Kalle Sirkesalo

Hadoop

Tietojenkäsittelyn
Tradenomi

Syksy 2015

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tekijä(t): Sirkesalo Kalle

Työn nimi: Hadoop

Tutkintonimike: Tietojenkäsittelyn Tradenomi

Asiasanat: Hadoop, Big Data, Hive, HBase, NoSQL, SQL

Tämän opinnäytetyön tavoitteena oli tarjota kattava kokonaisuus siitä miltä Hadoop näyttää ja mitä sillä voidaan tehdä. Työssä pyritään esittelemään kattavasti erilaisia teknologioita, jotka mahdollistavat erinäisiä toimintoja mitä entiset ratkaisut eivät kyenneet.

Työ käsittelee erinäisiä tietokanta ratkaisuja ja näiden toimintaan tietokantoina. Miten näitä on kehitetty ja mitä puutteita on vielä löydettävissä. Aiheessa pyrittiin kuvaamaan sitä millaisesta ekosysteemistä on kyse, kun puhutaan yritysmaailman Hadoop ratkaisuista, sekä kilpailusta Hadoop tarjonnassa.

Pääasiana opinnäytetyössä itselleni oli tutkia erinäisiä tuotteita, joita markkinoilta löytyy ja näiden tuomia ratkaisuja. Kuvaavana tästä työssä käsiteltiin aina levyjaosta tiedon metatiedon hallintaan.

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Author(s): Sirkesalo Kalle

Title of the Publication: Hadoop

Degree Title: Bachelor of Business and Information technology

Keywords: Hadoop, Big Data, Hive, HBase NoSQL, SQL

This thesis had a goal to give comprehensive understanding of Hadoop and its technologies and how it's used. This work tries to show large scale of different technologies. There are multiple different database solutions and ways to work with these solutions.

During this I also tried to bring across how the technology might affect us in future. One of the sub goals is to give understanding to how organizations use Hadoop for business. Thesis also tries to introduce the different vendors that are currently out in Hadoop world. Overall the goal is to give you understanding of what someone is talking about when they talk about Hadoop.

Topics goes from network shared disk to data's metadata. Introducing such topics as NoSQL in Hadoop and how to bring log data into Hadoop.

SYMBOLISANAT

Agentti	Ohjelma, joka on tietokoneella ja ottaa komentoja muualta
ANSI	Amerikan standardisointi instituutti
API	Ohjelmisto rajapinta
Devops	Ohjelmistokehitys toiminta
ISO	Maailmanlaajuinen standardi organisaatio
Klusteri	Monen tietokoneen muodostama laskentayksikkö
NewSQL	SQL ja NoSQL yhdistelmä
NFS	Verkossa sijaitseva levypalvelin
NoSQL	Strukturoimaton tietokantarakenne
REST	Ohjelmistorajapinta tyyli
SQL	Strukturoitu tietokantarakenne

SISÄLLYS

1 JOHDANTO

Työntarkoituksena oli kuvata erinäisiä Hadoop tietokanta ja Big Data ympäristöjä asiakkaan tarpeiden mukaisesti. Kyseessä oli myös tarve kartoittaa nykypäivän ratkaisujen tilaa, sekä näiden julkaisua. Tärkeää oli myös tuoda esiin mahdollisuuksia ja erinäisten Big Data järjestelmien hyviä, sekä huonoja puolia. Työn tarkoitus oli mahdollistaa laajan käsitteistön ymmärrys ja tämän kanssa toiminta. Asiakkaalle tärkeitä asioita olivat mahdollisuus tuottaa auttava opas erinäisille organisaation jäsenille, joita asia kiinnostaa, mutta eivät ole vielä ihan varmoja miten asiaa lähestyttäisiin. Tavoitteena työllä oli pyrkiä kehittämään niin omaa, kuin asiakkaiden ymmärrystä tietojärjestelmistä ja näiden hyödyistä.

Työssä otetaan myös kantaa asioiden automatisoinnin ja hallitsemisen puolesta. Tietojärjestelmien kanssa nämä ovat yleisesti ottaen tärkeitä käsitteitä ja tulevat jatkossa kasvamaan sitä mukaa kuin projektit kehittyvät eteenpäin. Tavoitteena oli tuottaa yleispätevä opas uusien teknologioiden käyttöönottoon ja ymmärtämiseen yritysmaailmassa ja siinä mitä menetetään ja mitä saadaan kun näihin järjestelmiin siirrytään.

Työnantajana toimii 2008 vuodesta saakka Big Datan kanssa työtä tehnyt Eficode, jonka reitti Big Dataan lähti HP Idolin strukturoimattoman tiedon hallinnasta ja jatkui siitä Veikkauksen ja Supercellin Vertica asennuksiin. Hadoop kuitenkin on monelle suomalaiselle firmalle uusi asia, joten oli tarve selventää asiaa niin kehittäjille, kuin johdolle, joten sain tehtäväksi kirjoittaa ja kasata Eficodelle lyhyen selityksen Big Datasta Hadoop puolelta. Työssä pyritään esittelemään erinäisiä teknologioita ja näiden toimintoja, sekä yhdistämistä toimivaksi yritys Hadoopiksi.

Eficodelle on tärkeää, myös yhdistää ymmärrys Big Datasta, Devopsista ja ohjelmistokehityksestä, koska nämä ovat aloja joissa toimimme ja haluamme tehdä asioita fiksusti. Näin ollen pelkkä ympäristön ylös saaminen ei riitä vaan siitä tulee tehdä hallittava, skaalattava ja automatisoitu. Näin ollen työnkäytännön osuudessa tullaan pääsemään haasteisiin ja ratkaisuihin automatisoinnin saralla. Automatisointia katsottiin Eficoden suosittelemien automatisointi ohjelmien puolesta, jotka ovat Ansible ja Docker.

2 TIETOKANNAT

Tietokannat ovat aikojen saatossa kehittyneet tarpeen mukaisesti. Markkinoilla oli pitkään käytössä pelkästään SQL pohjaisia tietokantoja, mutta tämä muuttui NoSQL (ei pelkästään SQL) terminologian julkaisun myötä. NoSQL käsittelee useita eri ratkaisuja erinäisiin ongelmiin. Tämän jälkeen Hadoop ja suurien tietomassojen käsittely ja analysointi vaati uusia teknologioita, jota varten kehitettiin uusia teknologioita. (Lo Frank 2015)

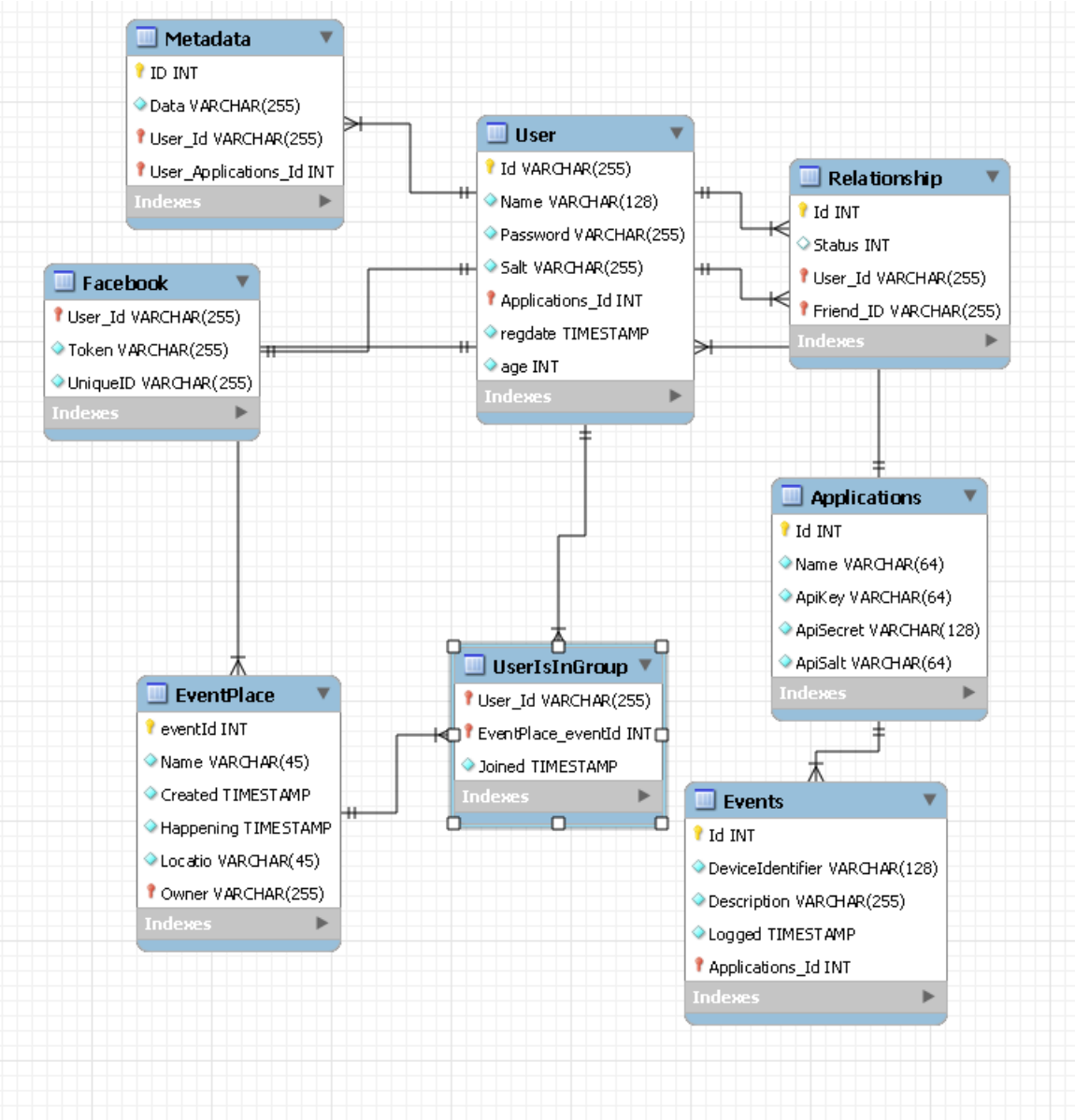
2.1 SQL

SQL eli strukturoitu kieli, perustuu relaatiomatematiikkaan, mutta tämä ei silti ole aivan suoranaisesti relaatiomatematiikkaa. Esimerkiksi terminologiassa käytetään rivi, kolumni ja taulu sanoja, jotta pystymme ymmärtämään mitä osaa relaatiosta käsitellään. SQL on määritelty ISO (maailmanlaajuinen standardi organisaatio) standardin, johon löytyy useita malleja, jotka nimetään vuosilukujen mukaan. (Christopher J. Date 2009)

SQL-kieli kehitettiin alkujaan IBM:n toimesta 1974 – 1975 vuosien aikana. Ensimmäinen kaupallinen versio julkaistiin 1979 Oraclen toimesta. SQL on historian suosituin kieli tietokantajärjestelmissä. Ensimmäisen standardinsa SQL sai ANSI:lta, eli Amerikan standardisointi instituutilta vuonna 1986. Kyseinen standardi määritteli pitkälle, sen miltä SQL nykyään näyttää. Nykyään suosituin ja käytetyin standardi on vuodelta 1992, joka on hieman laajempi kuin 1986 vuonna julkaistu standardi. (Vangie Beal 2015)

Oracle:n mielestä tietokanta on järjestelmä, joka mahdollistaa tiedon hakemisen. Yksinkertaisimmillaan SQL kantojen koostumuksen voisi kuvata riveinä, kolumneina ja tauluina. Tauluja yhdistetään relaatioilla, jotka perustuvat kolumneihin, joissa yhdistävä tekijä on yleensä tietty rivi. Relaatioita syntyy, kun kahdella rivillä on sama avain. Relaationaaliseksi tietokannaksi luokiteltavan kannan on pakko

pystyä yhdistämään toisiinsa kytkeytyvä tieto. Kuvassa1 esitellään pientä analyyttistä relaationaalista mallia. (Oracle 2014)



Kuva1. VSAA-serverin tietokantarakenne. (Sirkesalo & Kauppinen 2015)

2.2 NOSQL

NoSQL on huono termi, koska se käsittää laajan joukon eri tietokantoja, mikä tekee sen ymmärtämisestä vaikeaa. Kyseessä ovat siis tietokannat, jotka eivät perustu SQL ajattelumalliin. Datamäärien ja tapojen kasvu, sekä kerääminen loivat NoSQL kannat. Tämä vaati uuden lähestymisen tietojärjestelmien kehitykseen ja käyttöön. Yleisesti näille kannoille tietorakennetta ei ole määritetty ja järjestelmiä on helppo ajaa klustereina. Tämä tarkoittaa tiedon samana pysymisen ja varmuuden katoamista. Informaatio ei ole varmassa turvassa kantaan saavuttuaan vaan se voi vielä kadota. (Pramod J. Sadalage & Martin Fowler 2013)

NoSQL ei ole ratkaisu, joka korvaisi relaatiokannat vaan se pyrkii täydentämään tätä teknologiaa. Sen tavoite on tehdä asioita, joita relaatiokannat eivät tällä hetkellä pysty tekemään. Ratkaisun tarkoitus on enemmänkin tuoda ymmärrystä ja kykyä ratkaista ongelmia joita relaatiokannoilla ei pystytä ratkaisemaan. (Pramod J. Sadalage & Martin Fowler 2013)

NoSQL suosio nähdään usein kahden asian yhtymänä. Ensinnäkin NoSQL nähdään parantavan ohjelmistotuottamista. Tämä syntyy siitä, että NoSQL ei vaadi tarkkaa rakennetta missään vaiheessa voimme kehittää, sekä lisätä asioita hyvinkin vapaasti. Tämä vähentää tarvittavaa työmäärää uuden funktion lisäyksessä. Toiseksi syyksi useasti nähdään tiedonmäärän kasvu, joka syntyy kun yritykset pyrkivät keräämään enemmän tietoa asiakkaista ja ohjelmistaan. (Pramod J. Sadalage & Martin Fowler 2013)

2.2.1 Dokumenttipohjaiset ratkaisut

Dokumenttipohjaisten tietokantojen ideana on korvata rivi konsepti muuttuvammalla mallilla. Tässä tapauksessa parhaaksi muuttuvaksi malliksi nousivat taulukkojen ja dokumenttien tapaiset rakenteet, koska nämä ovat yleisesti erittäin vapaa muotoisia ja helposti muunneltavissa. Dokumenttipohjainen rakenne mahdollistaa

vaativien rakenteiden muunnoksen yksinkertaisiksi struktuureiksi. Alkuperäisenä tavoitteena oli tehostaa ohjelmoijien aikaa. Näiden jo ohjelmoidessa olioperäisesti tällä saavutettiin huomattavia helpotuksia ohjelmoijan arkeen. (Kristina Chodorow & Michael Dirolf 2010)

Yleisesti dokumenttipohjaisissa ratkaisuissa pyritään toimimaan ilman selkeää rakennetta. Näin on paljon helpompaa käsitellä muuttuvia tiedostomuotoja ja tallentaa tietoa järjestelmään. Dokumenttipohjainen rakenne suunniteltiin alusta loppuun laajennettavaksi, joka teki usealle palvelimelle laajenemisesta helpompaa kuin aikaisemmissa järjestelmissä. (Kristina Chodorow & Michael Dirolf 2010)

2.2.2 Avainparipohjaiset ratkaisut

Avainpohjaisissa tietokannoissa tietoa haetaan avaimien perusteella. Tällä tavalla saadaan kasvatettua nopeutta niin kirjoittamisessa kuin lukemisessa, koska tiedossa tarvitsee olla vain yhden kolumnin arvot ja toiseen ei tarvitse koskea tietokannassa ollenkaan. Useimpien avainpohjaisten ja dokumenttipohjaisten tietokantojen ero on hyvin pieni. Useasti avainpohjaisiin ratkaisuihin on kuitenkin sisällytetty datan jonkinmoinen hakeminen avainsanoilla. Avainpohjaisia kantoja kehittävät monet tahot, mutta useat tuntevat esimerkiksi Riakin ja Rediksen tällaisina tuotteina. (Pramod J. Sadalage & Martin Fowler 2013)

2.2.3 Kolumnipohjaiset ratkaisut

Kolumnipohjaiset tietokannat syntyivät alkujaan Googlen BigTable teknologiasta julkaisun tieteellisen paperin yhteydessä. Nykyään BigTablea pystyy käyttämään Googlen palvelujen avulla, mutta aikanaan tämä vaikutti suuresti Cassandran ja HBasen kehitykseen. Kolumnipohjaiset tietokannat perustuvat avaimiin, jotka kertovat mikä rivi on kyseessä, jonka jälkeen kolumneille kerrotaan, mitä tietoa tahdotaan riviltä saada ulos. (Pramod J. Sadalage & Martin Fowler 2013)

Kolumnipohjaiset järjestelmät yleensä jaotellaan kahtia. Ensimmäinen versio perustuu riveihin ja näissä asiat kytketään rivien perusteella samaan tyyliin kuin avainpohjaisissa järjestelmissä. Toisessa ratkaisuissa taas pyritään hakemaan tieto kolumnien avulla. Tällöin rivi on yhdistelmä erinäisiä kolumneja. (Pramod J. Sadalage & Martin Fowler 2013)

Näissä kannoissa ei ole perinteiseen tapaan tauluja vaan, jokainen rivi voi lisätä tai vähentää kolumneja tarpeensa mukaisesti. Tämä vaikuttaa tiedon helpompaan syöttämiseen. Lisäksi näin saadaan aikaiseksi kehittäjäystävällisempi järjestelmä. Tämä myös siirtää kannan pois perinteisestä relaationaalisesta ajattelusta, vaikka yleisesti näissä käytetään SQL tyylisiä hakuja. (Pramod J. Sadalage & Martin Fowler 2013)

2.2.4 Graph-pohjaiset ratkaisut

Graph-pohjaiset ratkaisut eivät perustu datan määrään tai suuruuteen vaan sen yhtymiseen. Näissä pyritään kuvaamaan ja kytkemään asioita toisiinsa. Näin saadaan aikaiseksi enemmänkin verkonmuotoinen tietokanta kuin tavallinen tauluja, rivejä ja kolumneja sisältävä tietokanta. Kyseisillä tietokannoilla pyritään selittämään ja etsimään uusia yhtymiä erinäisten asioiden välillä. Näissä kannoissa tiedon yhdistely tehdään jo tiedon saapuessa sisään ja tämä säästää tehoa, kun kyseisiä linkkejä aletaan tutkimaan. Yleisesti näillä pyritään selvittämään asioita, jotka ovat liian komplekseja ja tehoa syöviä normaaleissa tietokannoissa. Näin löydetään uusia kytköksiä asioihin. Tiedon normaaliyhdistely olisi normaaleissa relaatiokannoissa erittäin kallista ja Graph-kanta tuo mahdollisuuden juuri näille kannoille loistaa. Kytkökset kuitenkin tarvitsevat lähtöpisteen, kuten vaikka käyttäjätunnisteen tai nimen. Graph-kantojen ainut yhtäläisyys muiden NoSQL kantojen kanssa on relaatiomallin hylkäys. Graph-kantoja ovat esimerkiksi Neo4J ja FlockDB. (Pramod J. Sadalage & Martin Fowler 2013)

2.3 NewSQL

NewSQL on uusi termi, joka syntyi, kun NoSQL toi esiin suuria ongelmia SQL tietokantojen käytössä. SQL on 30 miljardin dollarin liikevaihdon peruskiviä. Tämä osoittaa, kuinka tarpeellisena SQL on nähty. NoSQL toi mukanaan opetuksen siitä, kuinka SQL:llä on tarve kehittyä paremmaksi ongelmien hallinnassa, skaalautuvuudessa ja laajennettavuudessa. Lisäksi se osoitti, kuinka pilvessä ajettavuus on tärkeää nykypäivän järjestelmille. NewSQL tavoite on tarjota sekä SQL:n hyvät puolet, että NoSQL:n hyvät puolet. Se pyrkii tuomaan SQL:n rakenteellisuuden ja haku kyvykkyyden modernille jaettavuudelle ja ongelmien hallittavuudelle samalla kun se pyrkii tarjoamaan pilvi ja klusterikyvykkyyttä näille tietokannoille. Se tuo SQL:n rikkauden NoSQL:n venyvyydelle. (Ryan Betts 2014)

3 ANSIBLE

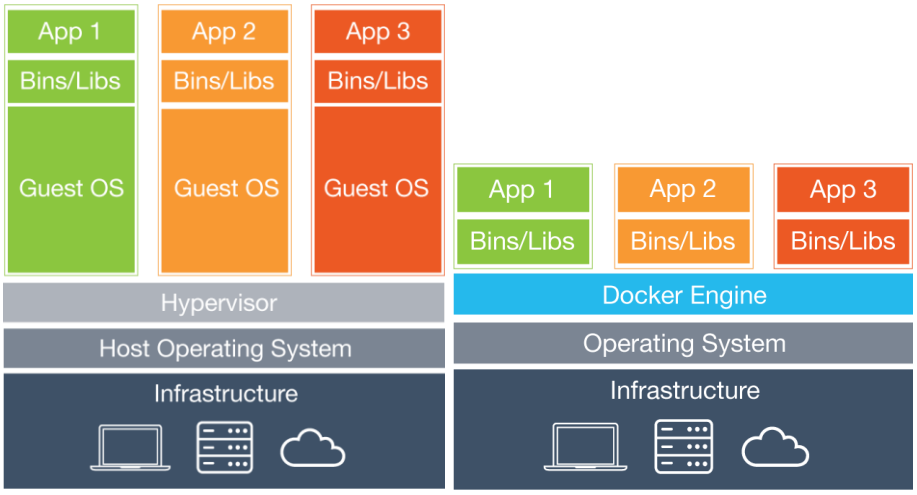
Ansible on selkeä tapa automatisoida konfigurointi, pilvipalvelujen käyttöönotto, hallinta, applikaation julkaisu-, järjestelmien yhdenmukaistamis- ja monia muita tietoteknisiä tarpeita. Ansible on suunniteltu mahdollistamaan useiden eri järjestelmien julkaisun ja käytön. Tavoitteena on ollut ymmärtää miten järjestelmät toimivat toistensa kanssa, jotta hallittaisiin vain yhtä järjestelmää kerrallaan. (Ansible 2015)

Ansible perustuu puhtaasti agentittomaan toimintaan eli pyrkii pyörimään lähes joka tietokoneella ilman erillistä ohjelmaa. Tämä mahdollistaa helpon käyttöönoton ja selkeämmän arkkitehtuurin. Ansible käyttää YAML-kieltä, jota lukemalla se osaa ajaa oikeita komentoja. (Ansible 2015)

Ansible tavoittelee palveliminen yhdistelyä toisiinsa ja ohjaa ohjelmia näille helposti. Ohjelmat on kirjoitettu seuraamaan laitteen sen hetkistä tilaa ja suorittamaan ohjelmia SSH:n avulla. Kirjastojen itsessään ei tarvitse olla kohde tietokoneella, vaan ne voivat sijaita missä tahansa, koska palvelimille ei asenneta mitään. Yleisesti Ansiblen kanssa työskennellään lempitekstieditorin, versionhallintaohjelman ja terminaalin kanssa. (Ansible 2015)

4 DOCKER

Docker mahdollistaa ohjelmien kokonaisuudeksi, joka sisältää kyseisen ohjelman tarvitsemat ohjelmat. Näin pyritään tarjoamaan standardisoidusti sama ohjelma, joka ei muutu vaikka alla oleva tietokone muuttuisi. Docker pyrkii olemaan kevyt yksittäinen tietokone tietokoneen sisällä ja samalla käyttämään muistia tehokkaasti. Dockerit jakavat tiedostoja ja levyä, sekä pohjalla olevaa käyttöjärjestelmää. Docker eroaa virtualisoinnista poistamalla koko käyttöjärjestelmä osan toiminnastaan Kuvan2 tapaan. Tällä mahdollistetaan pienempi jalanjälki, kun ajetaan ohjelmistoja. Docker mahdollistaa kehittäjän työn helpotusta ja ympäristöjen standardisointia. Tarjoten mahdollisuuden että palvelimella kaikki toimii samalla tavalla, mikä helpottaa järjestelmien asennusta ja konfigurointia. Samalla järjestelmän julkaiseminen nopeutuu. Mikä mahdollistaa nopeamman kehitystahdin ja paremman mahdollisuuden saada uusia versioita asiakkaille. (Docker 2015)



Kuva2. Dockerin ja virtualisoinnin ero. (Docker 2015)

5 BIG DATA

Big Data eli suomeksi Suuri Tieto, on alalla vakiintunut termi kuvaamaan tiedon räjähdysmäistä kasvua ja sen ymmärrystä. Kyseistä termiä määritellään yleensä kolmella termillä, tietojärjestelmä valmistaja SAS on päättänyt lisätä kaksi omaa termiä sekaan. Yleisesti kolmeksi termiksi on nähty tiedon koko, nopeus ja erimuotoisuus. SAS on määritellyt, että eroavuus ja kompleksisuus ovat tärkeitä tekijöitä myös Big Datan ymmärryksessä. SAS sanoo Big Datan tulevan olemaan yrityksille yhtä suuri tekijä, kuin Internet aikanaan. SAS:in mukaan syvät ja paremmat analyysit ovat tärkeitä. Niiden saatavuuden ja käsiteltävyyden parantuminen Big Datan myötä tulee kasvamaan jokaisessa yrityksessä. (SAS 2015)

Big Datalla ei tarkoiteta että tietoa on pakko olla paljon vaan, miten tietoa käsitellään. SAS:in suuria toiveita olisi yrityksien kyky pystyä etsimään ja lukemaan tietoa mistä tahansa tietolähteestä sekä tutkimaan ja analysoimaan kyseistä tietoa. Tällä tiedolla voisi sitten tuottaa kulujen optimointia, työmäärien tasaamista, käytetyn ajan optimointia, uusien tuotteiden kehitystä ja päätöksen teon parantamista. Tiedon avulla on helppo saada vastauksia erinäisiin asioihin, kuten matkan optimointiin, kuponkien generointiin ja erinäisten riskien analysoimiseksi. (SAS 2015)

SAS käyttää paketinkuljetus yritys UPS:ää esimerkkinä Big Datan hyödyntämisessä. UPS on kerännyt tietoa jo 1980 vuodesta saakka. Näin ollen heillä on nyt käytettävissään massiiviset määrät tietoa, mitä hyödyntää pakettien kulkemisesta ja miten paketteja on kuljetettu. Tästä voidaan vetää nykypäivänä luvut, jotka sanovat UPSin toimittavan noin 16,3 miljoonaa pakettia päivässä noin 8,8 miljoonalle käyttäjälle. Datamäärissä tämä tarkoittaisi noin 16 petatavua. Suurin osa tästä datasta generoituu nykyään autojen sijainnista, joita UPS seuraa, optimoidakseen reittejä. Autoista kerätään nopeutta, suuntaa, jarrutusta ja ajoa, jolla seurataan autojen päivittäistä käyttöä. (SAS 2015)

5.1 Tiedon koko

SAS kuvaa koon muodostuvan useasta asiasta kuten vanhoista tietovarannoista tai sosiaalisen median tuottamasta tiedosta, joita yritys seuraa. Tämän lisäksi sensori ja koneellinen tieto, sekä kommunikaatiodata kasvavat koko ajan. Tietoa syntyy siis enemmän ja sitä halutaan analysoida ja kehittää pidemmälle. Menneisyydessä tiedon tallennus oli vaikeaa, mutta nykyään tietoa on triviaalia tallentaa. On syntynyt ongelma, kun tarvitsee tunnistaa oleellista tietoa. (SAS 2015)

5.2 Tiedon nopeus

Datan nopeutta on vaikea mitata. Sitä pyritään mittaamaan pitkälti käsittelyyn menevällä ajalla, sekä kauanko datalla kestää saapua järjestelmään. Esimerkkeinä näistä käytetään usein sensoreita ja mittareita. Näiden tarvitsee saada tieto käsiinsä nopeasti, sekä lähettää tieto eteenpäin analyysiä varten. (SAS 2015)

5.3 Tiedon erimuotoisuus

Erimuotoinen data on erityisen ongelmallista, kun kyseessä on useita järjestelmiä, joiden tieto halutaan saada selkeästi yhteen paikkaan. Nykyään on olemassa tuhansia tietotyyppejä, jotka aiheuttavat omia ongelmia. Tietotyyppejä ovat muun muassa videot, strukturoitu data, strukturoimaton data, numerot, dokumentit, kuvat ja äänet. Tämä tarkoittaa, että tätä massiivista määrää tietoa tarvitsee hallinnoida, kontrolloida ja analysoida jotenkin. Lisäksi erinäisiä lähteitä pyritään yhdistämään toisiinsa, jotta tiedosta saadaan hyviä analyysejä. (SAS 2015)

5.4 Tiedon eroavuus

Määrän ja nopeuden kasvaessa, myös tietoon syntyvät erot alkavat kasvamaan. Tässä on kyse siitä, kuinka tietoa tulee erityyppisissä piikeissä, eikä vanhaan totuttuun tapaan tasaisena virtana. Nykyään ei enää tarvita koko ajan samaa laskentatehoa vaan enemmänkin tarpeen mukaan muuttuvaa klusteria. Tähän ongelmaan törmätään varsinkin sosiaalisen median mahdollisuuksista. Asia saattaa kasvaa nopeasti todella suureksi, joka vaatii analyysiä ja kuukauden päästä ketään ei kiinnosta tämä asia enää. Tämä kaikki kasvattaa strukturoimattoman datan määrää, jota tarvitsee hallita enemmän kuin koskaan. (SAS 2015)

5.5 Tiedon kompleksisuus

Data tulee nykyään erilaisista rajapinnoista ja lähteistä ympäri asiakkaan järjestelmää ja verkkoa. Tämä luo laajan tarpeen tiedon kytkemiselle toisiinsa. Lisäksi tiedon puhdistus ja muuntaminen ovat raskaita prosesseja, jotka saattavat laajentua yli kymmenien jos ei satojen järjestelmien välille. Kyseinen työ on kuitenkin pakko tehdä, että tietoa voidaan tutkia ja analysoida. Kytkennät voivat kuitenkin lähteä todella kasvamaan todella laajoiksi, jolloin asiaan tarvitsee tutustua paljon alkua laajemmalla tavalla. (SAS 2015)

5.6 Tieto ympäristöt

Big Data on monen toimijan alue, jossa tunnetuimpana tuotteena toimii Hadoop, joka kytkeytyy lähes kaikkiin muihin ratkaisuihin ja toimii hyvänä pohjana omille dataratkaisuille. (Hortonworks 2014) Ratkaisuja on kuitenkin satoja. Ratkaisuja löytyy lähes joka tarkoitukseen. Kilpailussa on mukana niin HP, IBM kuin Intel. Tämän lisäksi on olemassa Hadooppia itsekseen tarjoavia yrityksiä. Tästä valikoimasta löytyy ratkaisuja useisiin ongelmiin, mutta silti uusia teknologioita tuntuu syntyvän säännöllisin väliajoin. (Big Data Vendors 2013)

Big Datan avulla yleensä haetaan liikevaihtoon, joustavuuteen tai kustannukseen säästöä. Tämän lisäksi moni näkee tärkeänä, että toimitettavat ratkaisut ovat saatavilla nopeasti ja edullisesti. Eficoden ratkaisussa on käytetty pitkälti HP teknologioita, kuten sivustolla esitetään, eli HP:n H.A.V.E.N arkkitehtuuria, joka koostuu erinäisistä tuotteista.. Tämä mahdollistaa erinäisten ratkaisujen toteutuksen toimivalla teknologia ideologialla. Tämän lisäksi on haettu lisää nopeudesta eräällä ratkaisulla. (Eficode 2015)

5.6.1 Hadoop tarjonta

Hadoop on alkujaan avoimen lähdekoodin järjestelmä, mutta sen kaupallistumisen ja yritysten tarpeen vuoksi sen ympärille syntyi kolme suurta tarjoajaa, jotka ovat MapR, Cloudera ja Hortonworks. Näiden tavoitteena oli tarjota helpotusta Hadoopin käyttöön ja kehittää Hadooppia paremmaksi ohjelmistoksi. Näiden kolmen välillä suurin ero on, että MapR sisältää muutamia moduuleja, jotka eivät ole avointa lähdekoodia, kun taas Hortonworks ja Cloudera ovat avoimen lähdekoodin järjestelmiä. (Experfy Editor 2014)

5.6.2 MapR

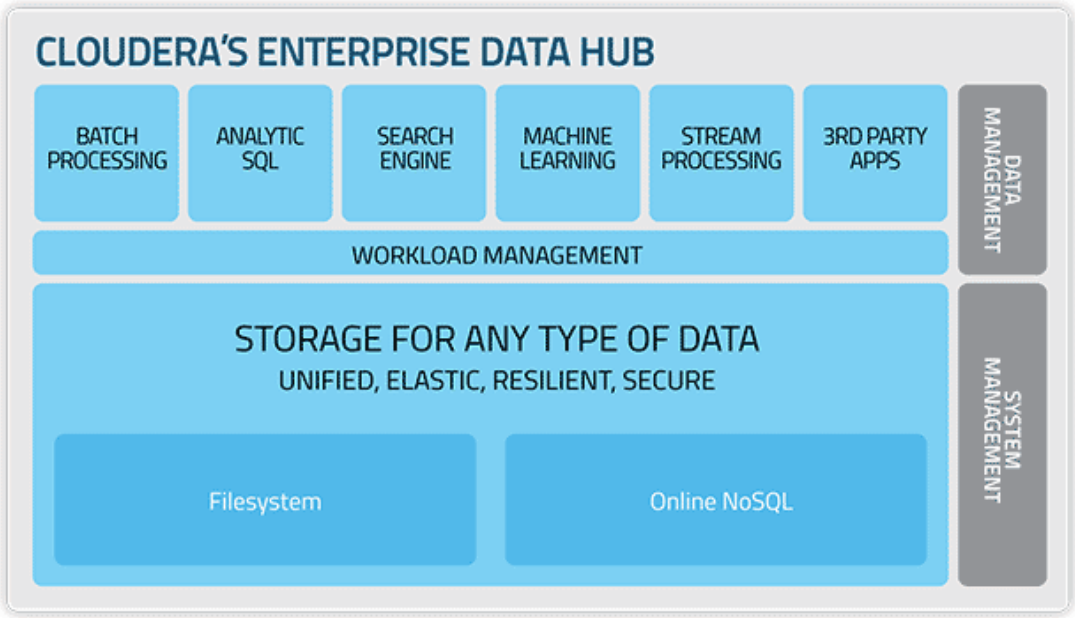
MapR eroaa ehkä eniten kahdesta kilpailijastaan tarjoamalla ratkaisussaan muunnettavuutta. Sen tavoitteena on yhdistää olemassa olevia arkkitehtuurillisia rakennelmia uusiin ratkaisuihin, kuten MapR Hadoop versioonsa. MapR on rakentanut avoimen lähdekoodin alle oman maksullisen versionsa, joka perustuu MapRFS ja MapR-DB työkaluihin, jotka näkyvät kuvassa 3. MapR tavoitteena on tuoda MapR mahdollisimman monen saataville. MapR pyrkii korporaatiotasolla tuomaan NoSQL ja SQL ratkaisut yhteen pakettiin. Tämä helpottaa MapR:n käyttöönottoa yrityksissä. Sen suurin ero kilpailijoihinsa nähden on MapR:n tuki NFS:lle Hadoop järjestelmissä, joka näkyy hyvin, kuvassa 3. (MapR 2015a)

Kuva3 MapR arkkitehtuurikuva, joka kuvaa miten eri työkalut on rakennettu MapR arkkitehtuurille. (MapR 2015)

5.6.3 Cloudera

Clouderan tavoite on ratkaista ja korjata nykyiset arkkitehtuurit tuomalla oma järjestelmänsä nykyisten järjestelmien rinnalle. Ratkaisun tavoitteenaan on helpottaa erinäköisen tiedon hallinta. Tavoitteena on pyrkiä yhdistämään Hadoopin arkkitehtuuri ja vanhat arkkitehtuurit. Näin yrityksien järjestelmät pystyvät laajempaan analyysiin ja työkuormaan kuin ennen. (Cloudera 2015a)

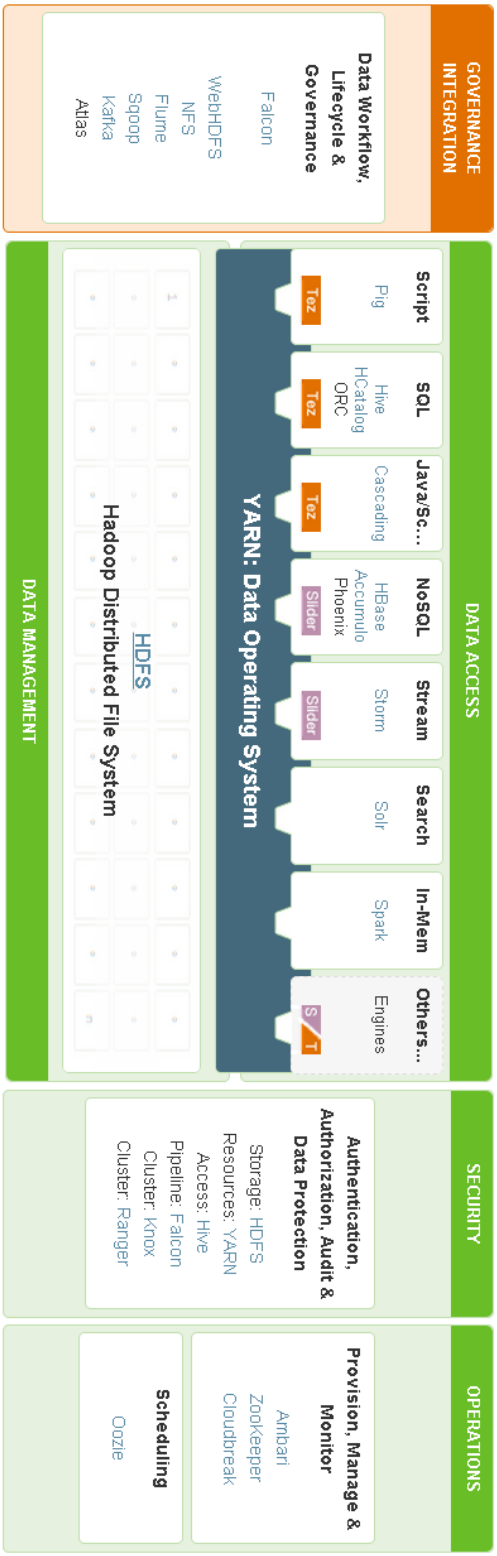
Clouderan yritysarkkitehtuuri perustuu erinäisten komponenttien rakentamiseen Hadoopin peruskomponenttien päälle. Tämän jälkeen siihen kytketään jokin työnhallitsija kuten YARN ja tämän päälle kytketään erilaisia teknologioita. Kuva4 kuvaa sitä miten Clouderaan perustuva Hadoop on rakennettu. (Cloudera 2015c)



Kuva4. Clouderan Hadoop arkkitehtuuri (Cloudera 2015c)

5.6.4 Hortonworks

Hortonworks pyrkii ja on täyteen avoimuuteen. Ratkaisun dokumentaatio- ja koodipohja löytyvät internetistä kaikkien saatavilta. Hortonworksin tavoitteena on yhdistää tietolähteet ja standardoida Hadoop. Tämä mahdollistaa kaikille sen päälle rakennetuille ohjelmille toimivuuden, jokaisessa Hadoop järjestelmässä, joka noudattaa kyseistä standardia. Hortonworks myös pyrkii työllään kehittämään järjestelmää lähemmäksi yritys maailmaa. (Hortonworks 2014)



Kuva5 Hortonworksin arkkitehtuurikuvaus, kuinka eri komponentit kytkeytyvät toisiinsa. (Hortonworks 2015j)

6 HADOOP

Hadoop on suunniteltu alkujaan suurien tietomassojen hallintaan. Tämä syntyy pitkälti tallennuskapasiteetin kasvusta vuosien varrella, joka on johtanut tiedon eksponentiaaliseen kasvuun. Nykypäivänä tilaa menee hukkaan tietokoneista ja palvelimista. Tähän kuitenkin kehitetään ratkaisuja, kuten HDFS eli Hadoopin jaettu levynhallinta. Hadoop pyrki ratkaisemaan siis kapasiteetin hukkakäyttöä. (Tom White 2009)

6.1 HDFS

HDFS suunniteltiin pitkälti Big Datan ympärille. Tästä johtuen sen rakenne oli alkujaan kirjoita kerran ja lue useasti. Hadoop on muuttunut kuitenkin Hadoopin kehityttyä. HDFS lyhenne tulee alkujaan sanoista jaettu tallennustila, johon lisättiin Hadoop sana myöhemmin. Tämä loi erilaisen kanavan, joka helpottaa tiedon käsittelyä. Hadoopin tavoitteena oli alkuun, että sitä voidaan ajaa halvalla ja vanhentuneella raudalla, ilman turhaa investointia. Näin saavutettiin kyky hallita massiivisia määriä tietoa ilman jättiläismäisiä kustannuksia. (Tom White 2009)

Tietojärjestelmä rakentuu kahdesta komponentista eli nimipalvelusta ja tietopalvelusta. Nimipalvelu toimii mestarina ja hallitsee tiedostojen hierarkiaa, sekä näiden viitetietoa. Nimipalvelu tallentaa tiedon oikealle levylle luettavaksi, sekä vahtii, missä mikäkin tieto palvelu on. Tietopalvelun tehtävänä on pääasiassa toimia työläisenä ja komentaa tietomassoja. (Tom White 2009)

HDFS mahdollistaa järjestelmän osien hajoavan koko ajan ja toimintakyky pysyy ilman puuttuvia osiakin. Järjestelmässä on tiukka rakenne siitä miten, milloin ja kuka pääsee kiinni järjestelmiin. Näin ollen jokaisen käyttäjän pitää ilmoittaa HDFS:lle, kenen oikeuksilla hakee ja ajaa tietoa järjestelmään. Näihin tietoihin pääsee käsiksi erilaisten rajapintojen kautta. (Dhruba Borthakur 2008)

6.2 YARN

YARN on Hadooppiin myöhemmin lisätty komponentti. YARN hallitsee klusterin työkuormaa ja resursseja. YARN jakaa työnsä resurssien valvomiseen ja ohjelmien seuraamiseen. Näiden työkalujen tehtävänä on toimia päättäjänä klusterissa. YARN:in tavoite on myös päättää, mikä siihen kytketty työkalu saa milloinkin klusterista tehoa. Jokainen elementti Hadoopissa keskustelee YARN:in kanssa tarpeistaan ja tämä päättää, kuka saa mitä. (Apache 2015)

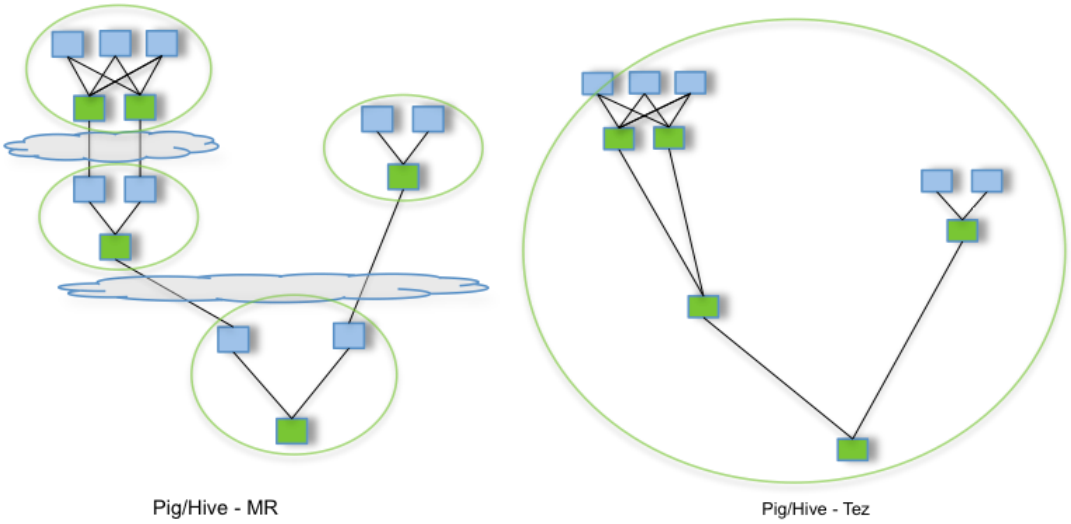
YARN:in resursointi jakautuu kahteen osaan. Toinen hoitaa töiden ajoittamisen klusterissa eli koska mikäkin työ ajetaan. Toisen tehtävä on hoitaa ohjelmienkontrollia. Ajoittaja hoitaa työn sijoituksen, sekä paljonko tälle työlle annetaan resursseja. Ajoittajan tehtävä ei ole valvoa applikaatioin toimintaa vaan se siirtää vastuun kontrollerille. Kontrolleri hallitsee applikaation resursseja ja toimintaa. Näiden komponenttien toimintaa voidaan laajentaa erilaisilla työkaluilla. Ohjelmakontrolleri toimii töiden hyväksynnässä ja neuvottelussa, sen tehtävä on myös tarpeen mukaan käynnistää resursseja uudelleen. (Apache 2015)

6.3 Map/Reduce

Map/Reduce on Hadoopin pohjakäsitteitä. Se on ollut olemassa alusta asti ja sen ympärille on rakennettu laajasti erilaisia työkaluja. Map/Reduce perustuu kahteen erinäiseen tapaan käsitellä dataa. Yleensä pyritään ensiksi kartoittamaan ja muuntamaan tieto oikeaan muotoon, että sitä voidaan myöhemmin yhdistellä. Tämä johtaa tiedon yhdistämiseen, jossa pyritään yhdistelemään avattu tieto, joksikin luettavaksi ja analysoitavaksi tiedoksi. Vähennys tulee aina kartoituksen jälkeen, minkä avulla luodaan harmoniaa algoritmien välille. (IBM 2015)

6.4 Tez

Tez on uusi Map/Reduce järjestelmä, jonka tarkoituksena on nopeuttaa pitkiä töitä. Tez toi mukanaan mahdollisuuden ajaa usean työn sijaan yhdellä ajolla monta ajoa. Tämä mahdollisti paremman näkymän tietovirtaan, sekä paremman datan tunnistuksen avulla tehokkaamman algoritmisen arvauksen lopputuloksesta. Kuva6 osoittaa, kuinka Tez arkkitehtuuri vahvisti aikaisemmin toiminutta, Map/Reduce hitautta ja optimoi resurssien käyttöä. (Apache Tez 2015)



Kuva6. Tezin ja Map/Reducen toiminnallinen ero. (Apache Tez 2015)

6.5 HBase

HBase on skaalautuva tietokanta Hadoopin päällä. HBase perustuu laajasti Googlen BigTable teknologiaan, mutta toimii Hadoopin päällä ja on näin kaikkien saatavilla. Sen tavoite on tarjota palvelu johon voidaan tallentaa todella suuria tauluja, miljardeja rivejä ja miljoonia kolumneja halvan raudan päällä. HBase on kolumnipohjainen tietokantaratkaisu, joka tarjoaa lineaarista ja modulaarista skaalautuvuutta. HBasen ominaisuuksia ovat palvelimelta toiselle siirtyminen ilman uudelleenkäynnistystä, sekä Hadoopin Map/Reduce piilottaminen taaksensa. Se

pyrkii myös tarjoamaan helppoja rajapintoja, sekä laajennettavuutta. (Apache HBase 2015)

YARN mahdollistaa helpon integroinnin erinäisille moottoreille. Yksi näistä on HBase, joka tarjoaa skaalautuvan NoSQL tietolähteen Hadoop klusteriin. HBase skaalautuu hallitsemaan suuria tietomassoja ja mahdollistaa tiedon yhdistelyn erinäisten struktuurien lävitse. HBasen tärkeitä tavoitteita on performanssi, integroitavuus ja ohjelmoijan työn helpotus. HBasen viimeisin versio on virallinen versio yksi eli tuotantovalmis HBase. Tämä sisälsi korkean saatavuuden parannuksia ja kyvyn päivittää ilman palvelun alasajoa. (Hortonworks 2015f)

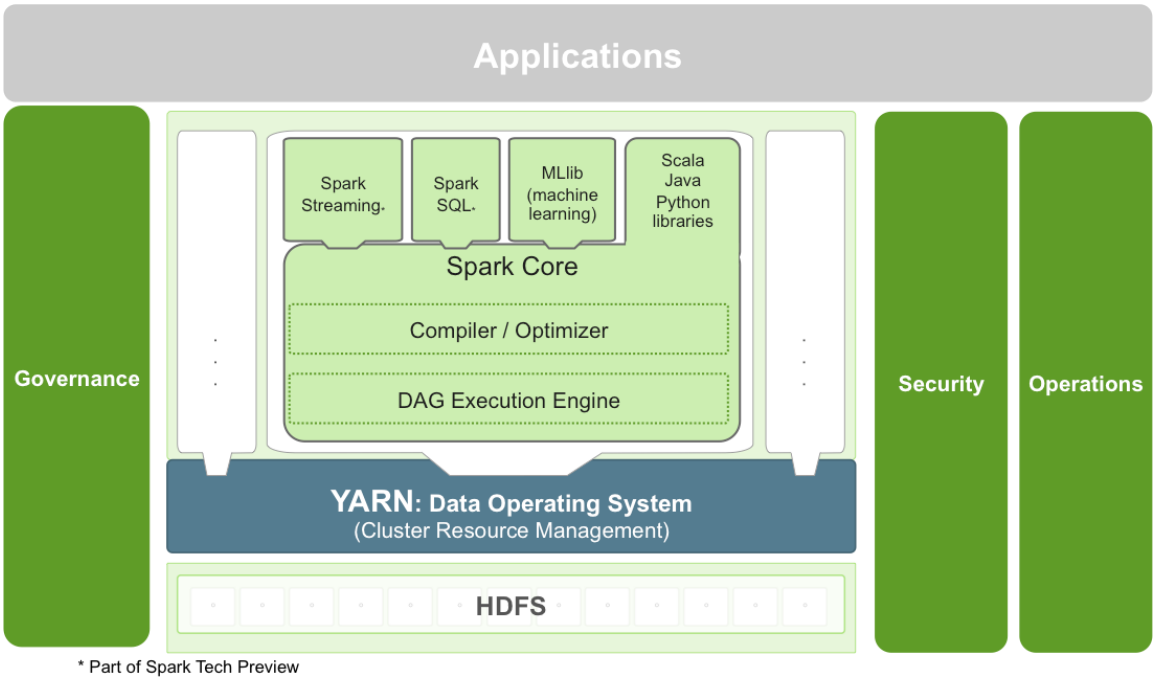
HBase perustuu avaimeen, joita tarvitaan jokaisessa taulussa. Tämä avain jakaa tiedon paketteihin, joita sijoitellaan ympäri klusteria. Jokainen HBase komponentti voi omistaa useita alueita, joille tietoa tallennetaan tai vain yhden. Tällä mahdollistetaan kuorman tasainen jakautuminen klusterissa. HBase osaa jakaa tietoa ympäri klusteria automaattisesti ja pitää tiedon tallessa erillisessä palvelussa. (Hortonworks 2015f)

6.6 Spark

Spark on työkalu, joka on suunniteltu laskemaan suurta dataa. Sparkin alkuperäinen resurssienhallintatyökalu perustuu Apache Mesos projektiin, joka voidaan korvata Hadoop klusterilla. Spark ajaa laskentansa pitkälti tietokoneen muistissa. Spark:ia komennetaan kielillä kuten R, Java, Scala ja Python. Spark kytkeytyy useisiin SQL-moottoreihin. Spark toimii myös levyllä, mikä nopeuttaa myös suurempien tietomassojen laskemista. (Apache Spark 2015)

Hortonworks on yhdistänyt Sparkin omaan järjestelmäänsä ja kehittää Spark:ia kohti parempaa integraatiota ja Hadoop tukea. Hortonworks pyrkii toimimaan Spark:in alla ja yhdistämään Spark:in tuotteisiin, kuten Hive, Storm ja HBase.

Kuva7 osoittaa arkkitehtuuria, jolla Spark ajaa Hortonworkin päällä. Tämä kuitenkin vaatii YARN:in toimiakseen. Samalla Sparkiin on tuotu Hadoopin turvallisuus funktionaalisuutta. (Hortonworks 2015a)



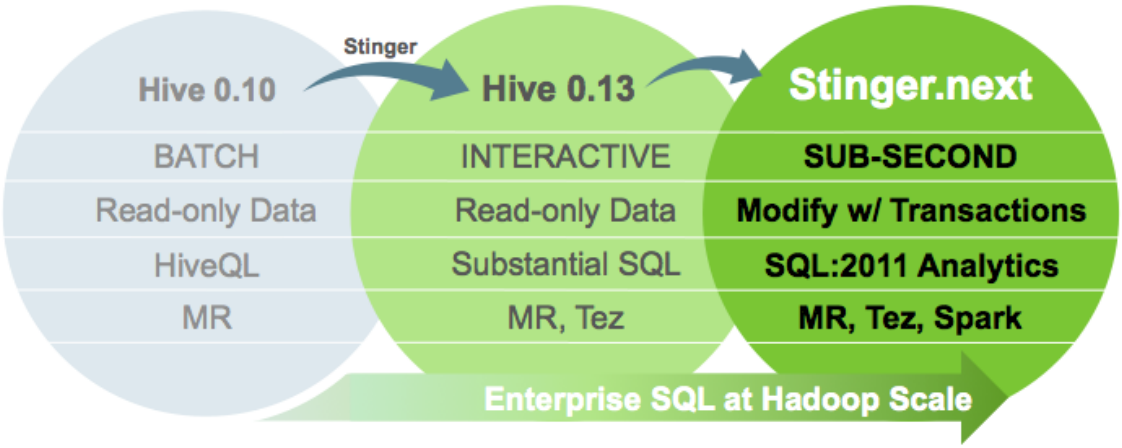
Kuva7. Sparkin toiminnallisuus Hadoop klusterin päällä. (Hortonworks 2015a)

6.7 Hive

Hive on SQL tietokantamoottori Hadoopin päällä. Hive on suunniteltu varastoimaan dataa ja antamaan pihalle analytiikkaa laajoista määristä tietoa. Sen tavoite on tarjota paikka, josta voi kysellä ja hallita tietoa. Hive tarjoaa työkalut rakentaa struktuurisia tietokantoja HDFS datan päälle. Tämän jälkeen dataa voidaan kysellä HiveQL-kielen avulla. Tämän lisäksi Hive antaa myös mahdollisuuden ajaa Map/Reduce töitä Hiven kautta. (Apache Hive 2015)

Apache Hive aloitti matkansa kohti nykypäiväistä versiotaan vuonna 2008. Hive on noussut yhdeksi johtavista SQL moottoreista Hadoopin kanssa. Kuva8. kuvaa Hive:n kehityspolkua. Hive:n seuraavassa suuressa päivityksessä tavoitteena on,

että HIVE tukisi 2011 vuoden standardia ja tarjoaisi kyvyn ajaa millisekuntitason kyselyitä. Hive on pyrkinyt tuomaan tietovaraston perustoiminnallisuudet. Hive tarjoaa JDBC ja ODBC ajurit, joilla se voidaan kytkeä erilaisiin analyyttisiin järjestelmiin. (Hortonworks 2015d)



Kuva8. Hiven kehityspolku.(Hortonworks 2015d)

6.8 Pig

Apache Pig on suoraa YARN:iin integroitu kyselymoottori Hadoop kyselyille. Pigin kaksi suurinta kehitysaluetta tällä hetkellä on performanssin ja analytiikan kautta. Pigiä koodataan PigLatin nimisellä kielellä ja tämä konvertoidaan Map/Reduce töiksi, jotka ajetaan Hadoop klusterissa. Pigin viimeisin versio toi mukanaan tuen Tez moottorille ja automaattiselle rinnakkaisajolle. (Hortonworks 2015b.)

Apache Pig:in tavoite on kehittää helposti ohjelmoitava data-analytiikka kieli. Pig on helposti laajennettava ja optimoitava. Se on rakennettu helpottamaan Map/Reduce töitä. Pig mahdollistaa myös laaja-alaisen rinnakkaisajon. Pig toimii

myös Tez rajapinnan kanssa. Pig:in tavoite on optimoida työskentelyä analyysien kanssa.(Apache Pig 2015)

6.9 Ambari

Ambarin tarkoituksena on luoda hallintatyökalu Hadooppiin, joka mahdollistaa Hadoop klusterin pystytyksen, hallinnan ja monitoroinnin. Ambari tarjoaa helpon ja intuitiivisen verkkokäyttöliittymän. Ambari tarjoaa myös REST-rajapinnan, jonka avulla hallintaa voidaan automatisoida. Ambari tarjoaa integraatiojärjestelmän, johon on helppo kytkeä kehittäjänä ja näin myös laajentaa applikaation tarpeiden mukaisesti.(Apache Ambari 2015)

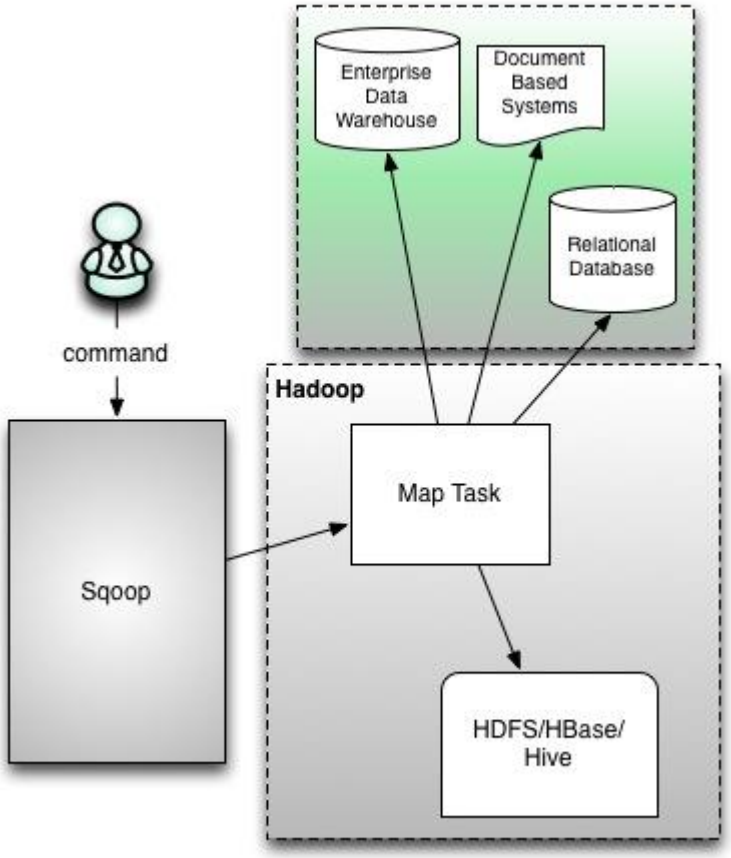
Apache Ambari on täysin avoin järjestelmä, joka sisältää API- ja REST-rajapinnan, asetuksienhallinta, kustomoitavan käyttäjänäkymän, asennustyökalun ja keskitetyn palvelujenhallinnan. Ambari mahdollistaa myös Hadoopin päivittämisen ilman koko klusterin alas ajamista. Lisäksi sen avulla on helppo hallita hälytyksiä ja klusterin turvallisuutta. Ambari kytkeytyy suurimpaan osaan työkaluista helposti ja sitä kehitetään jatkuvasti eteenpäin. Ambari pyrkii tarjoamaan keskitetyn näkymän järjestelmä ylläpitäjälle. (Hortonworks 2015c)

Ambariin ollaan tällä hetkellä kehittämässä Hive näkymiä joilla ylläpitäjät voivat hallita, lukea ja kirjoittaa Hiveen. Samalla sen kautta on tulevaisuudessa helppo valvoa Hive:n käyttöastetta, Tezin hyväksikäyttöä ja ajettuja prosesseja. Samat ominaisuudet kuin Hive tulevat myös Pig:n puolelle, jonka avulla sen hallinta helpottuu ja komentosarjojen tallentaminen tulee mahdolliseksi myös Ambarin puolelta. Samalla Ambariin tulee kapasiteetin suunnittelutyökaluja, joilla voidaan seurata paremmin kuinka rinnakkaiskomennot ajavat ja missä on tarvetta kehitykselle. Lisäksi Ambariin pyritään integroimaan tiedostonäkymä, jonka avulla voidaan seurata HDFS tiedostojärjestelmää. (Hortonworks 2015c)

6.10 Sqoop

Sqoop on suunniteltu tehokkaaseen massadatan siirtoon Hadoopin ja SQL tietokantojen välillä. Tämä mahdollistaa helpomman integraation ohjelmistoille. Sqoop sai ensimmäisen korkeatasoisen julkaisunsa maaliskuussa 2012 ja on jatkanut kehittymistä myös sen jälkeen. Sqoopin versio 2 ei tule olemaan yhteensopiva ensimmäisen version kanssa ainakaan tällä hetkellä. (Apache Sqoop 2015)

Sqoop on yksi Hadoopin arkkitehtuurisista keskittymistä ja tärkeä työkalu datan sisään tuomisen kannalta. Sqoop mahdollistaa sen että Hadoopiin voidaan siirtää yritysten tietovarannoista isoja määriä massadataa. Sqoopia voi myös käyttää datan ulostuomiseen Hadoopista tuettuihin tietovarantoihin. Sqoopin rakennetta on kuvattu Kuvassa 9, joka osoittaa sen kykyä ajaa klusteroituja töitä. Tässä näkyy Sqoopin tapa tehdä kartoitustehtäviä erinäisiin tietokantoihin ja tuoda data takaisin Hadooppiin. Tällä perustalla saadaan helposti käytettyä koko klusterin resursseja tiedon hakemiseen. (Hortonworks 2015e)



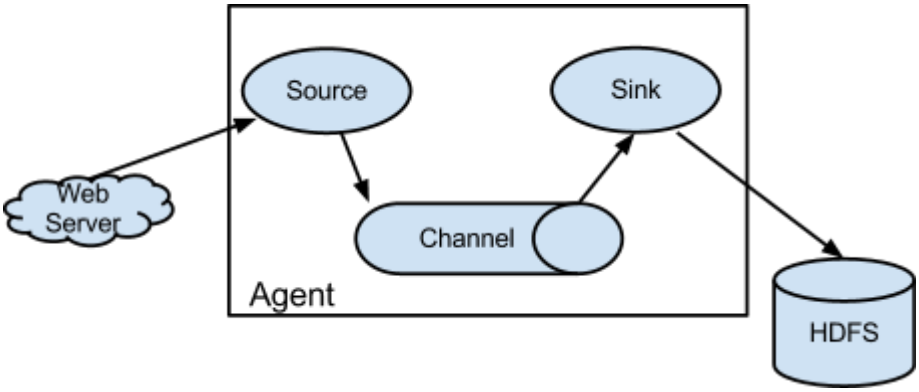
Kuva9. Sqoopin toiminta Hadoop klusterissa.

6.11 Trafodion

Trafodion on HP:n tuottama avoimen lähdekoodin SQL tietokanta. Se toimii Hadoopin päällä käyttäen hyväkseen HBasea. Trafodion on kehitetty tarjoamaan transaktionaalista kyvykkyyttä, sekä mahdollistamaan Hadoopin avulla normaalien yritysohjelmien toimimisen. Trafodion on hakemassa pääsyä Apache ohjelmistokehitys järjestön alle. Trafodion julkaistiin Githubissa 2014 ja on siitä saakka toiminut avoimen lähdekoodin ohjelmana ja kasvattanut asiakaspohjaansa. Trafodionin ensimmäinen virallinen versio julkaistiin 2015 vuoden alussa ja se tukee täysin ANSI SQL:ää. Trafodionilla pystytään lisäksi korvaamaan osa yritystietovarannoista. (Stack Michael 2015)

6.12 Flume

Flumen tarkoitus on tarjota jaettu ja luotettava palvelu, joka kerää ja hallitsee datan, joita palvelut tuottavat kuten lokitiedostoja. Sen tavoite on olla simppeli ja notkea, jonka avulla on helppo käsitellä datavirtaa, jota syntyy nykypäivän järjestelmissä. Kuva10. esittelee Flumen yksinkertaista arkkitehtuuria. Flumella on olemassa tietty kohde, josta otetaan tieto ja joka lopulta tiputetaan Hadooppiin. (Apache Flume 2015)



Kuva10. Apache Flumen toiminta korkealla tasolla.(Apache Flume 2015)

Flumen suurin tarkoitus Hadoop jakeluissa on tuoda raakadataa suoraan Hadooppiin ja näin mahdollistaa datan käyttäminen suoraan analytiikkaan muiden työkalujen kanssa. Flumessa tällä hetkellä suurin kehityssuunta on parantaa kykyä yhdistyä muiden Hadoop komponenttien kanssa ja mahdollistaa parempi analyyttinen toimivuus myös Flumessa. Flume mahdollistaa datavirran hallinnan ja varman tuonnin tietojärjestelmään. (Hortonworks 2015g)

Flumen korkeantason arkkitehtuuri perustuu pitkälti helposti laajennettavaan ja hyvin luotettavaan koodiin. Flumen komponentit kasaantuvat tapahtumien hallinnasta, lähteiden hallinnasta, datansaapumispisteiden hallinnasta, kanavista, agenteista ja asiakkaista. Asiakas toimii ensimmäisenä komponenttina ja sen tavoite on luoda ja vahtia tapahtumia, sekä lähettää nämä tapahtumat eteenpäin lähteiden hallinnalle. Lähteiden hallintaa valvoo agentti. Agentin tehtävä on pitää

huoli, että Flume pyörii siellä, missä sen pitää ja että tapahtumat, lähteet ja saapumispisteet ovat toiminnassa. Kanavan tavoite on toimia keskusteluväylänä lähteen ja saapumispisteen välillä. Saapumispiste pyrkii toimittamaan tiedon järjestelmään, joka yleisesti on HDFS tallennustilaa. Lähde taas pyrkii ottamaan vastaan tapahtumia ja siirtämään ne kanavalle, joka toimittaa ne tallennettavaksi. Tapahtuma taas kuvaa yhtä tietoriviä tai tietoa, joka tulee Flumen lävitse. (Hortonworks 2015g)

6.13 Mahout

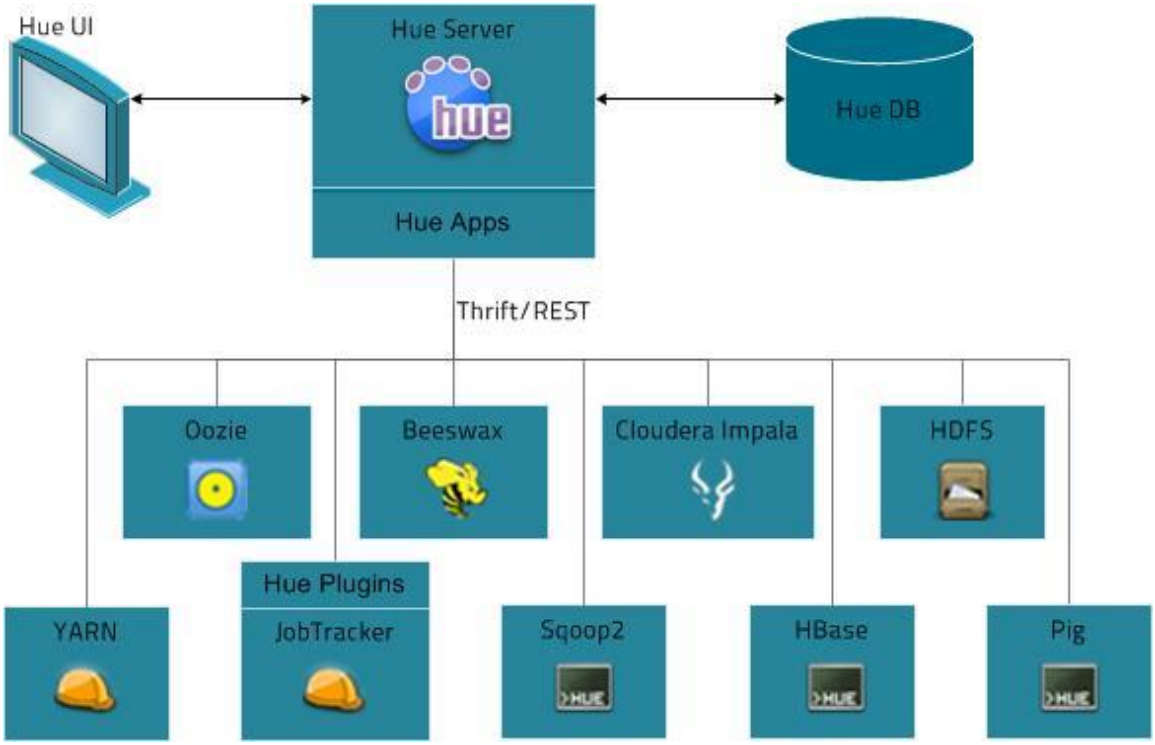
Mahout on skaalautuva koneoppimiskirjasto. Mahout jakautuu kolmeen kategoriaan, jotka ovat skaalautuvat algoritmit, Spark toiminnallisuuden kasvatus ja Map/Reduce algoritmit. Mahoutin kehityksessä otetaan huomioon skaalautuvuus ja algoritmien implementoinnin helppous, joka mahdollistaa käytön ilman algoritmin itsekirjoittamista. (Apache Mahout 2015)

Hadoopissa Mahout tarjoaa helposti skaalautuvaa algoritmista laskentaa Hadoop klusterin Map/Reduce teknologialla. Mahoutin avulla voidaan myös automaattisesti etsiä tärkeitä kaavoja tiedosta, joka mahdollistaa paremman analytiikan. Mahout tarjoaa useita koneoppimiskirjastoja ja algoritmeja, jotka voidaan ajaa lokaalisti tai skaalatusti ympäri klusteria. Mahoutin algoritmit voidaan laukaista komentokehotteelta. Mahout tulee neljällätoista valmiilla algoritmilla, jotka voidaan lajitella neljään kategoriaan. Nämä ovat tiedon rajaus, yhdistäminen, luokittelu ja useasti löytyvä tieto. (Hortonworks 2015h)

6.14 Hue

Huen tarkoitus on mahdollistaa Hadoop klusterin kanssa toimiminen verkkosivulta. Se tarjoaa välineet HDFS:n selaamiseen, Hiven, Map/Reduce töiden sekä Oozien seurantaan. Hue perustuu ajatteluun, että et tarvitse erillistä ohjelmaa ajaaksesi

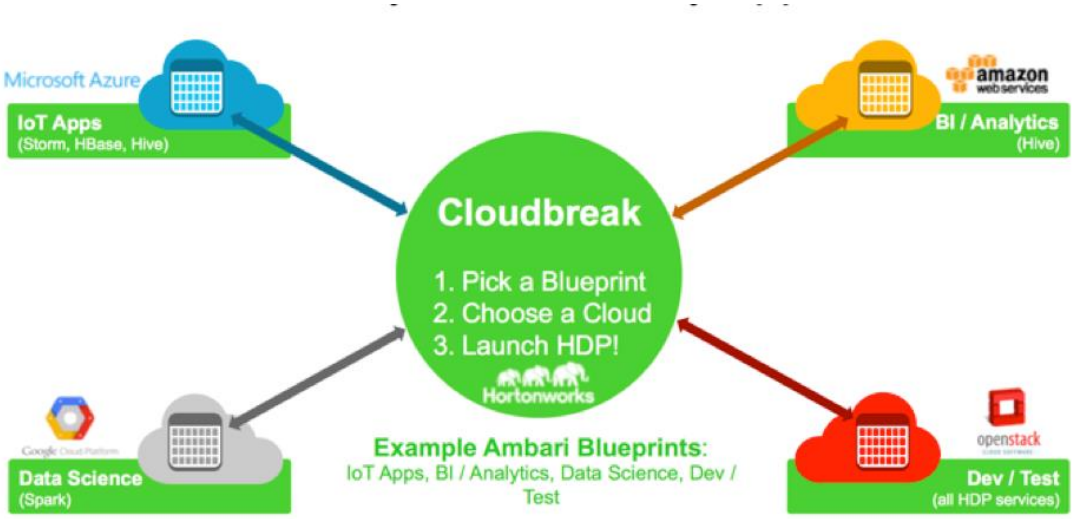
palveluja. Kuva11. näyttää miten Hue toimii sen asentamisen jälkeen ja miten se integroituu työkaluihin. (Cloudera 2015b)



Kuva11. Hue toimintalogiikka Hadoop klusterin päällä. (Cloudera 2015b)

6.15 CloudBreak

Cloudbreakilla voidaan julkaista Hadoop helposti pilveen ja hallita sitä yhden järjestelmän kautta. CloudBreak tukee Azurea, AWS:ää, Google Cloud:iin ja OpenStack:iin. Se tuottaa palvelunsa luomalla Docker instansseja, jotka mahdollistavat paremman skaalautumisen ja asioiden liikuttamisen klusterin käynnistyksen jälkeenkin. CloudBreak on suunniteltu tarjoamaan tarpeen mukaan palvelua Hadoopille ja samalla kytkeytymään erinäisiin ulkoisiin järjestelmiin. CloudBreak toimii Ambarin kanssa yhdessä ja luo klusterinsa Ambarin työkalujen avulla. Kuva12. osoittaa kuinka CloudBreak toimii ja miten paljon CloudBreak yksinkertaistaa rakennetta. (Hortonworks 2015i)

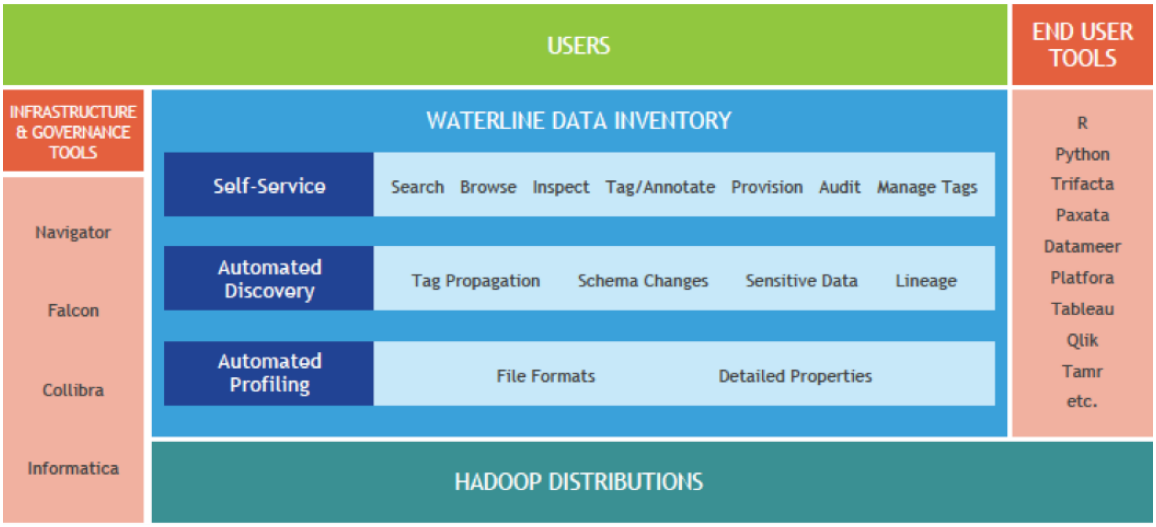


Kuva12. CloudBreakin toiminta yksinkertaisuudessaan ja miten sitä käytetään. (Hortonworks 2015i)

6.16 Waterline

Waterline-ohjelma rakentuu vaatimuksiin, joihin yritykset ovat tottuneet vuosien varrella eli tiedonhallinta- ja metadata-palveluja. Hadoop lupaa uutta näkymää tietoon ja uusien asioiden löytämistä tästä, mutta tällä hetkellä tämä on ollut vaikeaa, koska Hadoop on nähty vaikeana käyttää ja hallita. Suurin ongelma on, että Hadooppiin kertyvät datamäärät ovat massiivisia. Tarkoittaen, että niiden manuaalinen kategorisointi ja tutkiminen lähestyvät mahdotonta. Tätä varten tiedon löytäminen, ymmärtäminen ja kategorisointi tarvitsee automaattisia työkaluja. Waterline kartoittaa automaattisesti Hadoopissa olevan tiedon. Tämän avulla tietoa voidaan paremmin analysoida, koska on olemassa tieto mitä dataa järjestelmään on saapunut. Waterline pyrkii myös seuraamaan tiedon alkuperää ja kytkeytymistä muuhun dataan. Waterlinen isoin työkalu on sen katalogi siitä mitä tietoa järjestelmässä on. Tämän lisäksi se tarjoaa mahdollisuuden osoittaa tietoa henkilöille tai järjestelmille, kuten Hive:lle käyttöliittymästään. Tietoa voidaan myös tutkia ja ymmärtää Waterlinen lävitse, jonka avulla voidaan helpommin päättää tarvitaanko

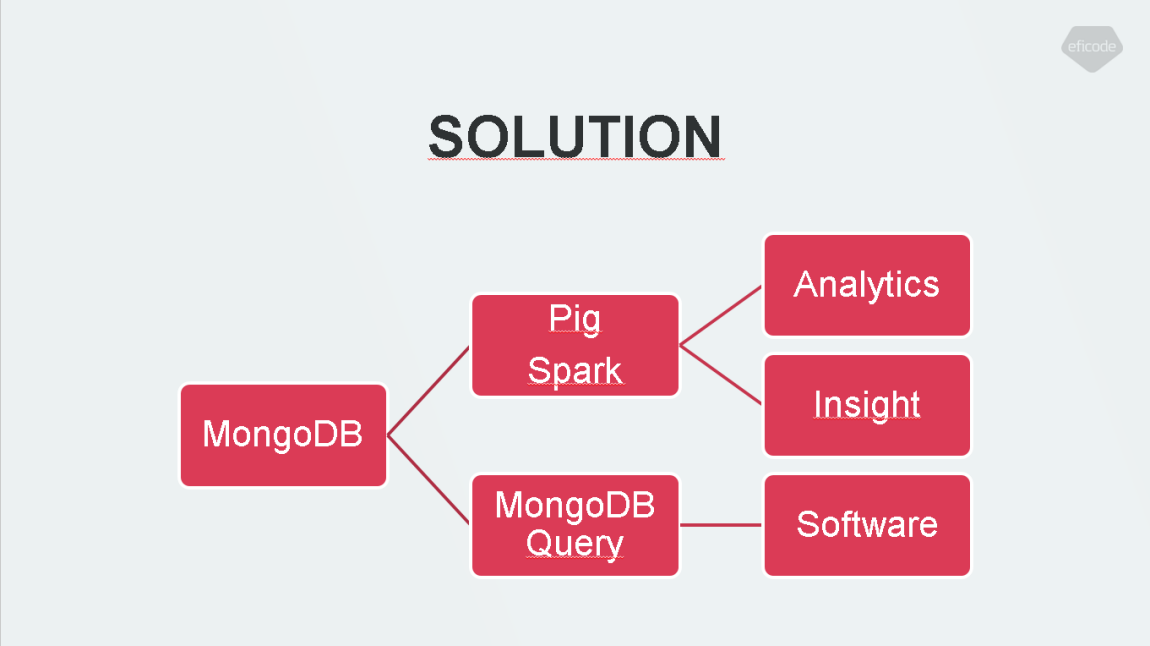
kyseistä tietoa järjestelmässä. Kuva13. Selkeyttää Waterlinen sopimista arkkitehtuuriin ja sen toimimista Hadoopin päällä. Kuvassa näkyy kuinka käyttäjä saa itsepalvelua, automaattista tiedon löytämistä ja mahdollisuuden profiloida tietoa samalla, kuin erilaiset analytiikka työkalut voivat suoraa yhdistää itsensä tietoon. (Waterline 2015)



Kuva13. Waterlinen referenssiarkkitehtuuri. (Waterline 2015)

7 KÄYTÄNTÖ

Hadoopista käytännössä tullaan käymään lävitse miten työkalut asentuvat sekä miten ne toimivat. Tämän lisäksi pyrin tuomaan esiin helposti ongelmiin johtavia kohtia, jotka ainakin itselleni aiheuttivat päänvaivaa aikanaan. Lisäksi pyrin käymään lävitse miksi nämä työkalut on hyvä olla käytettävissä. Kuvan 14 tapaan on olemassa useita tapoja käyttää Hadooppia toisten tuotteiden kanssa.



Kuva14. Esimerkki arkkitehtuuri MongoDB datan analysoinnista.

7.1 Asennus

Hadoopin asennukseen on olemassa työkaluja kuten Ambari, jotka helpottavat klusterin käyttämistä ja asentamista. Ambari antaa suoraan listan siitä, mitä asetuksia tulee muuttaa klusterissa palvelimittain. Ambari vaatii toimiakseen SSH-avaimen, jolla on pääsy jokaiselle klusteriin liittyvään serveriin. Tämän lisäksi palvelimien tulee tietää toistensa DNS nimi eli se nimi millä nimipalvelimet tuntisivat

serverin. Tämä voidaan hoitaa esimerkiksi kirjoittamalla nämä manuaalisesti koneiden omaan hosts-tiedostoon. Tämän jälkeen Ambarin avulla voidaan päättää, mille palvelimille ohjelmia jaetaan. Tarjolla on tällä hetkellä laaja tarjonta, joka kasvaa uusien versioiden myötä. Opinnäytetyön teko hetkellä pystyin asentamaan seuraavat ohjelmat automaattisesti HBase, Hive, Spark, Tez, Flume, Sqoop, Pig, HDFS, YARN ja Map/Reduce. Kun nämä palvelut on määritetty asentumaan Ambari pyytää esimerkiksi Hiven kanssa tiedon siitä minne metadata tulee tallentaa. Tässä vaiheessa on myös mahdollista säätää eri palvelujen asetuksia ja toimintaa erilaisilla tavoilla ja konfiguraatioilla.

Tärkeitä asioita on hallita klusteria jotenkin keskitetysti, että jokainen palvelin on samalla tavalla asennettu. Tämä helpottaa virhetilanteissa koska kaikki palvelimet ovat samalla tavalla tehtyjä. Ansible sopii tähän palvelimien vakioimiseen hyvin ja tuo tarvittavaa mahdollisuutta hallita näitä tarpeen tullessa. Ambarilla asentaessa tulee lista mahdollisista kehityskohteista järjestelmän tilassa. Nämä kannattaa ottaa tosissaan tai muuten ongelmia tulee nousemaan hyvinkin nopeasti. Tämän takia Ansiblella on hyvä automatisoida erinäisten asioiden konfigurointi kaikille palvelimille.

Järjestelmän asennuttua on hyvä asentaa tarpeelliset lisäkomponentit, jotka tämän opinnäytetyön sisällössä olivat Trafodion, Mahout, Hue ja Waterline. Cloudbreak on oma palvelunsa, joka ei suoraan liity jo olemassa olevaan klusteriin vaan enemmänkin yhdistää sitä ja pilveä, mutta se ei itsessään pyöri suoraan Hadoopin päällä, joten käsittelemme sen jälkeenpäin.

Trafodionin kohdalla joutuu vielä tekemään hieman manuaalista työtä. Tärkeä asia huomata on, että Trafodion vaatii koko klusteria olemaan mukana HBase klusterissa tällä hetkellä. Se ei osaa päätellä, mitkä palvelimista eivät olisi HBase klusterissa. Tästä johtuen Trafodionin asennusta ennen on tärkeää tarkistaa, että jokainen palvelin on asetettu sisältämään HBase komponentit. Tämän jälkeen Trafodion tarvitsee omat tiedostonsa. Trafodionin tarjoamilla työkaluilla voi tarkistaa klusterin olevan oikealla tavalla asetettu, jos näin ei ole on suositeltavaa tehdä muutokset, joista tulee maininta. Tämän jälkeen Trafodion asentaja käynnistetään,

jolle annetaan asennettavien palvelimien nimet ja Ambarin osoite. Tätä seuraa asennus, joka palveluiden uudelleenkäynnistyksiä, kuten HBasen ja YARNin alhaalla käymisen. Trafodionin asennuttua se on käyttövalmis ja siihen voi ottaa yhteyttä ulkopuolelta.

Mahout on huomattavasti helpompi asentaa, koska se voidaan asentaa vain kertomalla servereille, että asenna Mahout. Tämän jälkeen Mahout on valmis toimintaan ja siirrytään sen käyttämiseen.

Huen asennus seuraa Mahout:tin ohjeistusta laajasti, mutta sen kohdalla tulee käytössä olevat komponentit erikseen asettaa järjestelmälle. Näin Hue tietää, mihin se kytkeytyy ja miten se toimii tämän klusterin päällä.

Waterline valitettavasti on vielä helposti saatavilla vain testiversiona, mutta asennus itsessään on suhteellisen yksinkertaista. Waterline tarvitsee oman käyttäjänsä, jolle annetaan oikeudet HDFS-levylle ja se saa oikeuden toimia ylläpitotehtävissä järjestelmässä. Waterline pitää tämän jälkeen purkaa ja ajaa asennusohjelmat ohjeiden mukaisessa järjestyksessä, jolla saadaan Waterline kytkettyä Hadoop järjestelmään. Tämän jälkeen Waterline käynnistyy verkkokäyttöliittymässä ja sen käyttäminen ja profiloiminen alkaa.

CloudBreakin kohdalla niin kuin aikaisemmin sanoin ei ole kyse Hadoop klusteriin kytkeytymisestä vaan sen laajentamisesta pilveen helposti. CloudBreak tarjoaa ylläpitopalvelua, mutta myös mahdollisuutta itse asentaa järjestelmä. CloudBreak tukee tällä hetkellä vain yhtä palvelintyyppiä per pilvi, joten se voi vaikuttaa pilvien valintaan. Yksinkertaisuudessaan CloudBreak asennetaan lataamalla asennustiedosto heidän ohjeellaan, jonka jälkeen järjestelmä initialisoidaan ja käynnistetään.

7.2 Käyttäminen

Pohdinta osio tulee sisältämään mielipiteeni käytettävyydestä ja kytkemisestä. Tässä osiossa kerrotaan miten asiat toimivat teknisesti ja kuinka niihin sai yhteyden, miten niitä määritettiin ja millaisia ongelmia syntyi. Tästä johtuvana syynä lähden otsikkojärjestyksessä menemään työkaluja läpi ja kerron niiden käyttämisestä.

7.2.1 HDFS

HDFS toimii kahdella muodolla. Toisessa sitä käskytetään REST-rajapinnan kautta ja toisessa tuttujen komentojen avulla suoraan käyttöjärjestelmältä. Käytössä on suurin osa Linux komennoista, joilla tietoa voi tuoda sisään ja ulos, sekä selata. Käyttäjät luodaan Linuxin puolella ja niiden oikeuksia hallitaan XML-tiedostoilla, jotka kertovat mitä oikeuksia käyttäjällä on.

Näitä voidaan myös graafisesti hallita Ambarin avulla. Ambari versioi muutokset, joten on helppo palata takaisin jos jokin muutos ei toimi. HDFS käyttäminen yleisesti menee useilla komennoilla kuten Hadoop FS, joka tarkoittaa aina Hadoop tiedostojärjestelmä komentoa. Yleisesti näillä jaetaan muun muassa käyttäjäoikeuksia ja pyritään tarkistamaan virtaako data oikein Hadooppiin.

7.2.2 YARN

YARN itsessään ei ole komponentti johon yhdistetään vaan, se tarjoaa rajapintoja joita voi käyttää järjestelmien kytkemiseen. Tästä johtuen sitä kannattaa ohjastaa Ambarin versionhallinnoitujen asetustiedostojen kautta. Näin vältytään tuhoisilta muutoksilta ja on helppo tuoda takaisin toimivat muutokset, sekä saada järjestelmä takaisin toimintaan. Sparkin asennus manuaalisesti Hadoop klusteriin on

kahden asetuksen työ Ambarin päädyssä, koska YARN on tehnyt asioiden kytkemisestä helppoa. Käyttäjänoikeuksien antamisen pääty Hadoopin päästä on myös helpottanut prosesseja. Tässä abstraktio on tehty Sparkin päästä ja näin on helppoa olla keskittymättä YARNin osuuteen.

7.2.3 Map/Reduce ja Spark

Map/Reducen käyttöön on useita rajapintoja. Käytetyin näistä on Javaan perustuva rajapinta. Suurin osa käytöstäkin on yleensä opetettu juuri tällä rajapinnalla. Map/Reduce pyrkii tarjoamaan hyvät työkalut tiedon analysointiin ja asioiden työstämiseen tarvittavaan muotoon. Java on tunnettu ja osattu kieli, joten sen käyttäminen on selkeää ja dokumentoitua. Tämän lisäksi Map/Reduce tukee muita kieliä, jotka mahdollistavat laajemman käytön.

Spark mahdollistaa erinäisten kartoitus ja vähennys töiden tekemisen Scala-, Python- ja R-kielen kautta. Scala on näistä parhaiten tunnettu ja sille löytyy paljon valmiita ohjelma esimerkkejä. Pythonin kanssa vastaan tulee ongelmia tietyissä tehtävissä, koska se on vielä kehityksessä. R on näistä uusin ja en lähtenyt testaamaan sitä vielä laajemmin. Sparkin ajaminen ja kirjoittaminen onnistuvat erinäisten Eclipse kytkentöjen kautta.

Map/Reduce elementtiä ei kovinkaan usein kirjoiteta enää nykyään. Kyseistä teknologiaa korvaa huomattavasti Spark:illa. Spark on helppo asentaa YARN:in avulla. Näin saadaan nopeammin tuloksia ja tarve vähemmälle ohjelmoinnille. Spark:in suurimpia ongelmia on lähiaikoina ollut siihen tuotujen kielien toimimattomuus. Omissa testeissäni esimerkiksi Python-kielellä tekeminen on ollut lähes mahdotonta, koska se ei tue suurinta osaa kirjastoista.

7.2.4 Pig

Pigin kirjoittaminen vaatii, että henkilö ymmärtää kartoitus ja vähennys töiden teorian, eikä vain Hadoopin Map/Reduce tyyliä. Pig on kuitenkin tehokas työkalu NoSQL datan laskemiseen samalla tavalla kuin Map/Reduce ja Spark ovat, koska näitä kaikkia ohjelmoidaan vastaamaan sitä mitä tiedosta yritetään saada pihalle. Pig kuitenkaan ei perustu valmiiseen ohjelmointikieliin tästä johtuen sen käyttäminen perustuu PigLatin kieleen. Pig:llä on tähän mennessä ollut todella hyvä kirjoittaa muun muassa analyyttisiä kyselyitä erinäisille NoSQL-tietokannoille, koska siihen on helppo rakentaa samanlaista funktionaalisuutta, kuin nämä tarjoavat, mutta toimivammalla syntaksilla.

7.2.5 Ambari

Ambari tarjoaa graafisen näkymän klusterintoiminnasta ja konfiguraatioiden versionhallinnan. Lisäksi sen avulla on helppo lisätä palveluja palvelimille, sekä uudelleen käynnistää palveluja. Ambari tarjoaa myös virheraportit eri asioista, kuten klusterin toiminnasta, sekä tietoja esimerkiksi levynkäyttämisestä. Ambari:lla pystyy myös luomaan useita Hadoop klustereita käyttämällä REST-rajapintaa, sekä säätämään jo olemassa olevia rajapintoja REST-rajapinnan lävitse. Ambari:lla on mukava hallita minkä tahansa kokoista klusteria, sekä asetuksien muokkaamista, koska se antaa mukavamman käyttöliittymän kuin mitä XML-tiedostot, joita yleensä konfiguroidaan.

7.2.6 Hive

Hive on Hadoopin analyyttinen kanta. Sen tarkoitus on ajaa pitkiä kyselyjä ja mahdollistaa tuttu SQL-kieli, joka mahdollistaa erinäisten tehtävien ajamisen Hive:n lävitse. Hive:en kytkeytyminen on helppoa ja sille löytyy useita kirjastoja, joiden

avulla pääsee käsiksi Hive tietoon. Hive:n metadata tallennus on myös hyvä työkalu erinäisten muiden työkalujen yhdistelyssä, koska se mahdollistaa Hive:n kytkemisen Sparkkiin ja muihin tietokantoihin. Hive:llä on helppo visualisoida dataa muun muassa Excelin avulla. Hive:n ja Excelin avulla voidaan muun muassa piirtää kartalle missä alueilla käyttäjät asustavat IP-osoitteen perusteella.

7.2.7 Sqoop

Sqoop perustuu JDBC ajurien toimintaan, joten niiden asettaminen tulee tutuksi tämän työkalun kanssa. Lisäksi Sqoopin kanssa on tarve tietää miltä tietokanta näyttää, joten taulujen nimien ja kenttien kirjoittaminen tulee hyvin äkkiä tutuksi. Sqoopin toimintaan saaminen on nopeaa ja se toimii nopeasti erinäisten tietokantojen kanssa.

7.2.8 HBase

HBase on NoSQL tietokannaksi luokiteltava kanta, joka tukee useita lähestymisiä siihen kytkeytymiseen. HBase tukee erinäisiä rajapintoja ja ehkä suosituin on sen Java rajapinta. Lisäksi, sitä käytetään paljon myös komentokehotteelta. HBase toisin kuin useat muut kolumnimaiset tietokannat, ei tue SQL-kieltä tiedon hakemiseen. Tämä kuitenkin tarkoittaa, että HBasen kyselyrajapinta on kehitetty erilainen lähestyminen mielessä. HBasen vahvuuksia on juurikin sen laajennettavuus ja päälle rakennettavuus.

7.2.9 Trafodion

Trafodion pyörii HBasen päällä ja näin ollen tuo HBasesta puuttuvan SQL-tuen sen päälle. Trafodionia kutsutaan NewSQL-kannaksi, koska se toimii niin samalla

tavalla kuin monet NewSQL-kannat, mutta itse se pyrkii tuomaan Oraclen toiminallisuuksia Hadoopin päälle. Tässä se on onnistunut ainakin osittain. Trafodionista löytyy Oraclen käyttämiä haku ja kirjoitus vaihtoehtoja. Se pyrkiikin tarjoamaan SQL toiminnallisuutta Hadoopin päällä. Tässä se on onnistunut hyvin, mutta valitettavasti yhdistys Trafodioniin on vielä työlästä muualta kuin Java koodista. Tämä johtuu erilaisten asiakasohjelmien puutteesta ja pakotteesta ODBC käyttämiseen.

7.2.10 Flume

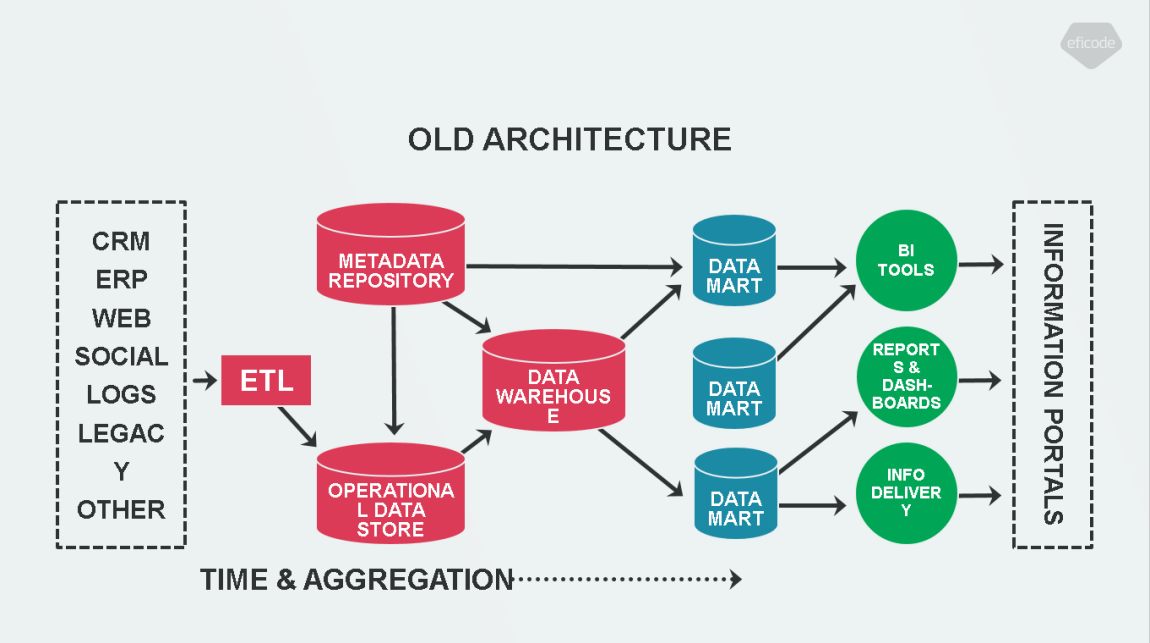
Flumelle osoitetaan kohde, jonka jälkeen se alkaa tuottamaan tiedostoja pihalle. Flumen asettaminen on helppoa ja vähä vaivaista. Flume soveltuu hyvin lokitiedostojen sisälle ajamiseen. Flumea voi käyttää moneen muuhunkin tarkoitukseen, kuten NoSQL kantojen sisällönlukemiseen. Flumen asetuksien hallinta on helppoa ja siinä mukavaa on, että sille rakennetut lokidatan haku ohjelmat jatkavat tiedonhakemista, jolloin datan virtaa voidaan jatkaa aina Hive:en. Tämän avulla voidaan piirtää kartalle sopivalla tahdilla informaatiota.

7.2.11 Waterline

Waterline tarjoaa työkalun, joka sopii tarpeeseen eli Hadoopin inventoimiseen. Tällä työkalulla saadaan helposti tietoa irti siitä mitä Hadoop klusterissa on. Lisäksi verkkosivuna toimivuus antaa sen käytettäväksi helposti ympäri yritystä. Sen tärkeänä pointtina oleva itsepalvelun tarjonta on myös mukava yllätys, sekä datan siirtäminen Hive:en auttaa työlästä prosessia.

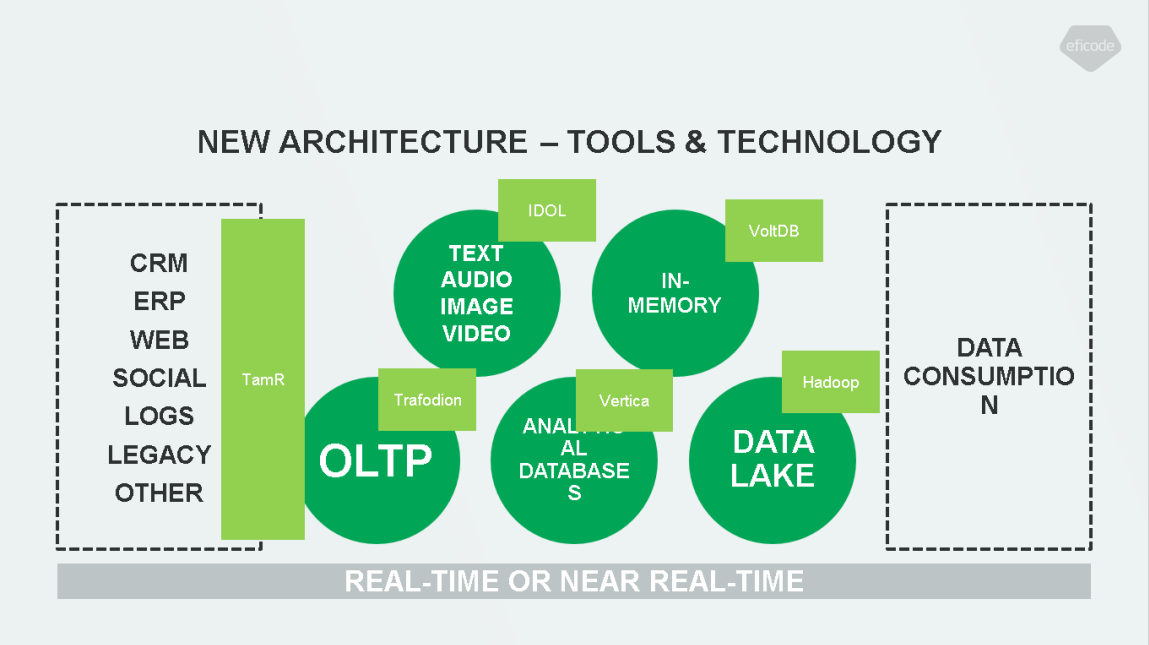
7.3 Hadoop ja Yritysmaailma

Hadoop soveltuvuus yritysmaailmaan tulee pitkälti siitä, että sen avulla pyritään saavuttamaan yrityksissä kolmea eri tavoitetta eli kustannus säästöjä, keskitettyä tietovarantoa ja prosessien pois siirtoa kalliista järjestelmistä. Hadoop toimii yhtenä tärkeänä työkaluna tietovarastojen hajanaisuutta vastaan. Kuvassa 15 esiteltyyn tapaan on yleisesti pyritty arkkitehtuurillisesti tekemään analytiikkaa. Tämä prosessi on kuitenkin hidas ja yleisesti yhden raportin ulos saanti voi kestää jopa kaksi päivää.



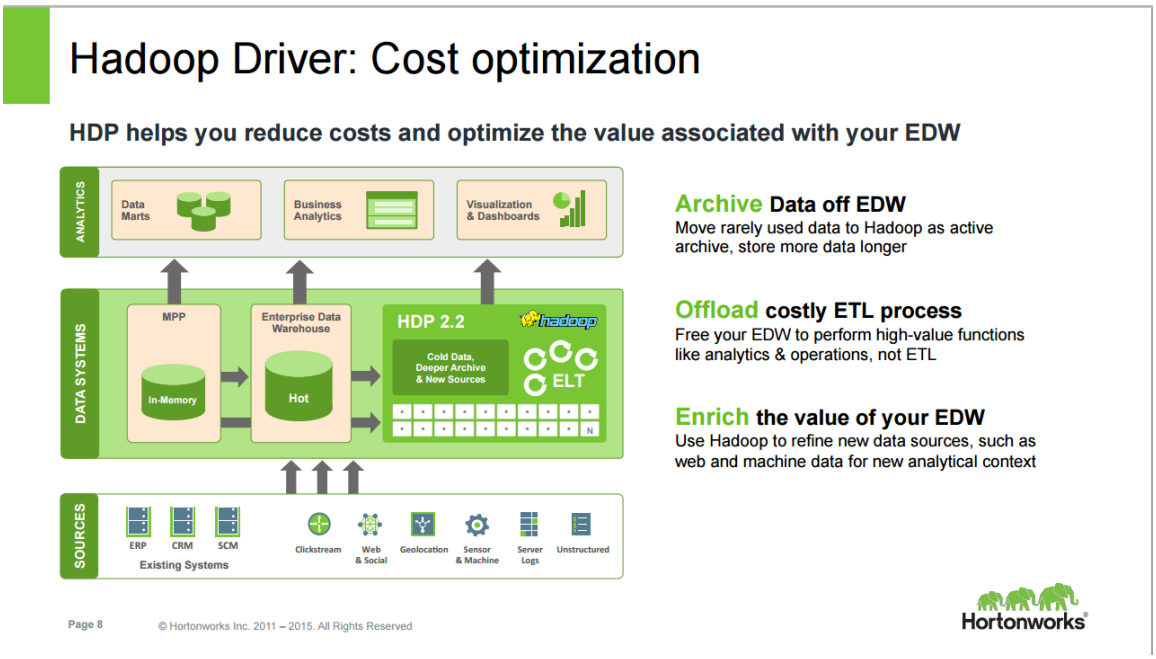
Kuva15. Ennen Hadooppia voimassa ollut arkkitehtuuri.

Hadoop:in suurin tarkoitus on toimia tiedonkeskitettynä varastona, jolle voidaan siirtää raskaita ja kalliita prosesseja. Näin saadaan arkistointi kuluja laskettua, siirrettyä kalliita tiedon transformointi työtä ja rikastettua olemassa olevia työtapoja. Lisäksi kuvan 16 tapaan voimme yhdistellä erinäisiä teknologioita toimimaan erinäisillä vahvuus alueilla, jolloin pystymme yhdistämään esimerkiksi tarpeen nopealle analytiikalle ja laajalle analytiikalle yhdeksi palveluksi käyttäjille.



Kuva 16. Big Datan muuttama yrityskuva

Suurin osa keskusteluista Hadoopista, sekä muista teknologioista joita Big Data puolella työstetään, aloitetaan kertomalla teknologioiden hyödyistä. Kyseessä yleensä ei ole kuitenkaan tarkoitus suoraa korvata vanhoja teknologioita vaan tuoda Hadoop vierelle auttamaan raskaissa kuormissa. Hadoopin pääasiallinen tarkoitus on tarjota työkalut datamassojen hallintaan kustannustehokkaasti. Tällä tavoin se on helppo tuoda olemassa olevien järjestelmien vierelle ja laajentaa näiden toimivuutta ja samalla laskea kustannuksia niin raudassa kuin lisensseissä. Kuvassa 17 koitetaan kuvata kuinka Hadoop tuodaan olemassa olevien järjestelmien vierelle jolla saadaan aikaiseksi kuorman vähennystä erinäisiltä olemassa olevilta ratkaisuilta. Lisäksi Hadoopin laajennettavuus mahdollistaa rakennettaessa tietovarastoa Hadoopin ympärille saadaan suurin osa nykypäivänä kehitettävistä työkaluista Hadooppiin. Näin Hadoop voidaan kytkeä esimerkiksi Verticaan, joka voi toimia täysin Hiven kanssa yhdessä.



Kuva 17. Kuvaa Hadoopin ja vanhanaikaisten tietovarastojen yhteiseloa (Vuorela & Johansson 2015)

8 POHDINTA

Hadoop on laaja-aihe ja tässä opinnäytetyössä ei todellakaan käyty koko teknologia lajitelmaa lävitse. Työssä pyrin tuomaan sen tuomat edut ja miksi sitä on alkujaan lähdetty kehittämään, sekä tärkeimpiä teknologioita ja näiden käyttämistä. Hadoopin ehkä vaikeimpia asioita on kuitenkin sen kehitystahti. Olen jo monta vuotta ollut tekemisissä Hadoopin kanssa ja sen päivittymistahti on niin huimaava, että vaikka se ei tällä hetkellä ole nopein, eikä paras kaikessa mitä se tekee. Sen kehitystahti on vakaa ja hinta on omaa luokkaansa. Sen paras asia ehdottomasti on se että se tuo ennen vain kovalla rahalla saadut asiat pienien ja uusien yrityksien saataville ilman liiallista alkuinvestointia. Samalla se tuo mahdollisuuden tutkia ennen lähes mahdottomalta tuntunutta tietoa. Kaiken tämän lisäksi se on muodostunut standardiksi, johon lähes jokainen järjestelmä nykyään jotenkin kytkeytyy.

Työssäni olisin voinut myös syventyä paljon enemmän Hadoop klusterin turvaamiseen tai sen hallitsemiseen. Omaksi tavoitteekseni asettaman Hadoop ymmärryksen saavutuksen kannalta työ luultavasti on tarpeeksi laaja ja käy asioita hyvin lävitse. Työn käyttämistä suoraan myymiseen en voi itse suositella, koska työstä puuttuu esimerkiksi tuotteiden vertailua ja näiden erojen läpi tuontia. Tämän lisäksi työssä olisi voitu käydä useampia komponentteja tai jopa näiden käyttöesimerkkejä enemmän.

Rakenteeltaan työssä olisi voinut myös vähentää teknologioiden kertomista, jos kyseessä olisi ollut teknisemmälle yleisölle kirjoitus. Tavoitteenani oli tarjota helposti luettava materiaali niin itselleni kuin muille siitä mitä Hadoop on alkujaan. Olisin myös halunnut lisätä enemmän esimerkkejä siitä miten työtä oikeasti tehdään eri työkaluilla, esimerkiksi Hive:n avulla sijaintien piirtäminen kartalle Excelissä tai Trafodionin avulla ohjelman ajaminen.

Työssä kuitenkin on mielestäni paljon hyvääkin. Sen sijaan että olisin keskittynyt kaikkiin tunnettuihin teknologioihin, onnistuin tuomaan mukaan muutaman uuden

teknologian joihin suurin osa alalla olijoistakaan ei välttämättä ole tutustunut. Tämän arvo on korkea jokaiselle näihin tutustumaan lähtevälle. Työssä on myös paljon helpottavaa termistöä, kuten jo pelkästään NoSQL määritys, jonka näin tarpeelliseksi käytyäni useissa Meetupeissa ja huomattuani, kuinka ihmiset eivät vielä ole sisällä siinä, mitä vaihtoehtoja maailma tarjoaa tiedon tallennukseen. Olen myös käynyt pitkiä keskusteluja eri kantojen hyödyistä.

Näkisin, että tästä parempi markkinatutkimus ehdottomasti olisi kannattavaa ja näyttäisi paremmin ihmisille miksi esimerkiksi Graafiseen teknologiaan pohjautuvat kannat ovat hyviä. Nämä tietokannat ovat luultavasti seuraavaa Hadoop aaltoa. Hadoopilla ei ole vielä olemassa yhtään graafista tietokantaa, joista tietäisin, muutenkin tämä tietokanta malli on vielä alkutekijöissään ja kehittymässä. Tämän vuoksi onkin mukava olettaa, että seuraavia tietokantoja joita Hadoopin päälle alkaa ilmestyä ovat Graafiset tietokannat, jotka tähän saakka ovat luultavasti vältelleet Hadooppia sen hitauden vuoksi. Tätä kuitenkin on koitettu ratkoa laajasti monien eri tekijöiden toimesta.

Tez ja Hiven nopeutus ovat olleet täydellisiä esimerkkejä siitä mitä Hadoop pystyy tekemään. Tämä on tällä hetkellä kehittyvä ala ja tämä vuoksi esimerkiksi Sparkin kehittymisen yhdistäminen Hadoop ympäristöihin on ollut erittäin mielenkiintoista seurattavaa. Tämän myötä Hadoop on siis saanut myös nopeutta analyyseihinsä.

Jätin tahallani työstä asioita pois, joilla hallitaan virtaavaa dataa. Näiden pois jättö johtui lähinnä omasta preferenssistäni tehdä tämä rakenne erilaisilla teknologioilla, jotka toimivat tässä välikerroksessa paremmin kuin tällä hetkellä muut teknologiat, joita olisin voinut kertoa. Sparkin funktionaalisuus on tähän mennessä todettu heikoksi virheiden käsittelyn kannalta. Storm taas on hieman hidas ja hankala käyttää sekä raskas ylläpitää.

Nämä työkalut kuitenkin ovat osa Hadooppia, mutta en nähnyt reaaliaikaisen datan käsittelyn olevan osa tämän työnmääritelmää, koska siitä kirjoittaminen tarkoittaisi, juuri näiden asioiden käsittelyä laajasti. Sqoop ja Flume taas olivat mukana,

koska näillä hallitaan yleensä massiivisia datan sisään tuonteja aina kun mahdollista. Tämä taas ei ole reaaliaikaista, koska kyseessä on enemmänkin silloin tällöin tehtävä ajo.

Käytännönosuudestani olisin voinut myös kirjoittaa paremmin opinnäytetyöhön, mutta sen kertominen olisi luultavasti ollut turhaa tämän työn puitteissa. Hadoopin käyttöön ei ole yhtä oikeaa tapaa vaan jokaisen on löydettävä oikea tapa käyttää Hadooppia heidän yrityksensä ja tapojensa mukaisesti. Suositeltua on kuitenkin lähteä kulujen ja tehojen säästöstä. Tämä onnistuu hyvin siirtämällä raskasta analytiikkaa, joka on aikaisemmin kestänyt kauan Hadoopin päälle, joka hallitsee tämän osion.

Teknologisesti Hadooppiin on myös nykyään helpompi yhdistää asioita, kuten HP Vertica, joka osaa kysellä Hive tauluilta dataa. En kuitenkaan halunnut tähänkään asiaan paneutua opinnäytetyössäni, koska se on koko ajan liikkuva ala, jossa joka yö tulee uusia löytöjä kuinka asioita voidaan yhdistää.

Tyytyväisyyttä työssäni aiheutti se että onnistuin osoittamaan, kuinka Hadoop toimii nykypäivän ketterien työkalujen kanssa ja miten se yhdistää niitä tehdäkseen itsestään tehokkaamman niin pilvessä kuin lokaalisti. Tämä mahdollistaa uuden näkymän virtuaaliseen maailmaan.

Hortonworks on avoimmen lähdekoodin talo, jonka perustaja jäsenistö kehitti Hadoopin aikanaan Yahoolla. Tämän lisäksi Hortonworksin työntekijät johtavat selkeästi Apache Hadoop projektien johtoryhmää kohti miltä tulevaisuuden Hadoopin tulisi näyttää ja mitä siihen tulisi kehittää.

MapR ongelma tässä on pitkälti, että he perustuvat pitkälti omaan arkkitehtuuriliseen valintaansa, joka ei ole avointa lähdekoodia ja sen kestämisestä ei ole vielä varmaa näyttöä. Tällä hetkellä suunta on hyvä, mutta tulevaisuus näyttää olinko väärässä suosiessani Hortonworksin avoimuutta yli MapR:n arkkitehtuurillisen valinnan.

Cloudera toisaalta on myös avoin nykyään. Avainsana nykyään. Cloudera kehitti pitkään sisäisesti projektejaan ja tästä johtuen, heidän siirtymänsä avoimuuteen alkoi Hortonworksin pakotteesta ja he ovat vielä hieman välivaiheessa avoimuuden ja laskutettavan mallin kanssa, kun taas Hortonworksin kanssa on selkeää, miten he tekevät rahansa ja näin ollen dokumentaatio on korkeatasoista ja helppoa luettavaa, sekä helposti seurattavaa.

LÄHTEET

Kirja:

Christopher J. Date 2009. SQL and Relational Theory, O'Reilly

Kristina Chodorow & Michael Dirolf 2010. MongoDB the Definitive Guide, O'Reilly

Pramod J. Sadalage & Martin Fowler 2013. NoSQL Distilled, Addison-Wesley

Tom White 2009. Hadoop the Definitive Guide , O'Reilly

Internet-sivut:

Ansible 2015. How Ansible Works. Haettu 7.7.2015. Sivustolta Ansible internetosoite http://www.ansible.com/how-ansible-works

Apache 2015. Apache Hadoop NextGen MapReduce (YARN). Haettu 17.2.2015. Sivustolta Apache internetosoite http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Apache Ambari 2015. Ambari. Haettu 10.7.2015. Sivustolta Apache Ambari internetosoite http://ambari.apache.org/

Apache Flume 2015. Haettu 13.7.2015. Sivustolta Apache Flume internetosoite https://flume.apache.org/

Apache HBase 2015. Haettu 13.7.2015. Sivustolta Apache HBase internetosoite http://hbase.apache.org/

Apache Hive 2015. Hive. Haettu 10.7.2015. Sivustolta Apache Hive internetosoite http://hive.apache.org/

Apache Mahout 2015. Mahout. Haettu 13.7.2015. Sivustolta Apache Mahout internetosoite http://mahout.apache.org/

Apache Pig 2015. Pig. Haettu 10.7.2015. Sivustolta Apache Pig internetosoite http://pig.apache.org/

Apache Sqoop 2015. Sqoop. Haettu 13.7.2015. Sivustolta Apache Sqoop internetosoite http://sqoop.apache.org/

Apache Spark 2015. Spark. Haettu 10.7.2015. Sivustolta Apache Spark internetosoite https://spark.apache.org/

Apache Tez 2015. Introduction. Haettu 10.7.2015. Sivustolta Apache internetosoite http://tez.apache.org/index.html

Betts Ryan 2014. NewSQL Cake You Can Eat. Haettu 7.7.2015, Sivustolta VoltDB internetosoite http://voltdb.com/blog/newsql-cake-you-can-eat

Big Data Vendors 2013. Big Data Vendors. Haettu 10.7.2015 Sivustolta http://bigdatavendors.com/top.php

Cloudera 2015a. An Operational Data Store Built for the World of Big Data. Haettu 17.2.2015. Sivustolta Cloudera internetosoite http://www.cloudera.com/content/cloudera/en/solutions/enterprise-solutions/operational-datastore.html

Cloudera 2015b. Introducing Hue. Haettu 13.7.2015. Sivustolta Cloudera internetosoite http://www.cloudera.com/content/cloudera/en/documentation/cdh4/v4-2-2/Hue-2-User-Guide/hue2.html

Cloudeta 2015c. Cloudera enterprise. Haettu 20.7.2015. Sivustolta Cloudera internetosoite http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise.html

Dhruba Borthakur 2008. HDFS Design. Haettu 17.2.2015. Sivustolta Apache internetosoite http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Docker 2015. What is Docker? Haettu 17.2.2015. Sivustolta Docker internetosoite https://www.docker.com/whatisdocker

Eficode 2015. Ainoa Big Data–kumppani, jonka tarvitset. Haettu 10.7.2015. Sivustolta Eficode internetosoite http://eficode.fi/palvelut/big-data/

Experfy Editor 2014. Cloudera vs Hortonworks vs MapR: Comparing Hadoop Distributions. Haettu 10.7.2015 sivustolta http://www.experfy.com/blog/cloudera-vs-hortonworks-comparing-hadoop-distributions/

Kalle Sirkesalo ja Teppo Kauppinen 2015. Github. Haettu 8.9.2015 sivustolta https://github.com/eimink/vsaa-server-nodejs/blob/master/schema/MySQL.md

Hortonworks 2014. Why Hortonworks. Haettu 17.2.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/why-hortonworks/

Hortonworks 2015a. Apache Spark. Haettu 10.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/spark/

Hortonworks 2015b. Apache Pig. Haettu 10.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/pig/

Hortonworks 2015c. Apache Ambari. Haettu 10.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/ambari/

Hortonworks 2015d. Apache Hive. Haettu 10.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/hive/

Hortonworks 2015e. Apache Sqoop. Haettu 13.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/sqoop/

Hortonworks 2015f. Hortonworks HBase. Haettu 13.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/hbase/

Hortonworks 2015g. Hortonworks Flume. Haettu 13.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/flume/

Hortonworks 2015h. Hortonworks Mahout. Haettu 13.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/mahout/

Hortonworks 2015i. Hortonworks CloudBreak. Haettu 13.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hadoop/cloudbreak/

Hortonworks 2015j Hortonworks HDP. Haettu 20.7.2015. Sivustolta Hortonworks internetosoite http://hortonworks.com/hdp/

IBM 2015. What is MapReduce. Haettu 17.2.2015. Sivustolta IBM internetosoite http://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/ Viitattu 17.2.2015

Lo Frank 2015. Big Data Technology. Haettu 29.6.2015 Sivustolta DataJobs internetosoite https://datajobs.com/what-is-hadoop-and-nosql

MapR 2015. MapR Distribution. Haettu 18.2.2015. Sivustolta MapR internetosoite https://www.mapr.com/products/mapr-distribution-including-apache-hadoop

Oracle 2014. Oracle. Haettu 16.2.2015. Sivustolta Oracle internetosoite http://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

Pasi Vuorela & Mats Johansson 2015. Hadoop Meetup. Haettu 11.9.2015 internetosoite http://files.meetup.com/18217262/Meetup%20Helsinki%2020150316.pdf

SAS 2015. Big Data What it is & Why it matters. Haettu 18.2.2015. Sivustolta SAS internetosoite http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

Stack Michael 2015. Trafodion Apache Incubator Proposal. Haettu 13.7.2015. Sivustolta Apache Wiki http://wiki.apache.org/incubator/TrafodionProposal

Vangie Beal 2015. Webopedia. Haettu 16.2.2015. Sivustolta Webopedia internetosoite http://www.webopedia.com/TERM/S/SQL.html

Waterline 2015. Product Overview. Haettu 14.7.2015 Sivustolta Waterline internetosoite http://www.waterlinedata.com/prod