmäärittely ja sen ohjeistamana etsittiin muutama tietokantaratkaisuvaihtoehto testattaviksi. Lopulliseen vertailuun pääsi kolme vaihtoehtoa, joista Parse Server jäi pois jo asennusvaiheessa. Sen toimintakuntoon asentamiseen olisi ollut tarpeen asentaa ohjelmistokirjaston osa, joka oli vanhentunut ja jäänyt päivittämättä tekijöiltä jo vuosia sitten, eikä enää ollut ladattavissa tai saatavilla. Vertailu jäi näin ollen kahden vaihtoehdon välille.

Testauksista tehtiin mahdollisimman identtiset SeaweedFS:n ja Ambryn välillä, jotta ne olisivat toisiinsa helposti verrattavissa. Molemmat tietokantaratkaisuvaihtoehdot suorittivat omista dokumentaatioistaan löytyneet esimerkkitestaukset kevyesti. Esimerkkien pohjalta tehtyjen testausten päämääränä oli katsoa, että tietokantaratkaisut ovat asentuneet oikein ja vastaavat komentoihin oletetulla tavalla.

Rasitustestaukseen siirryttäessä huomattiin nopeasti, että SeaweedFS ei kestänyt suurempaa datamassaa ja tietokantaratkaisuvaihtoehto jouduttiin pudottamaan nopeasti tämän jälkeen. Ambrya testatessa tutkimusryhmä huomasi nopeasti, että tietokantaratkaisu suoritti testaukseen laitetut datamäärät tehokkaasti. Ylimääräisillä lisätestauksilla huomattiin sen kestävän myös suuremmat datamäärät ja ohjelman toiminta ei hidastunut laisinkaan. Rasitustestauksien jälkeen tutkimusryhmä päätyi Ambryyn, sillä se täytti vaatimukset. Lisäksi se tarjosi ylimääräisiä hyödyllisiä lisätoimintoja, jotka sopivat projektin toimintaan.

Hyvän alkutyön ja rasitustestauksien tekeminen tietokantaratkaisulle kannatti, ja projektille saatiin tehokas ja toimiva ratkaisu video- ja sensoridatan väliaikaiseen säilömiseen ja sen käsittelyyn. Itse tietokannan integroiminen projektiin ei kuulunut työn alueeseen, joten tässä opinnäytetyössä käsiteltiin vain tietokantaratkaisun valinta, testaus ja kaiken tehdyn työn asennuksesta eteenpäin dokumentointi.

5 Lähteet

AWS Azure. *AWS.* 2021. https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/ (haettu 16. lokakuu 2021).

Cambridge Innovation Institute. *Internet of Business.* 2021. https://internetofbusiness.com/camera-become-iot-sensor/ (haettu 30. lokakuu 2021).

Colyer, Adrian. ”Ambry: LinkedIn’s scalable geo-distributed object store.” *the morning paper* UNCATEGORIZED, nro Ambry (2016): 1.

Cottlehuber, Dave. *SeaweedFS wiki.* 2020. https://github.com/chrislusf/seaweedfs/wiki/ (haettu 6. joulukuu 2021).

Fatima, Nida. *Astera - A Quick Overview of Different Types of Databases.* 2019. https://www.astera.com/type/blog/a-quick-overview-of-different-types-of-databases/ (haettu 2. joulukuu 2021).

Hu, Han, Yonggang Wen, Tat-Seng Chua, ja Xuelong Li. ”Toward Scalable Systems for Big Data Analytics: A Technology Tutorial.” *IEEE Access* 2, nro Big Data (2014): 652-687.

IBM. *IBM Blog - Structured vs. Unstructured Data: What's the Difference?* 2021. https://www.ibm.com/cloud/blog/structured-vs-unstructured-data (haettu 23. lokakuu 2021).

IGI Global. *IGI Global - What is Multimedia Data.* 2009. https://www.igi-global.com/dictionary/towards-multimedia-digital-libraries/19578 (haettu 2. joulukuu 2021).

JetBrains. *Distributed File Systems / JetBrains Company Blog.* 15. Lokakuu 2020. https://habr.com/en/company/JetBrains/blog/523630/ (haettu 1. syyskuu 2021).

JoeDog. *GitHub - JoeDog/Siege.* 2021. https://github.com/JoeDog/siege (haettu 10. marraskuu 2021).

Microsoft. *Database - SQL quide.* 2021. https://docs.microsoft.com/fi-fi/sql/relational-databases/databases/databases?view=sql-server-ver15 (haettu 2. joulukuu 2021).

MongoDB. *MongoDB - NoSQL.* 2021. https://www.mongodb.com/nosql-explained (haettu 2. joulukuu 2021).

—. *MongoDB - Unstructured Data.* 2021. https://www.mongodb.com/unstructured-data (haettu 2. joulukuu 2021).

Noghabi, Shadi A., ym. ”Ambry: LinkedIn's Scalable Geo-Distributed Object Store.” *SIGMOD '16: Proceedings of the 2016 International Conference on Management of Data* 2016, nro June (2016): 253-265.

Oracle. *Oracle OCI.* 2021. https://www.oracle.com/database/what-is-database/ (haettu 30. lokakuu 2021).

Parse Server. *Parse Server.* 2020. https://parseplatform.org/ (haettu 6. joulukuu 2021).

ReQtest. *ReQtest - Database Testing.* 2020. https://reqtest.com/testing-blog/database-testing/ (haettu 30. lokakuu 2021).

Sears, Russel, Catharine Van Ingen, ja Jim Gray. ”To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?” *Computer Science* 2006, nro April (2007): 1-11.

Smallcombe, Mark. *Xplenty - Big Data.* 2020. https://www.xplenty.com/blog/structured-vs-unstructured-data-key-differences/ (haettu 2. joulukuu 2021).

Stackshare. *Stackshare.* 2016. https://stackshare.io/stackups/ambry-vs-azure-storage (haettu 1. syyskuu 2021).

Technopedia. *Technopedia: Binary Data.* 2021. https://www.techopedia.com/definition/17929/binary-data (haettu 2. joulukuu 2021).

Turku AMK. *TURKU AMK - Projektit.* 2021. https://www.turkuamk.fi/fi/tutkimus-kehitys-ja-innovaatiot/hae-projekteja/applied-research-platform-autonomous-systems-arpa/ (haettu 16. lokakuu 2021).

Wigmore, Ivy. *TechTarget - IoT Agenda.* 2015. https://internetofthingsagenda.techtarget.com/definition/sensor-data (haettu 30. lokakuu 2021).

Xia, Ming. *Ambry wiki.* 2016. https://github.com/linkedin/ambry/wiki (haettu 6. joulukuu 2021).