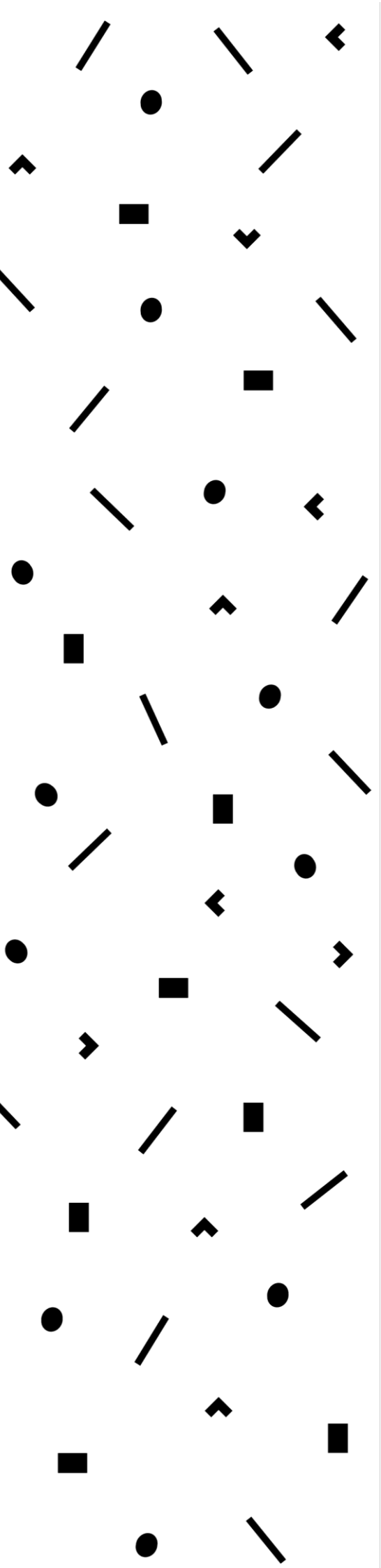


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Big datan analysointi

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2017
Mikael Jantunen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

JANTUNEN, MIKAEL: Big datan analysointi

Ohjelmistotekniikan opinnäytetyö, 50 sivua

Kevät 2017

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tutkia Big datan analysointia avoimen datan avulla sekä sen visualisointiin käytettäviä työkaluja. Työ suoritettiin virtuaalikoneelle asennetussa Apachen Hadoop -ympäristössä. Data työn tekemiseen haettiin Suomen Ilmatieteen laitoksen avoimesta rajapinnasta.

Opinnäytetyön teoriaosuudessa käsitellään yleisesti, mitä Big data on, Apache Hadoop -ekosysteemiä, data-analyysin eri vaiheita sekä R-ohjelmointikieltä. Big datasta käydään läpi sen ominaisuuksia sekä määritelmiä. Apachen Hadoopista kerrotaan sen oleellisimpia osia sekä niiden toimintaperiaatteita. Työssä käydään myös lyhyesti läpi Microsoft Excelin, Microsoft Power BI:n sekä Hadoop Huen käyttöä visualisointityökaluina.

Opinnäytetyön käytännön osuudessa käytiin läpi datan hakeminen, käsittely ja visualisointi. Datana käytettiin Ilmatieteen laitoksen avointa dataa Lahdesta, mikä sisälsi esimerkiksi lämpötiloja ja lumen syvyyksiä. Datan hakemiseksi ohjelmoitiin kaksi Java-ohjelmaa, joista toinen haki datan Ilmatieteen laitoksen palvelimelta ja toinen muutti sen käsiteltävämpään muotoon. Data käsiteltiin Apache Hadoopin ja R-ohjelmointikielen yhdistävällä RHadoop-paketilla, jossa MapReducen avulla laskettiin saadulle datalle päivittäinen keskiarvo. Tämän jälkeen dataa vielä visualisoitiin.

Asiasanat: Apache Hadoop, R-ohjelmointikieli, RHadoop, MapReduce, Big Data

Lahti University of Applied Sciences
Degree Programme in Information Technology

JANTUNEN, MIKAEL: Big data analysis

Bachelor's Thesis in Software Engineering, 50 pages

Spring 2017

ABSTRACT

The objective of this thesis was to study big data analysis using open data and also to examine the tools used for visualizing big data. A virtual machine with the Apache Hadoop environment was used to achieve this. The data was collected from the Finnish Meteorological Institute's open data API.

The theory part of the thesis deals with what big data is, what the Apache Hadoop ecosystem is, the different steps of data analysis and the R - programming language and environment. There is also a brief overview of how Microsoft Excel, Microsoft Power BI and Hadoop Hue can be used as visualization tools.

The practical part of the thesis explains the collecting of the data, its manipulation and visualization. The data from the Finnish Meteorological Institute contained values for example for temperature and snow depth in Lahti. To collect the data, two Java programs were made. One was used for collecting the data and the other converted the data for easier handling. The data was handled in RHadoop, which is a package to use R-language with Apache Hadoop's Mapreduce-operations. This was used to calculate the daily mean values for the data. After that the data was visualized.

Key words: Apache Hadoop, R-language, RHadoop, MapReduce, Big Data

SISÄLLYS

1 JOHDANTO

Big datan merkitys on kasvanut viime vuosina. Yhä useampi yritys käyttää big datasta analysoitua tietoa omien työprosessiensa ja asiakaspalvelunsa parantamiseen.

Opinnäytetyössä tavoitteena on analysoida säädataa ja samalla tutkia avoimen datan sopivuutta big datan tutkimiseen. Työn toteuttamiseksi pystytetään Hadoop-ympäristö virtuaalikoneelle, jossa datan käsittely tapahtuu. Datan noutamiseen tehdään oma ohjelma, joka hakee datan palvelimelta ja muuttaa sen käsiteltävämpään muotoon.

Tarvittava data haetaan Suomen Ilmatieteen laitoksen tarjoamasta avoimen datan rajapinnasta. Rajapinta vaatii ilmaista rekisteröitymistä, jonka jälkeen käyttöönsä käyttäjä saa henkilökohtaisen API-avaimen.

Työn tekemiseen käytetään Centos 7.0 -käyttöjärjestelmäalustalle asennettua Apache Hadoop 2.7.2 -versiota sekä R-kielen alustaa. Java-ohjelmoinnissa käytetään Eclipse Mars -versiota sekä sille asennettua Hadoop MapReduce -lisäosaa. R-kielen ohjelmointiin käytetään RStudiota sekä R-kielen ja Hadoopin integraatioon käytettävää RHadoopia.

Työssä käydään läpi, mitä Big Data on, Apache Hadoop -ekosysteemiä ja sen toimintaa, datan analysointiin tarvittavia vaiheita sekä R-kieltä ja ympäristöä. Työssä käydään myös lyhyesti läpi Microsoft Excelin, Microsoft Power BI:n ja Hadoop Huen käyttöä datan visualisointityökaluina.

2 BIG DATA

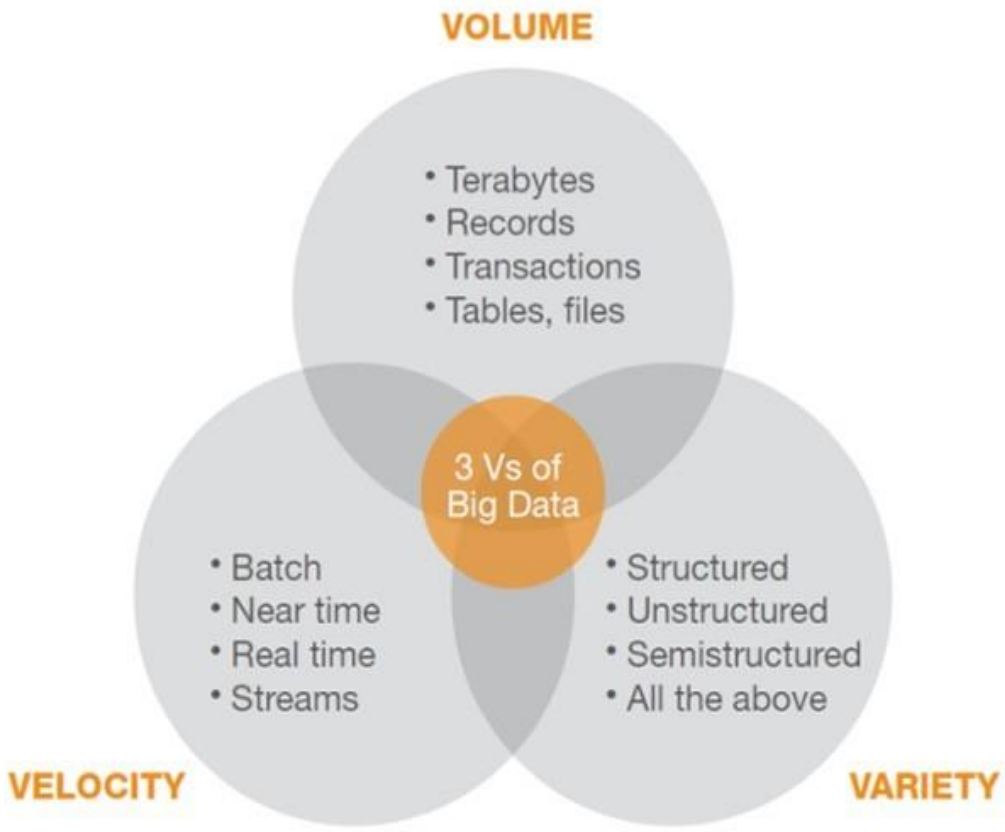
”Big Data” -termiä on yritetty määritellä lukuisilla eri tavoilla, mutta sille ei kuitenkaan ole annettu yhtä määritelmää. Termin merkitys muuttuu sen mukaan, mikä on määrittelijän tausta ja intressi. Oxfordin englannin sanakirja määrittelee big datan seuraavasti:

> Äärimmäisen suuri datajoukko, jota voidaan analysoida laskennallisesti paljastaen kuvioita, trendejä ja assosioita, eritoten liittyen ihmiskäyttäytymiseen ja –kanssakäymiseen (Oxford 2013).

Merrian Websterin sanakirjasta löytyy seuraavanlainen määritelmä:

> Dataa, joka on liian suurta ja monimutkaista tavanomaisien tietokantatyökalujen käsiteltäväksi (Merrian-Webster 2014).

Big Datan käsitteellä voidaan viitata kahteen asiaan. Ensimmäinen on nopeasti kasvavan ja monipuolistuvan datan määrän luoma haaste organisaatiolle ja yhteiskunnalle. Toinen on tähän haasteeseen tarjottavat ratkaisut. (Salo 2013, 10.)



KUVIO 1. Big datan kolme v:tä (Diadmin 2015)

Kuviossa 1 esitetään yleisin tapa kuvata big data ominaisuuksia. Se tapahtuu kolmen V:n avulla

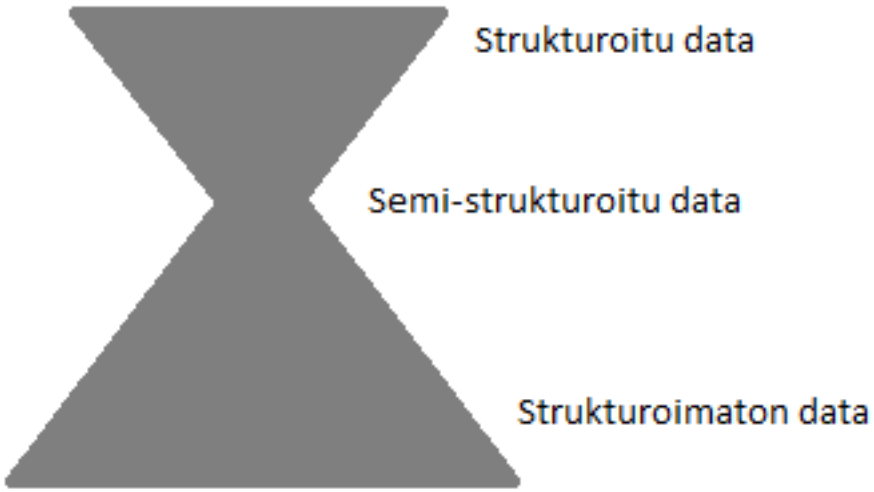
- Volyymilla (Volume) tarkoitetaan datan määrää. Dataa kertyy koko ajan maailmalla eksponentiaalisesti lisää eri muodoissa, kuten kuvina, videoina, datalokeina ja eri sensoreiden keräämänä datana.
- Vauhdilla (Velocity) tarkoitetaan datan kertymisen nopeutta ja kertyneen datan prosessointiin tarvittavaa aikaa. Dataa kertyy koko ajan lisää kiihtyvällä vauhdilla. Esimerkiksi Facebookiin ladataan yli 900 miljoonaa kuvaa joka päivä. (Miller 2015.)
- Vaihtelevuudella (Variety) tarkoitetaan datan monimuotoisuutta. Data ei ole enää rivejä ja sarakkeita, vaan datan muoto vaihtelee sovelluksesta ja käyttötarkoituksesta riippuen. Datalla ei siis ole

> tiettyä rakennetta tai muotoa, vaan suurin osa siitä on strukturoimatonta eli rakenteetonta. (Gewirtz 2016.)

Näiden lisäksi on myös olemassa lukuisia muita big dataan liittyviä ominaisuuksia, kuten veracity, volatility ja value.

2.1 Datan jaottelu

Dataa voidaan jaotella useilla eri tavoilla. Yksi tunnetuimmista tavoista jaotella dataa eri tyyppeihin on jakaa se strukturoituun ja strukturoimattomaan (Salo, 2013, 25). Kuvio 2 havainnollistaa datan jakautumista. Suurin osa datasta on strukturoimatonta. Strukturoitu data voidaan ajatella olevan suoraan tietokoneen käsiteltävänä olevaa dataa, kun taas strukturoimattoman käsittelyyn tarvitaan ihmislogiikkaa tai edistynyttä tekoälyä avuksi (BrightPlanet, 2012).



KUVIO 2. Strukturoitu, semi-strukturoitu, strukturoimaton data (Salo 2013, 23)

TAULUKKO 1. Esimerkki strukturoidusta datasta

Date	Temperature	Windspeed
2012-01-01T03:00:00Z	-4,4	0
2012-01-01T06:00:00Z	-8,3	0
2012-01-01T09:00:00Z	-9,5	0
2012-01-01T12:00:00Z	-6,3	0
2012-01-01T15:00:00Z	-5,3	0,9
2012-01-01T18:00:00Z	-5,6	1,5
2012-01-01T21:00:00Z	-5,2	0
2012-01-02T00:00:00Z	-3,1	1,8

Strukturoidulla datalla on olemassa tietty rakenne, joka sopii hyvin nykyisiin relaatiotietokantarakentaisiin. Strukturoitu data voidaan esittää nimettyjen sarakkeiden ja rivien avulla. Taulukossa 1 data on jaoteltu päivämäärän, lämpötilan ja tuulen nopeuden mukaan. Strukturoidun datan varastointi, prosessointi ja sen lukeminen on heti alussa määritelty. Määrittelyihin kuuluu esimerkiksi datan tyyppi sekä datalle asetetut rajoitukset. (Beal 2017.)

Toinen tyyppi on strukturoimaton data, jolle ei voida määrittää tiettyä rakennetta eikä tämän vuoksi sovi hyvin yhteen nykyisiin tietokantarakenteisiin. Selkeänä erona strukturoituun dataan on se, että strukturoimaton data ei ole tietokoneen suoraan luettavissa, vaan siihen tarvitaan ihmisen apua tai kehittynyttä tekoälyä. Esimerkkeinä strukturoimattomasta datasta ovat videot, kuvat ja äänitiedostot. Myös sähköpostiviestit voidaan laskea strukturoimattomaksi dataksi.

Näiden kahden lisäksi on myös niiden välimuoto, semistrukturoitu data. Avainsanoilla varustettu videomateriaali on esimerkkinä semistrukturoidusta datasta, joissa itse video on strukturoimatonta dataa, mutta siihen liitetyt avainsanat luovat useista videoista koostuvalle datamassalle struktuurin (Salo 2013, 22). Semistrukturoitu data on järjestelty esimerkiksi meta datan avulla, mutta sitä ei ole organisoitu niin

pitkälle, että sen tarkempi tutkiminen ja analysointi olisi mahdollista (Rouse 2014).

2.2 Big Datan käyttökohteita

Big dataa kerääntyy koko ajan kasvavalla vauhdilla lisää, ja yhä useampi yritys käyttää hyväkseen sen tarjoamia mahdollisuuksia. Big dataa voidaan käyttää monella tavalla yrityksen sisäisten asioiden käsittelystä asiakkaiden tarpeiden täyttämiseen ja tieteen edistämiseen.

2.2.1 Yrityksen toiminnan tehostaminen

Asiakkaiden tarpeiden ymmärtäminen ja niihin kohdistuva tarjonta on yksi suurimmista big data -analyysin kohteista. Analysoimalla esimerkiksi mahdollisen asiakkaan selainhistoriaa sekä aikaisempia käyntejä sivustolla voidaan hänelle kohdistaa tuotteita, jotka häntä eniten kiinnostavat. Tarkoituksena on siis löytää ennakoivia malleja, joita hyväksi käyttämällä tarjotaan asiakkaan mahdollisesti tarvitsemia tarvikkeita. (Bernard 2017.)

Yrityksen big dataa analysoimalla voidaan tutkia tuotteiden myyntiä sekä tarvetta liiketjujen eri liikkeissä. Dataa analysoimalla nähdään, jos jonkin tietyn tuotteen myynti on erityisen heikkoa tietyssä liikkeessä, jolloin voidaan tutkia, mistä myynnin heikkous johtuu. Tämän havaitseminen pelkästään ihmisten voimin on vaikeaa. (Bernard 2016.)

Big dataa käytetään yritysten sisäisten ja ulkoisten prosessien optimoimiseen. Yksi suurimmista big data -analyysin käyttötarkoituksista tällä alueella on tarvikkeiden tuonti- ja vientireittien optimointi. Analysoimalla erilaisista lähteistä tulevaa dataa voidaan ajoreittejä optimoida reaaliaikaisesti esimerkiksi liikenteenmäärien mukaisesti. (Bernard 2015.)

Myös yrityksen ulkopuolista dataa analysoimalla voidaan kasvattaa yrityksen liikevoittoa. Esimerkiksi kauppaketju voi analysoida ja ennustaa

tapahtumien tai katastrofien liikkeitä ja lisätä näillä alueilla tarvittavien tarvikkeiden määrää. Säätietoja tarkkailemalla kauppaketju voi ennustaa esimerkiksi sadealueen liikkeen ja lisätä sadevarusteiden määrää kaupoissa, jotka osuvat sadealueen reitille. (Bernard 2016.)

2.2.2 Yksityishenkilö

Big dataa käytetään hyväksi myös yksittäisten henkilöiden taholta. Erilaiset älylaitteet, kuten älykellot ja -rannekkeet, keräävät ihmisestä tietoa unirytmistä kalorin kulutukseen hänen itsensä käyttöön, mutta myös laitteen valmistajalle. Tätä voidaan käyttää antamaan yksittäisille käyttäjille tarkempaa tietoa heidän elämäntavoistaan. (Bernard 2017.)

Myös urheilussa käytetään big dataa hyväksi. Erilaiset sensorit keräävät dataa pelaajien suorituksista ja hyvinvoinnista sekä itse peleistä. Näitä analysoimalla voidaan löytää keinoja suoritusten parantamiseen. (Bernard 2017.)

2.2.3 Tiede ja teknologia

Big data on mahdollistanut uusia tapoja tieteelliseen tutkimukseen. Cernin datakeskuksessa on 65 000 prosessoria analysoimaan 30 petatavua dataa. Cern käyttää tuhansia tietokoneita, jotka on sijoitettu 150 datakeskukseen ympäri maailmaa datan analysoimiseksi. (Bernard 2017.)

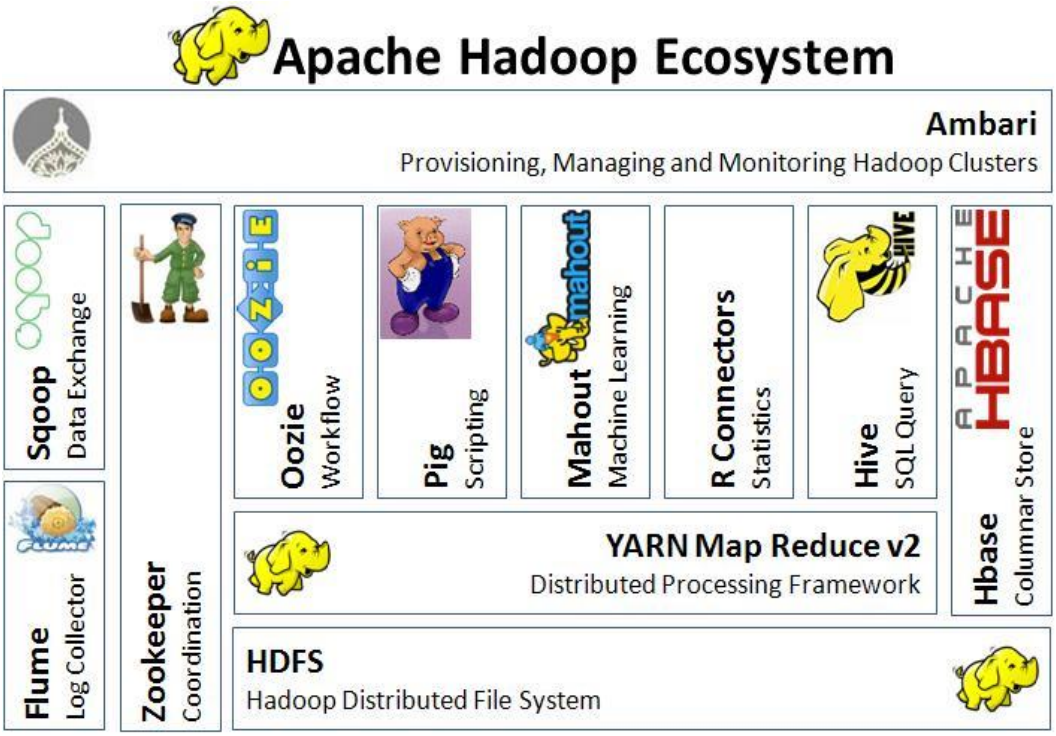
Big data -analyysin laskentatehoa käytetään hyväksi erilaisten tautien parannuskeinojen tutkimustyön edistämisessä. Tutkijat voivat luoda kyselyitä ja kerätä niihin tarvittavaa dataa käyttäjien puhelinten välityksellä. Toinen tapa käyttää big dataa yleisen terveyden seurannassa on seurata ja ennustaa sairausepidemioiden kehittymistä ja liikkumista. Tämä voidaan toteuttaa esimerkiksi keräämällä dataa sosiaalisesta mediasta. (Bernard 2016.)

Big dataa käytetään myös hyväksi laitteiden optimointiin ja suorituskykyyn. Googlen itseajavat autot käyttävät autoon asennettujen sensoreiden ja kameroiden tuottamaa dataa turvallisen ajon takaamiseen ilman ihmisen väliintuloa. (Bernard 2017.) Auton omistaja voi myös saada etukäteen varoituksen mahdollisesta tulevasta viasta.

3 HADOOPIN TOIMINTA

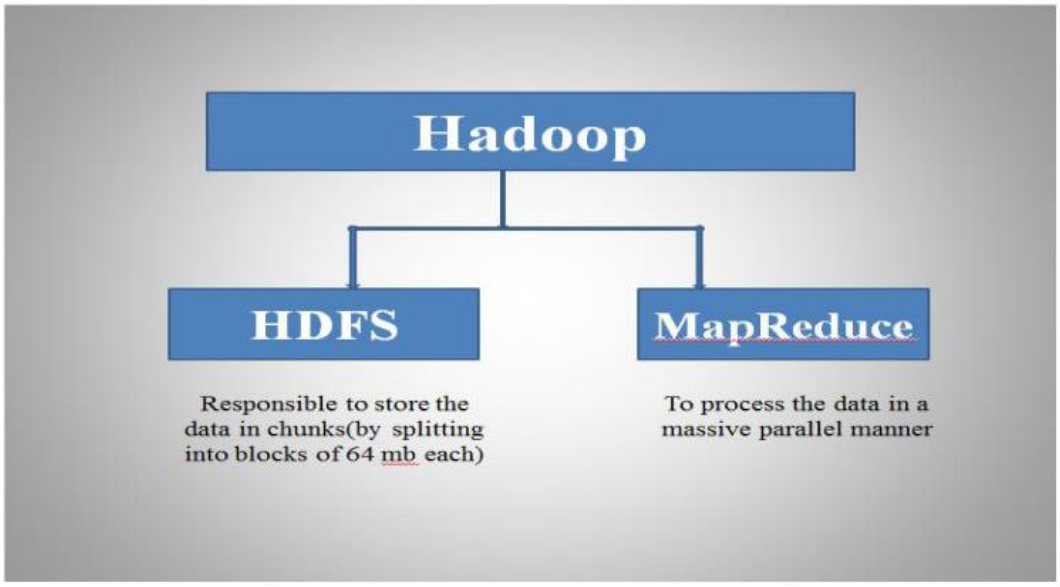
Apache Hadoop on Apache Software Foundationin kehittämä avoimen lähdekoodin ohjelmistokehys luotettavaan ja skaalautuvaan hajautettuun tietojenkäsittelyyn. Hadoop mahdollistaa suurten datamäärien hajautetun prosessoinnin eri tietokoneklustereiden välillä. Se on suunniteltu skaalautuvan yksittäisistä palvelimista (server) tuhansiin laitteisiin, joista jokainen tarjoaa klusterin käyttöön varastointitilaa sekä laskentatehoa. Hadoop-projekti koostuu eri moduuleista. (Apache Software 2017b.)

- Hadoop Common on päämoduuli, joka sisältää kirjastoja sekä ominaisuuksia, joita muut moduulit tarvitsevat (Bappalige 2014).
- Hadoop Distributed File System (HDFS) suomennettuna Hadoop hajautettu tiedostojärjestelmä, on hajautettu tiedostojärjestelmä, joka tarjoaa pääsyn ohjelmiston dataan.
- Hadoop YARN (Yet Another Resource Negotiator) on resurssinhallinta-alusta, jonka vastuuna on klustereiden laskentaresurssien hallinta ja niiden käytön aikataulutus käyttäjien sovellusten käyttöön (Bappalige 2014).
- Hadoop MapReduce on YARNia hyväksi käyttävä järjestelmä rinnakkaiseen dataprosessointiin.



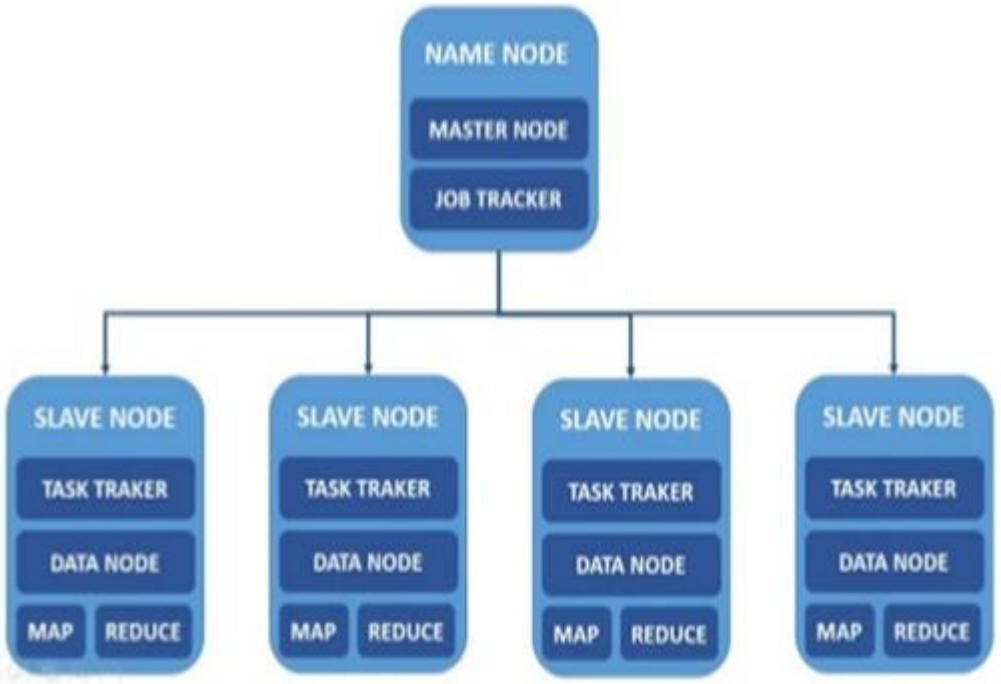
KUVIO 3. Apache Hadoopin ekosysteemi (Kaycee 2016)

Hadoopin ekosysteemi on laaja kokonaisuus, johon kuuluu lukuisia eri projekteja. Nämä projektit toimivat perus Hadoopin ympärillä tuoden työntekoa helpottavia ominaisuuksia. Projekteihin kuuluu esimerkiksi uusien ohjelmointikielien ja ympäristöjen käyttö Hadoopin kanssa, työnkulun (workflow) hallintaa ja käyttöä sekä ylemmältä tasolta Hadoopin eri osien hallintaan. (KUVIO 3.)



KUVIO 4. Hadoopin pääkomponentit (Bigdatajury 2017)

Hadoopin kaksi olennaisinta komponenttia ovat kuviossa 4 esitetyt HDFS ja MapReduce. HDFS:n rooli on datan varastoiminen hajautetusti sekä datan luku- ja kirjoitusmahdollisuudet. HDFS varastoi ja hallinnoi dataa klusterin eri tietokoneiden välillä. MapReducen vastuulla on datan prosessointi käyttäjän määrittelemällä tavalla. Dataa käsitellään rinnakkaisesti koko klusterin sisällä. Nämä kaksi komponenttia on suunniteltu toisistaan riippuvaisiksi.



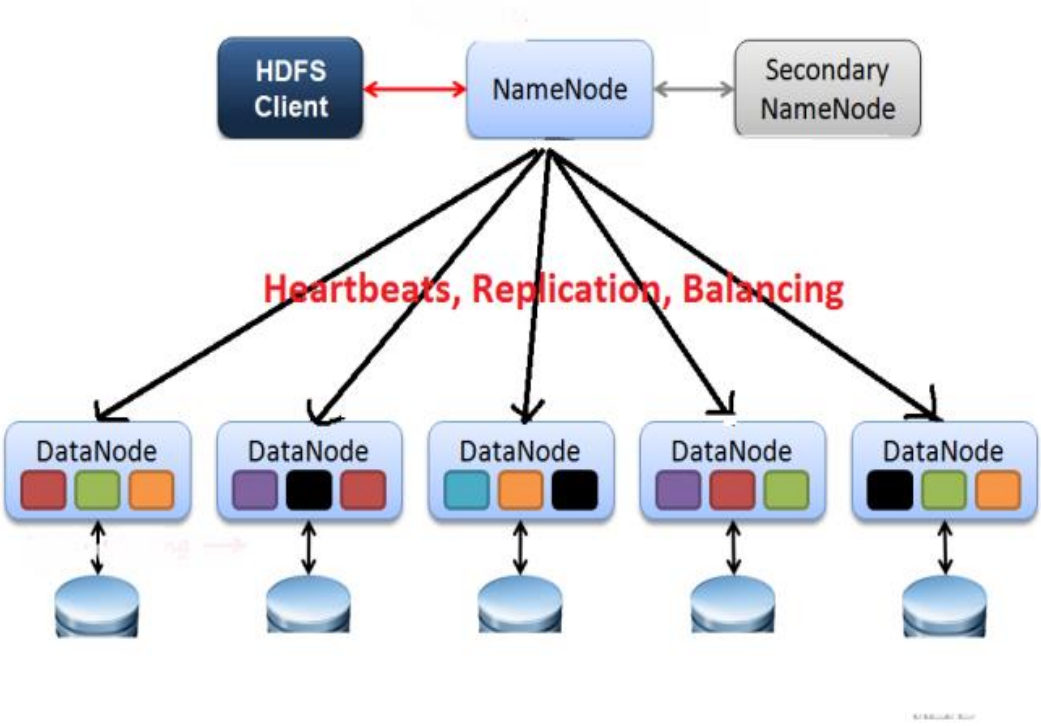
KUVIO 5. Hadoopin arkkitehtuuri (Singh 2013, 16)

Hadoop on mestari/orja arkkitehtuuri (master/slave architecture). Se voidaan jakaa kolmeen eri osaan: asiakaskoneisiin (client), yhteen mestariin ja orjiin (KUVIO 5). Mestari ja orjat määritellään Hadoopin konfiguraatiotiedostoissa (configuration file) antamalla kullekin oma osoite sekä hostname verkossa. Uusien koneiden lisääminen ja poistaminen tapahtuvat näiden asetustiedostojen avulla. Nämä muutokset klusteriin tapahtuvat dynaamisesti, eikä klusteria tarvitse uudelleen käynnistää.

Näiden ulkopuolella on asiakaskoneet, joihin on asennettuna Hadoop, mutta ne eivät toimi kummassakaan roolissa. Asiakaskoneet lataavat dataa klusteriin, lähettävät töitä (job) sekä hakevat ja tarkastelevat töiden lopputuloksia.

3.1 HDFS

HDFS on hajautettu tiedostojärjestelmä, jossa kukin Hadoop-instanssi sisältää yhden nimisolmun (namenode) sekä useita datasolmuja (datanode).

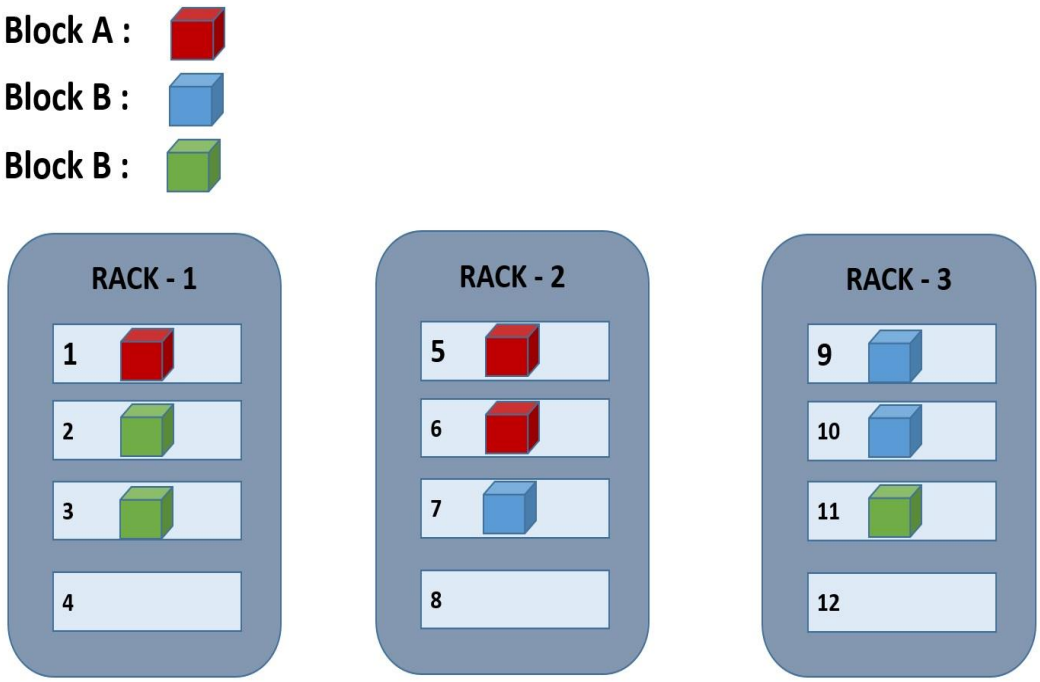


KUVIO 6. HDFS:n toimintaperiaate (Menon 2014)

Namenode sijaitsee master-laitteella, ja se varastoi ja ylläpitää klusterin metadataa. Metadata pitää sisällään tiedostojärjestelmän tiedostojen lohkojen (block) sijainnit klusterissa. Namenoden vastuulle kuuluu datan eheyden seuraaminen sekä osoittaa clientin pyytämän datan lohkojen sijainnit. Namenode saa määräajoin datanodeilta sydämenlyöntejä (heartbeat) sekä joka kymmenes sydämenlyönti sisältää lohkoraportin. Lohkoraportti kertoo namenodelle kaikki sen sisältämät lohkot (Hedlund, 2011.) Kuviossa 6 on esitettynä tämä toimintaketju.

Datanodet taas sijaitsevat orja -koneilla. Datanode pitää sisällään itse datan lohkoina. HDFS pilkkoo datan 64Mb - 128MB lohkoihin (block) ja jakaa sen klusterin tietovarastoihin. Vikasietoisuuden vuoksi kukin lohko

tallennetaan useaan eri paikkaan yhtä aikaa. Lohkojen sijoittelun logiikka on yleensä seuraavanlainen.



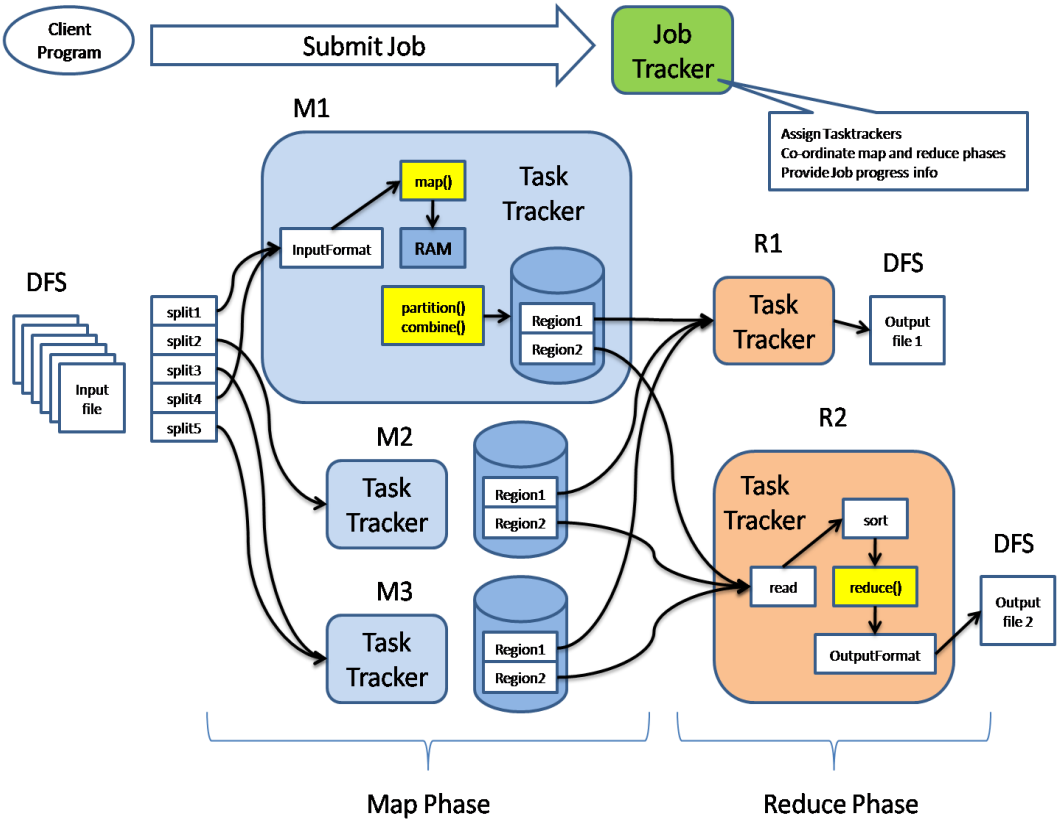
KUVIO 7. Datan jakautumisen logiikka (Hadoopbigdata 2016)

Ensimmäinen kopio sijoitetaan paikalliseen solmuun (local node). Toinen kopio sijoitetaan eri telineeseen (rack). Kolmas sijoitetaan eri nodelle, kuin ensimmäinen kopio, mutta samaan telineeseen. Tämän jälkeen loput sijoitetaan satunnanvaraisiin nodeihin niin että vain yksi kopio on yhdessä nodessa sekä enimmillään kaksi kopiota on samassa telineessä Kuvion 7 mukaisesti. Kuvion 6 tapauksessa 5. ja 6. node ovat ensimmäinen ja kolmas kopio samasta lohkosta. Seuraava kopio sijoitettaisiin joko ensimmäiseen tai kolmanteen telineeseen. (Hadoopbigdata 2016.)

3.2 MapReduce

MapReduce on Hadoopin toinen pääkomponentti. MapReducen tehtävänä on suuren datamäärän käsittely HDFS-klusterissa. MapReducen ideana on se, että klusterin sisällä liikutetaan ohjelmakoodia. Tämä koodi suoritetaan klusterin laitteissa, jotka sisältävät koodilla käsiteltävän datan.

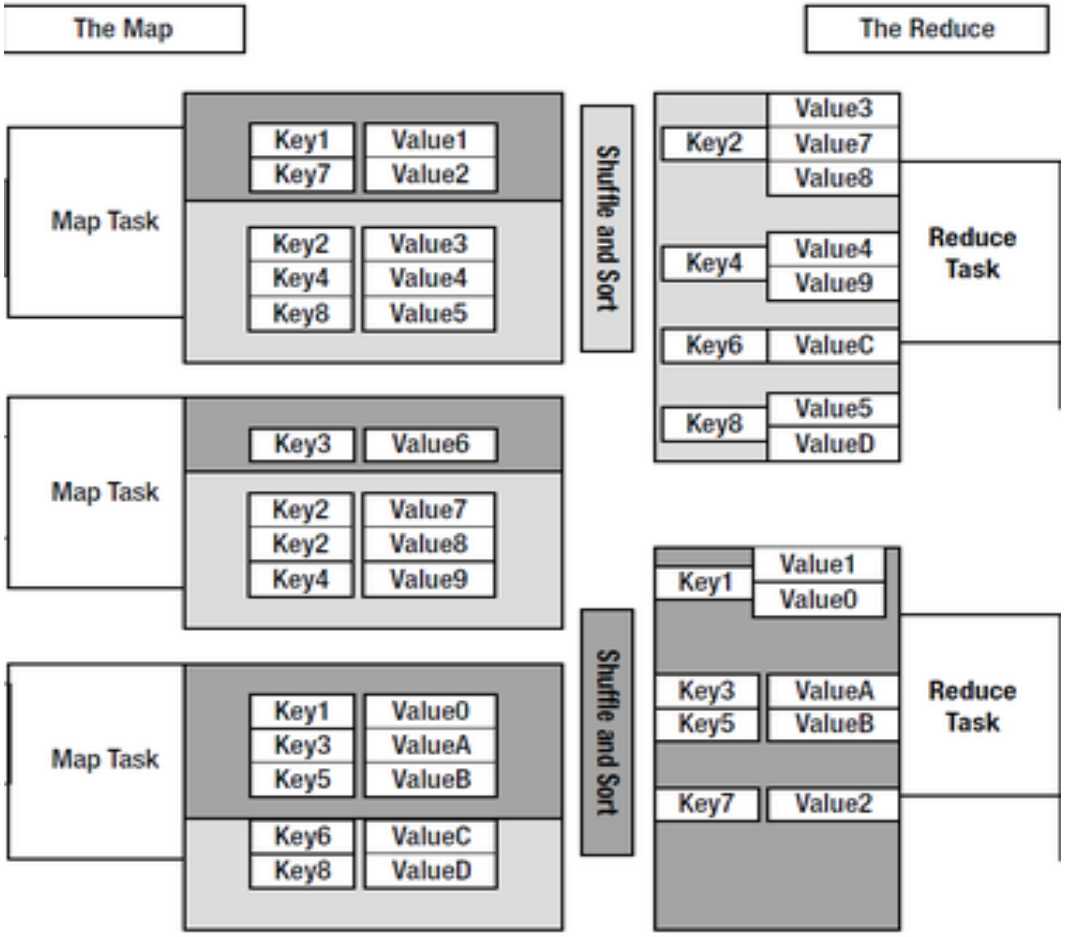
Data pysyy siis paikoillaan. Tämä poikkeaa perinteisestä mallista, jossa data liikkuu koodin luokse. MapReduce toimii työ -periaatteella, jossa asiakasohjelma lähettää sille työn (job), joka menee Job Trackerille. Job Trackerin tehtävänä on valita oikeat Task Trackerit nodeilta ja ohjata saapunut työ niille.



KUVIO 8. MapReducen toimintaperiaate (A4academics 2016)

MapReducessa on kaksi päävaihetta, Map ja Reduce, sekä välivaihde, jonka tehtävänä on datan siirto Map-vaiheelta Reduce-vaiheelle. Kuviossa 8 esitetään MapReducen toimintaperiaate. Datan käsittely MapReducen avulla alkaa, kun asiakaslaite lähettää Job Trackerille MapReduce-työn. Työ on Java-koodi, johon on ohjelmoitu haluttu tehtävä. Job Tracker kysyy Name Nodelta, missä Data Nodeissa sijaitsevat työhön tarvittavien tiedostojen lohkot. Tämän jälkeen Job Tracker lähettää kyseisten Data Nodejen Task Trackereille työn sisältämän Map-javakoodin. Map-koodi suoritetaan paikallisesti Data Noden sisällä käsitellen sille sijoitellut lohkot.

Task Tracker seuraa tehtävien (task) edistymistä sekä raportoi niistä Job Trackerille. (Hedlund 2011.)



KUVIO 9. Avain-arvo -parejen uudelleen järjestäminen (Wikispace 2017.)

Mapista saatu data varastoidaan paikalliseen tiedostojärjestelmään väliaikaisesti. Välivaiheessa Mapista saadulle ulostulo datalle (output data) tehdään sekoitus (shuffle) ja järjestely (sorting). Tässä vaiheessa kaikki tehtävästä saatu data järjestellään avainten perusteella niin, että kaikki samaa avainta vastaavat arvot järjestellään samalle data nodelle kuvion 9 esittämällä tavalla.

Reducer-vaiheessa käsitellään Mapista saadut tulokset. Task Tracker käynnistää Reducer-tehtävän, joka hakee valmistuneilta Map-tehtäviltä

niiden saamat tulokset. Reducerin tehtävänä on koota yhteen Mapeista saadut arvot avainten mukaan ja kirjoittaa saadut tulokset HDFS-tiedostojärjestelmään. Tämän jälkeen data voidaan siirtää paikalliseen tiedostojärjestelmään tarkasteltavaksi.

3.2.1 Map

Map on yksittäinen tehtävä, joka muuntaa annetun avain-arvopari -muotoisen input-datan haluttuun avain-arvopari -muotoon. Avain-arvoparin ei tarvitse olla samaa tyyppiä kuin sisään tullut data. Map-luokalle määritellään neljä parametria. Nämä parametrit ovat

- input datan avaimen tyyppi
- input datan avaimen arvo
- output datan avaimen tyyppi
- output datan avaimen arvo.

```
1  import java.io.IOException;
2
3  import org.apache.hadoop.io.IntWritable;
4  import org.apache.hadoop.io.Text;
5  import org.apache.hadoop.mapreduce.Mapper;
6
7  public class TemplateMapper extends Mapper<Object, Text, Text, IntWritable> {
8
9      public void map(Object input_key, Text input_value, Context context)
10         throws IOException, InterruptedException {
11         // map code here
12
13     }
14 }
```

KUVIO 10. Map-luokan template

Kuviossa 10 rivillä 7 nämä ovat tyyppiä Object, Text, Text ja IntWritable, mutta ne vaihtelevat työhön tarvittavien arvojen mukaisesti. Luokan map-metodi saa kolmanneksi parametrikseen Contextin. Contextiin kirjoitetaan

output-datan avain-arvo -pari kutsumalla context.write -metodia. Context-objektin mahdollistaa Mapperin/Reducerin vuorovaikutuksen muun Hadoop-järjestelmän kanssa. Mapissa sitä käytetään output-datan siirtämiseen Reducerille.

Hadoopin MapReduce muodostaa yhden map-tehtävän jokaista InputFormatille tehtyä InputSplitiä kohden. Nämä tehtävän toimivat hajautetusti klusterin eri laitteissa. Mapin output-data on aina avain-arvopari.

3.2.2 Reduce

Reducerin input-datana on Mapperista saatu data. Reducer käsittelee Mapperissa jokaiselle avaimelle osoitetut arvot. Reducer-vaiheella on kolme päävaihdetta: shuffle, sort ja reduce. Shuffle- ja sort-vaiheet suoritetaan ennen reduce-vaihetta. Jos MapReduce-työlle ei ole määritelty reducer-tehtävää, näitä vaiheita ei suoriteta.

Shuffle on prosessi, jossa data siirretään mappereilta reducereille. Shuffle-vaihe voidaan aloittaa ennen kuin kaikki mapperit ovat valmistuneet. Sort -vaihe ryhmittelee Reducerin input datan avainten mukaan. Reducer-tehtävää kutsutaan jokaiselle avaimelle. Reduce-vaiheessa suoritetaan käyttäjän tekemä reduce-koodi. Reduce-vaiheen jälkeen käsitelty data kirjoitetaan yleensä takaisin HDFS-tiedostojärjestelmään.

```
1  import java.io.IOException;
2
3  import org.apache.hadoop.io.IntWritable;
4  import org.apache.hadoop.io.Text;
5  import org.apache.hadoop.mapreduce.Reducer;
6
7  public class TemplateReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
8
9      public void reduce(Text input_key, Iterable<IntWritable> input_values, Context context)
10         throws IOException, InterruptedException {
11         // reduce code here
12     }
13
14 }
15
```

KUVIO 11. Reduce-luokan template

Reducer-luokan parametrit ovat lähes samat kuin Mapper-luokassa, input key ja value sekä output key ja value. Reduce-metodin parametrit ovat lähes identtiset map-metodiin. Erona map-metodiin on reducen toinen parametri, joka on lista input valueista. Reduce-metodia kutsutaan jokaiselle eri avaimelle.

3.3 Hadoopin tilat ja käyttöönotto

Hadoop voi toimia kolmessa eri tilassa. Nämä tilat ovat Standalone mode, Pseudo-distributed mode sekä Fully distributed mode. Hadoop on oletuksena asennuksen jälkeen Standalone modessa. Hadoopin tilaa voidaan vaihtaa konfigurointi-tiedostoja muokkaamalla.

Standalonen daemonit pyörivät yhdellä nodella paikallisesti yhdessä Java prosessissa. Standalone tila ei ole yhteydessä HDFS-tiedostojärjestelmään vaan käyttää tietokoneen paikallista tiedostojärjestelmää. Tätä tilaa käytetään pääsääntöisesti testaamiseen. (Hadoop Team 2015.)

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>

    <property>
        <name>dfs.name.dir</name>
        <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
    </property>

    <property>
        <name>dfs.data.dir</name>
        <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
    </property>
</configuration>
```

KUVIO 12. Esimerkki hdfs-site.xml asetus-tiedostosta

Pseudo-distributed tila toimii yhdellä nodella, kuten Standalone tila, mutta se simuloi hajautettua järjestelmää. Kukin Hadoop daemon suoritetaan erillisessä Java prosessissa. Tämä tila käyttää HDFS-tiedostojärjestelmää. Tähän tilaan päästään muokkaamalla core-site.xml, hdfs-site.xml ja yarn-site.xml -tiedostoja. Kuviossa 12 on kuva hdfs-site.xml -tiedostosta. Tässä kuviossa määritellään HDFS-tiedostojärjestelmän tiedostojen sijainti paikallisella tiedostojärjestelmällä. Dfs.name.dir-muuttujalla määritellään NameNoden ja dfs.data.dir -muuttujalla DataNoden tiedostojen sijainti.

Fully-distributed tila on Hadoopin varsinainen tila. Tässä tilassa klusterissa on muutamasta nodesta jopa tuhansiin nodeihin laitteita, jotka toimivat yhdessä. Jokaiselle nodelle on asennettu oma Hadoop-ohjelmisto. Yleensä yksi klusterin koneista toimii NameNodena ja toinen kone ResourceManagerina. Nämä ovat klusterin master-koneita, muut laitteet ovat näiden orjina.

```
[hadoop@localhost ~]$ jps
5584 SecondaryNameNode
6705 Jps
6213 NodeManager
5016 NameNode
6074 ResourceManager
5245 DataNode
[hadoop@localhost ~]$
```

KUVIO 13. jps-komennon tulos

Hadoop-klusteri käynnistetään konsolista komennoilla start-dfs.sh ja start-yarn.sh. Start-dfs.sh käynnistää klusterin HDFS-tiedostojärjestelmän. Kun tämä toiminto on suoritettu loppuun, voidaan suorittaa start-yarn.sh. Tämä käynnistää YARNin. Jps -komennolla voidaan tarkistaa, että kaikki tarvittavat daemonit ovat käynnistyneet. Listalla tulisi näkyä NameNode, SecondaryNamenode, DataNode, NodeManager ja ResourceManager kuvion 13 mukaisesti. Myös Hadoopin verkkokäyttöliittymästä näkee, kun klusteri on toiminnassa.

3.4 Hadoopin käyttöliittymä

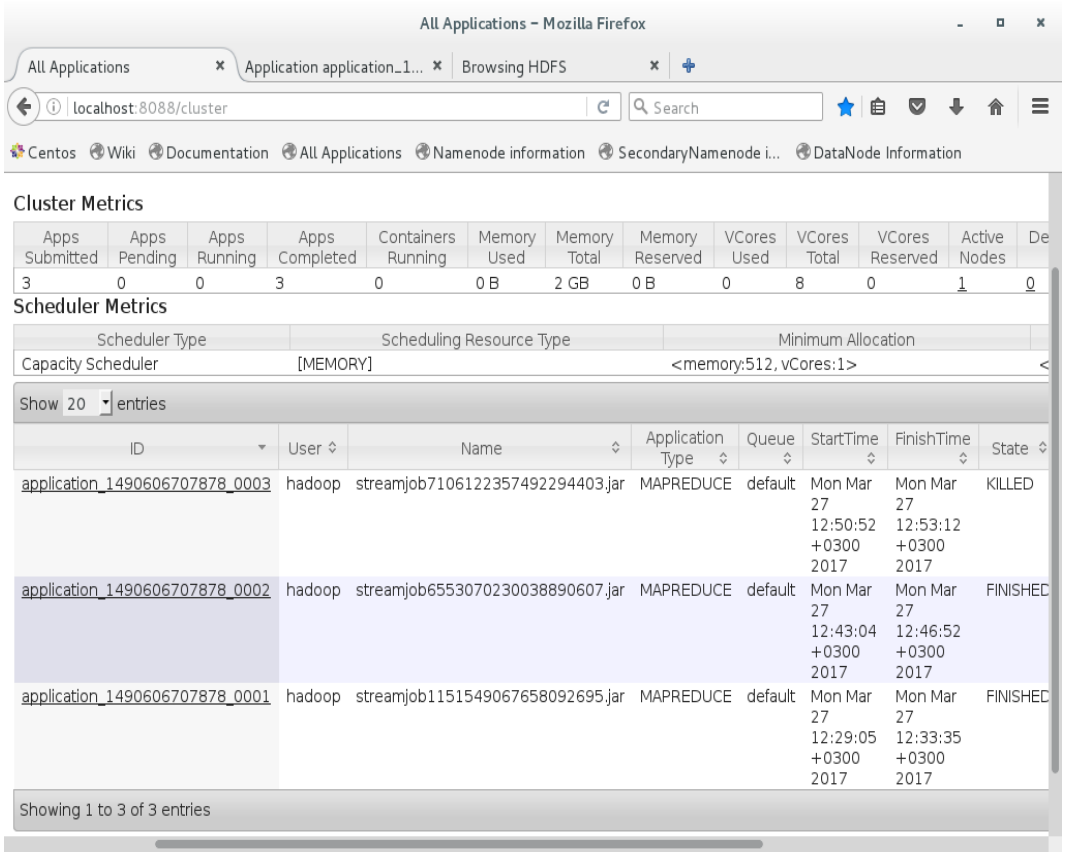
Hadoopin peruskäyttöliittymä on konsolipohjainen. Hadoopin hdfs dfs -komennolla pääsee käsittelemään hdfs-tiedostojärjestelmää. Suurin osa komennoista on samoja kuin perus linux-järjestelmässä. Esimerkiksi kuvion 14 rivillä 1 olevalla hdfs dfs -ls -komennolla voidaan listata tiedostojärjestelmän kansiot. Rivin 7 Hdfs dfs -rm -komento taas poistaa kansion tiedostojärjestelmästä. Komentoja copyFromLocal ja copyToLocal käytetään tiedostojen siirtämiseen paikallisen tiedostojärjestelmän ja HDFS:n välillä.

```
[hadoop@localhost ~]$ hdfs dfs -ls /user/hadoop/averagecalc/input/testdata/
17/04/11 21:30:59 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 hadoop supergroup        210 2017-03-14 21:20 /user/hadoop/averag
ecalc/input/testdata/testcsv.txt
[hadoop@localhost ~]$ hdfs dfs -rm /user/hadoop/averagecalc/input/testdata/testc
sv.txt
17/04/11 21:32:30 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
17/04/11 21:32:40 INFO fs.TrashPolicyDefault: Namenode trash configuration: Dele
tion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/hadoop/averagecalc/input/testdata/testcsv.txt
Did you mean -copyFromLocal?  This command begins with a dash.
[hadoop@localhost ~]$ hdfs dfs -copyFromLocal /home/hadoop/Downloads/tstcsv.txt
/user/hadoop/averagecalc/input/testdata/
17/04/11 21:38:29 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
[hadoop@localhost ~]$ hdfs dfs -cat /user/hadoop/averagecalc/input/testdata/*
```

KUVIO 14. Hadoopin konsolikäyttöliittymä

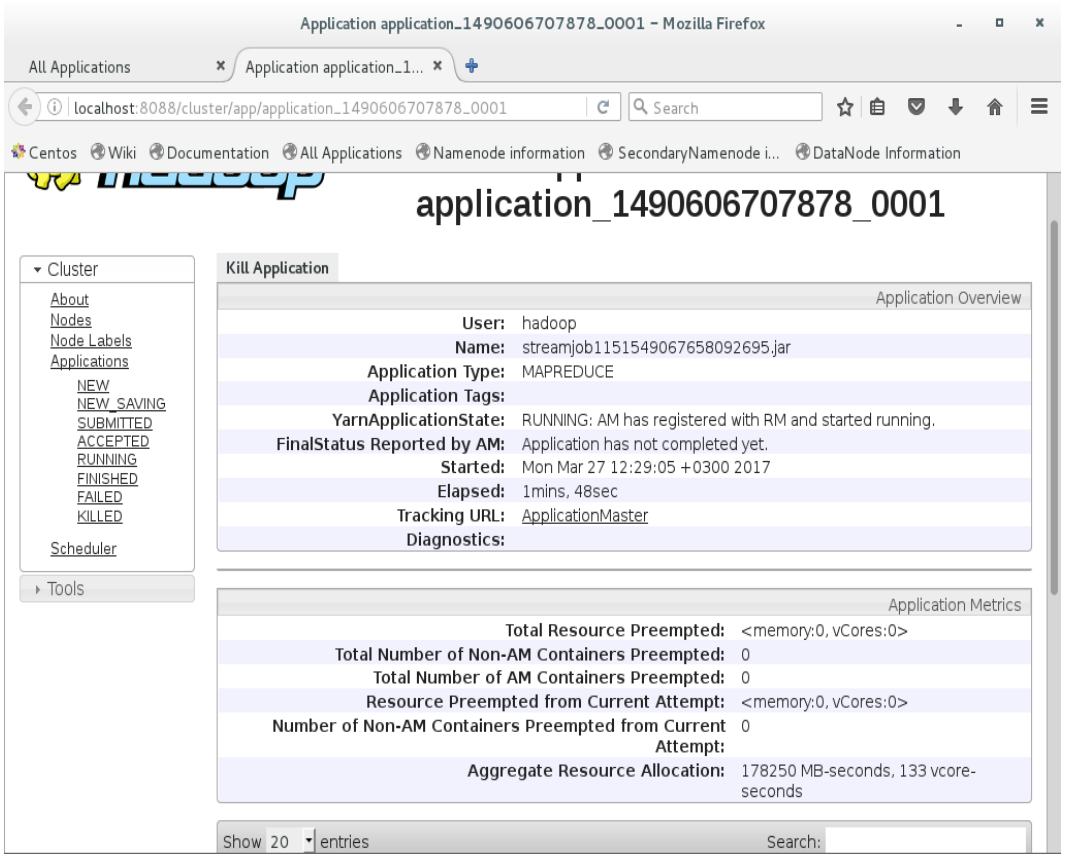
Hadoopin peruskäyttöliittymä on konsolipohjainen. Hadoopin hdfs dfs -komennolla pääsee käsittelemään hdfs-tiedostojärjestelmää. Suurin osa komennoista on samoja kuin perus linux-järjestelmässä. Esimerkiksi kuvion 14 rivillä 1 olevalla hdfs dfs -ls -komennolla voidaan listata tiedostojärjestelmän kansiot. Rivin 7 Hdfs dfs -rm -komento taas poistaa kansion tiedostojärjestelmästä. Komentoja copyFromLocal ja copyToLocal käytetään tiedostojen siirtämiseen paikallisen tiedostojärjestelmän ja HDFS:n välillä.

Hadoop sisältää myös verkkoselaimella hallittavan verkkokäyttöliittymän. Se mahdollistaa klusterin eri statistiikkojen seuraamisen sekä HDFS-tiedostojärjestelmän tiedostojen tarkastelun. Oleellisimmat osoitteet ovat http://localhost:8088 ja http://localhost:50070.



KUVIO 15. Klusterin seuranta sivusto

Osoitteesta http://localhost:8088 löytyy koko klusterin tarkasteluun tarvittavat työkalut ja tilastot. Sieltä näkee klusterin eri statistiikkoja, kuten klusterin muistin tilan, solmujen tilat sekä käynnissä olevat työt ja tehtävät. Sivustolta näkee myös palvelimen metriikoita. Tältä sivulta näkee myös tarkemmin nykyisten ja vanhojen töiden tiedot. Kuviossa 15 on esitetty osa klusterin päänäkymästä, jossa on nähtävissä esimerkiksi kolme viimeisintä työtä.



KUVIO 16. Työn yleistiedot

Käynnissä olevista ja valmistuneista töistä saa myös tarkemmat tiedot, kuten käyttäjän, aloitusajan ja keston. Käynnissä olevista jobeista on myös mahdollista päästä käsiksi sen lokeihin sekä saada tarkemmat tiedot suorituksen sen hetkisestä tilasta, kuten mapin ja reducen etenemisestä. Käynnissä olevan työn voi myös lopettaa sivuston kautta, mikä on yksinkertaisempaa kuin komentorivin käyttäminen. (KUVIO 16.)

Osoitteesta http://localhost:50070 näkee Namenoden tiedot. Täältä saa yleiskuvan Namenoden tilasta. Täällä pääsee myös tarkastelemaan HDFS:n sisältöä sekä lataamaan siellä sijaitsevia tiedostoja verkkoselaimen kautta. Tästä osoitteesta on myös mahdollista päästä käsiksi koko klusterin eri lokitiedostoihin.

Namenoden osoitteesta saadaan yleiskuva klusterin tiedostojärjestelmän toiminnasta. Sieltä näkee klusterin kapasiteetin sekä kuinka paljon siitä on

käytettynä. Yleiskuvaus myös kertoo elävät ja kuolleet nodet. Myös jokaisesta klusterin datanodesta on mahdollista saada tarkemmat tiedot. Jos datanodeissa esiintyy virheitä, ne on listattu täällä. Namenode myös listaa klusterin käynnistymisen etenemisen sekä sen, onko klusteri mahdollisesti turvallisuustilassa (safemode). Turvallisuustila estää klusterin tiedostojen lukemisen ja muokkaamisen.

Tiedostojärjestelmästä on myös mahdollista ottaa read-only snapshotti. Snapshotin voi ottaa joko yksittäisestä kansiopuusta tai sitten koko järjestelmästä. Näitä snapshotteja käytetään yleensä varmuuskopioina, suojauksena käyttäjien virheiden varalta tai hätätilanteesta toipumiseen. Nämä snapshotit on nähtävissä Namenoden osoitteesta. Täältä näkee listattuna kaikki klusterin snapshotit, niiden lukumäärän ja kenelle käyttäjälle ne kuuluvat sekä snapshotille annettu nimi.

Muita osoitteita ovat http:localhost:50090, jossa sijaitsee SecondaryNamenoden yleiskuvaus sekä http:localhost:50075, joka on Datanoden yleiskuvaus.

3.5 Muita Hadoop ekosysteemin projekteja

Näiden lisäksi Apache Hadoop ekosysteemi sisältää muita siihen liittyviä projekteja. Projekteihin kuuluu niin työntekoa helpottavia osia, kuten Hive ja Pig sekä ulkoisten työkalujen integrointia Hadoop ympäristöön. Hadoop ekosysteemin osat voidaan kategorisoida esimerkiksi varastointiin (storage), prosessointiin (prosessing), kyselyihin (querying) ja ulkoiseen integraatioon (external integration).

HBase (Hadoop DataBase) on hajautettu sarakemuotoinen tietokanta (column based database), joka käyttää HDFS: ää perustanaan. HBase mahdollistaa skaalautuvuuden sekä reaaliaikaisen datan käsittelyn. Sen käyttötarkoitus on äärimmäisen suuret tietokantataulut, jotka voivat sisältää miljoonia rivejä ja sarakkeita dataa. (credera 2014.)

TAULUKKO 2. SQL ja HQL lausekkeita

Query

Function	MySQL	HiveQL
Retrieving information	SELECT from_columns FROM table WHERE conditions;	SELECT from_columns FROM table WHERE conditions;
All values	SELECT * FROM table;	SELECT * FROM table;
Some values	SELECT * FROM table WHERE rec_name = "value";	SELECT * FROM table WHERE rec_name = "value";
Multiple criteria	SELECT * FROM table WHERE rec1="value1" AND rec2="value2";	SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";
Selecting specific columns	SELECT column_name FROM table;	SELECT column_name FROM table;
Selecting from multiple tables (Join same table using alias w/"AS")	SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;	SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);

Hadoop Hive tuo SQL:n kaltaisen käyttöliittymän Hadoopiin. Hiven hakukieli, HQL, on samankaltainen relaatiokannoissa käytettävän SQL-kielen kanssa, kuten Taulukosta 2 on nähtävissä. Tämä mahdollistaa nopean siirtymisen SQL-kehittäjästä Hadoop ympäristöön. Hive muuttaa HQL -lausekkeet MapReduce jobeiksi ja suorittaa ne Hadoop klusterissa. (IBM 2017.)

Apache Ambarin tavoitteena on Hadoop-ympäristön hallinnoinnin yksinkertaistaminen. Ambari on helppokäyttöinen web-käyttöliittymä, joka mahdollistaa niin Hadoop klusterin varustamisen, hallinnoinnin ja monitoroinnin. (Apache Software 2017a.)

4 DATAN ANALYSOINTI JA VISUALISOINTI

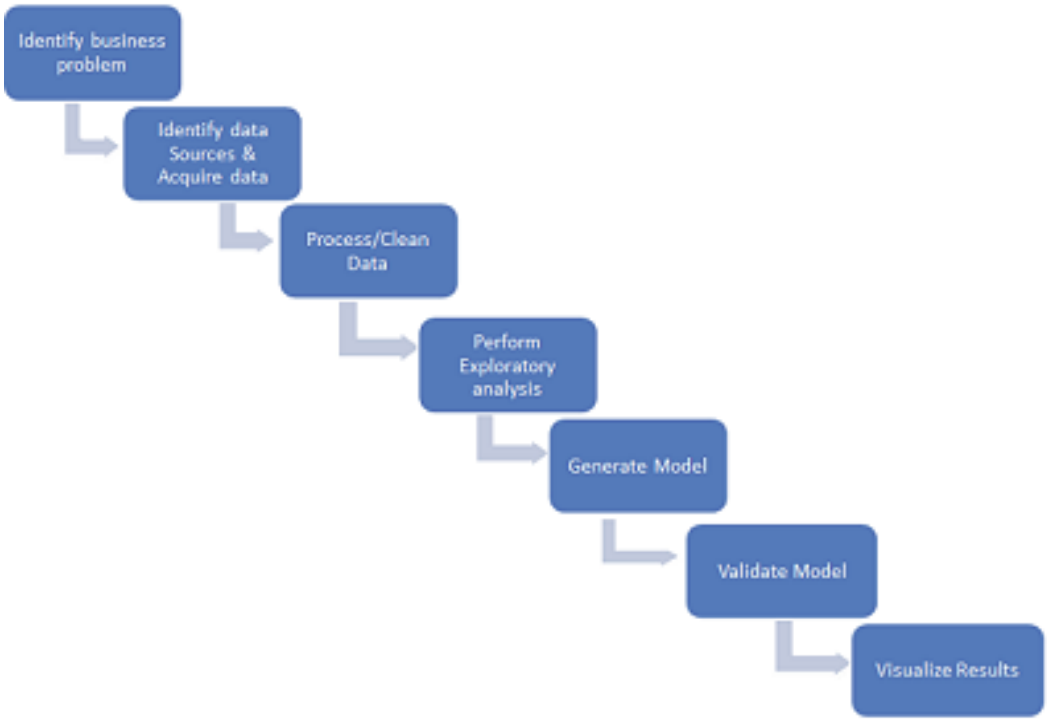
Data-analyysi on datan louhinnan ja Business Intelligencen (BI) pääkomponentti. Data-analyysin tavoitteena on löytää datan seasta sellaista tietoa, joka auttaa yrityksen päätöksenteossa. Eri lähteistä voidaan hankia big dataa, jota analysoidaan erilaisia menetelmiä hyväksikäyttäen.

Data-analyysin avulla pyritään saamaan yrityksen toiminnan kannalta merkityksellistä dataa parempien päätöksien tekemiseen, asiakaspalvelun parantamiseen sekä tuottavuuden kasvattamiseen. Data-analyysin haasteina on datan suuri määrä ja sen käsittely sekä esittäminen ymmärrettävässä muodossa.

4.1 Analysointi vaiheet

Datan analysointi prosessi jaetaan useaan eri vaiheeseen. Pääsääntöisesti nämä vaiheet ovat

- haettavan kysymyksen hahmottaminen ja ymmärtäminen
- mitä dataa analysoidaan ja miten kyseistä dataa analysoidaan
- datan keräys ja käsittely
- datan analysointi
- tulosten tulkitseminen.



KUVIO 17. Esimerkki datan analysoinnin vaiheista (Gorakala 2014)

Datan analysoinnin ensimmäinen ja tärkein vaihe on määritellä haettava ongelma. Datan analysointi alkaa määrittelemällä analysoitavan tehtävän vaatimat kysymykset. Kysymysten tulisi olla mitattavia sekä selkeitä. Tärkeintä on löytää selkeä lähtökohta analysoinnille.

Kun analysoinnin lähtökohdat ovat selvät määritellään, minkälaista dataa tulisi analysoida laadittuihin kysymyksiin vastaamiseksi sekä mistä kyseistä dataa saadaan. Data rajataan sellaiseksi, että saadaan mahdollisimman kattavasti vastattua haettavaan ongelmaan.

Analyysissä käytettävä data pyritään keräämään mahdollisimman monesta lähteestä kattavan analyysin varmistamiseksi. Kun tarvittava data on kerätty käsiteltäväksi, muutetaan se analyysissä käytettävään muotoon. Datan muoto riippuu siitä, millaisilla työkaluilla sitä käsitellään ja millaista dataa se sisältää. Datasta myös pyritään poistamaan kaikki epätavalliset muuttujat kuten tyhjät arvot, käsiteltävän datan rajojen ylittävät arvot ja muut tulosta väärentävät tekijät sekä mahdolliset kirjoitusvirheet.

Kun tarvittavat valmistelut on tehty, datan analysointiprosessi voidaan aloittaa. Datan analysointi voidaan suorittaa erilaisilla menetelmillä, kuten graafisilla esityksillä ja korrelaatioiden etsimisellä. Data analysointiin suunnitellut työkalut ovat hyödyllisiä. Näitä ovat esimerkiksi Visio, Minitab ja Microsoft Excel.

Viimeisenä vaiheena on tutkia toteuttaako saadut tiedot alussa määritellyn ongelman. Saatujen tietojen perusteella päätetään jatkotoimista. On myös mahdollista, etteivät tulokset vastaa odotettua, jolloin prosessi uusitaan muunnelluilla parametreilla.

4.2 R-kieli

R-kieli on avoimen lähdekoodin ohjelmointikieli ja -ympäristö, joka on suunniteltu tilastolliseen laskentaan ja graafiseen esitykseen. R-kieli mahdollistaa erilaisten tilastollisen ja graafisten tekniikoiden monipuolisen käytön. Se on myös hyvin laajennettava erilaisten pakettien (package) vuoksi. (r-project 2017.)

R ympäristönä tarjoaa tehokkaat työkalut datan manipulointiin, laskentaan sekä graafiseen esitykseen. R toimii komentorivi-scriptien avulla. Se mahdollistaa ehtolauseiden sekä silmukoiden käytön, ja modulaarisen ohjelmoinnin funktioiden avulla. R sallii integroinnin C, C++, .Net sekä Pythonilla kirjoitettujen ohjelmien kanssa. (r-project 2017.)

4.2.1 Datatyypit

Toisin kuin muissa ohjelmointikielissä, R:ssä muuttujia ei esitellä tiettyyn datatyyppiin. R:ssä muuttujat ovat R-objekteja, joista tulee muuttujan tarvitsema datatyyppi. R-objekteja on monia eri tyyppisiä, joista käytetyimpiä ovat vektorit, listat, matriisit, taulukot, faktorit ja Datakehykset.

Yksinkertaisin näistä objekteista on vektor-objekti. Näitä atomisia vektoreita on kuusi eri datatyyppiä. Niitä kutsutaan myös vektorin luokiksi.

Nämä vektorin data tyypit ovat looginen, numeerinen, kokonaisluku, kompleksi, merkki (character) sekä raakadata. Nämä kuusi tyyppiä eivät kuitenkaan ole ainoat mahdolliset luokat. Esimerkiksi useasta atomisesta vectorista voidaan muodostaa array, jonka luokaksi muodostuu array. Muut R-objektit on rakennettu näistä atomisista vektoreista.

Lista on R-objekti, joka voi sisältää monia eri tyypin elementtejä sisällään, kuten numeroita, vektoreita, funktioita ja muita listoja. Listan elementeille voidaan antaa nimiä, ja niihin voidaan päästä käsiksi annetuilla nimillä. Listan elementteihin päästään myös käsiksi indeksien avulla. Mitä tahansa listan elementtiä voidaan muokata, mutta lisääminen ja poistaminen tapahtuvat aina listan loppuun. Listoja voidaan myös yhdistää toisiinsa sekä muuttaa vektoreiksi lisäkäsittelyä varten.

Matriisit ovat R objekteja, joiden elementit on järjestetty kaksiulotteiseen nelikulmion muotoon. Matriisin elementit ovat samaa atomista tyyppiä. Matriiseja käytetään yleensä matemaattisten laskutoimitusten suorittamiseen. Matriisin elementteihin päästään käsiksi rivin ja sarakkeen indeksien avulla.

```
vector <- c(1:9)
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("MATRIISI1","MATRIISI2","MATRIISI3")
row.names <- c("RIVI1","RIVI2","RIVI3")
column.names <- c("SAR1","SAR2","SAR3")
matrix.names <- c("MATRIISI1","MATRIISI2")

result <- array(vector,dim = c(3,3,2),
        dimnames = list(row.names,column.names, matrix.names))
```

```
> print(result)
, , MATRIISI1

      SAR1 SAR2 SAR3
RIVI1    1    4    7
RIVI2    2    5    8
RIVI3    3    6    9

, , MATRIISI2

      SAR1 SAR2 SAR3
RIVI1    1    4    7
RIVI2    2    5    8
RIVI3    3    6    9
```

KUVIO 18. Taulukon osien nimeäminen

Taulukot (array) mahdollistavat datan varastoinnin useammassa kuin kahdessa ulottuvuudessa. Taulukot voivat koostua useasta matriisista. Taulukon rivit, sarakkeet ja matriisit voidaan nimetä.

```
> print(flets)
 [1] d c a b a b a d b b a b a b a
Levels: a b c d
> table(flets)
flets
a b c d
6 6 1 2
```

KUVIO 19. Faktorin esitys

Faktorit ovat data objekteja, joita käytetään datan kategorioimiseen ja varastointiin tasoina. Ne voivat sisältää joko merkkijonoja tai numeerisia lukuja. Faktori varastoi kunkin uniikin merkin kerran tasoksi Kuvion 19 ylemmän komennon mukaisesti, ja data itsessään varastoidaan kokonaislukuvektoriin. Faktorin tasot muunnetaan aina merkkijonoksi. Ne ovat hyödyllisiä tilastollisessa esityksessä.

Datakehys (Data Frame) on taulu tai kaksiulotteisen arrayn kaltainen rakenne, jonka jokaisessa sarakkeessa on yhden muuttujan arvot sekä jokaisella rivillä yksi sarja arvoja jokaisesta sarakkeesta. Datakehyksen piirteisiin kuuluvat seuraavanlaiset ominaisuudet

- Sarakkeiden nimet eivät ole tyhjiä.
- Rivien nimet ovat uniikkeja.
- Varastoitu data on numeerinen, faktori tai merkki-tyyppiä.
- Jokaisella sarakkeella on yhtä monta data alkiota.

4.2.2 Muuttujat

R-kielen muuttujat voivat pitää sisällään atomisen vektorin, ryhmän atomisia vektoreita tai yhdistelmän eri R-objekteja. Muuttujan nimi on vapaamuotoisempi kuin useimmissa muissa ohjelmointikielissä. Muuttujan nimi voi pitää sisällään numeroita, kirjaimia ja pisteitä tai alaviivoja.

TAULUKKO 3. Muuttujan nimisäännöt

Variable Name	Validity	Reason
var_name2.	valid	Has letters, numbers, dot and underscore
var_name%	Invalid	Has the character '%'. Only dot(.) and underscore allowed.
2var_name	invalid	Starts with a number
.var_name , var.name	valid	Can start with a dot(.) but the dot(.)should not be followed by a number.
.2var_name	invalid	The starting dot is followed by a number making it invalid.
_var_name	invalid	Starts with _ which is not valid

Taulukosta 3 näkyy sallitut muuttujien nimet. Kunhan muuttuja ei ala numerolla, alaviivalla tai pisteellä jonka perässä on numero, tai siinä ei ole kiellettyjä merkkejä, on sen nimi sallittu. Muuttujan nimellä ei myöskään ole pituus rajoitusta.

Muuttujan arvon asetus voidaan tehdä kolmella eri merkintä tavalla. Arvo voidaan asettaa =-operaattorilla, tai nuolioperaattoreilla <- ja ->. Nämä kaikki tarkoittavat samaa, joten merkintä on ohjelmoijasta kiinni. R-kieli on dynaamisesti tyypitetty kieli. R-kielessä muuttujalle ei aseteta datatyyppiä, vaan se on R-objekti, joka muuttuu tarvittavaan tyyppiin. Tämä mahdollistaa sen, että samaa muuttujan data tyyppiä voidaan vaihtaa yhä uudelleen ohjelman sisällä. Muita tämän kaltaisia kieliä ovat JavaScript, Python ja PHP.

4.2.3 R-paketit

R-paketit ovat kokoelma R-funktioita, dataa ja käännettyä koodia. Ne ovat R-yhteisön tekemiä ilmaisia koodikirjastoja. Ne asennetaan library-kansion alle R-ympäristössä. Oletuksena R asentaa muutaman paketin valmiiksi. Lisää paketteja voidaan asentaa myöhemmin tarpeen mukaan. Paketit, jotka on asennettu näin, täytyy erikseen ladata ohjelmaan, jossa niitä

aikoo käyttää. Tällä hetkellä CRAN-pakettivarastossa (repository) on 10331 pakettia.

```
1
2  install.packages("ggplots2", lib="/data/Rpackages/")
3
4  install.packages("ggplots2")
5
6  install.packages("ggplots2",
7                   repos="http://cran.revolutionanalytics.com")
8
9  install.packages(c('functional','stringr','plyr'),
10                  repos='http://cran.revolutionanalytics.com')
11 library(ggplots2)
```

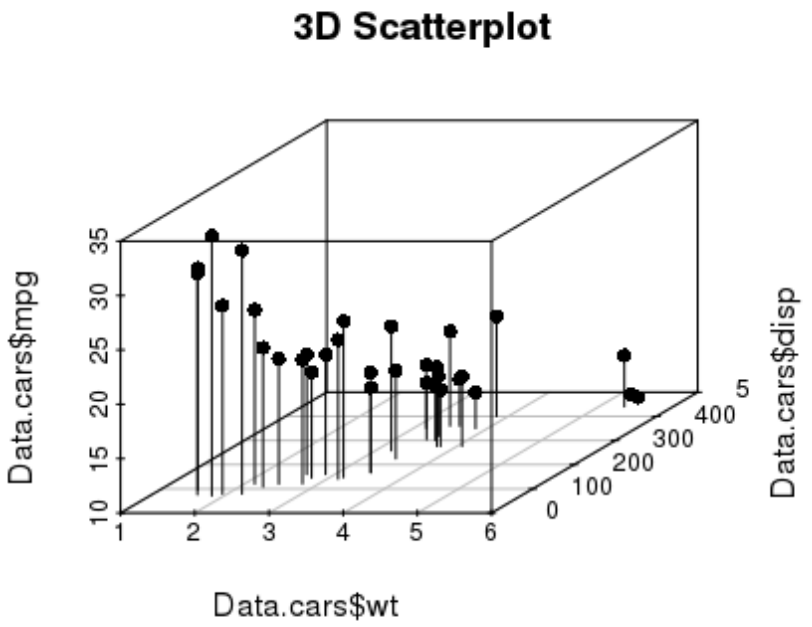
KUVIO 20. R-pakettien asennus

Pakettien asentaminen on yksinkertainen prosessi. Paketti voidaan ladata joko paikalliselle asemalle ja sieltä asentaa R-ympäristöön, kuten kuviossa 20 rivillä 2 näkyy, tai sitten ladata suoraan CRAN-pakettivarastosta rivin 4 mukaisesti. Jos käyttäjällä ei ole root-oikeuksia käyttöjärjestelmään, täytyy paketit asentaa paikalliselta asemalta komennolla. Paketin asennusvaiheessa on mahdollista määrittää pakettivaraston sijainti rivin 6 mukaisesti. Paketteja on myös mahdollista ladata useampia kerralla antamalla nimet parametri vektorina kuten rivillä 9 näytetään. Paketin käyttöönotto tapahtuu yksinkertaisesti rivin 11 komennolla.

4.2.4 Kuviot

R-kielessä on myös lukuisia kirjastopaketteja erilaisten kaavioiden ja graafien esittämiseen. R-yhteisöstä löytyy paketteja lukuisten eri tyyppisten graafien muodostamiseen. Näitä graafeja ovat esimerkiksi piirakkamallit, histogrammit ja pylväsdiagrammit.

```
> library(scatterplot3d)
> Data.cars = mtcars
> scatterplot3d(Data.cars$wt,Data.cars$disp,Data.cars$mpg,
pch=16, hightlight.3d=TRUE,type="h",main="3D Scatterplot")
```

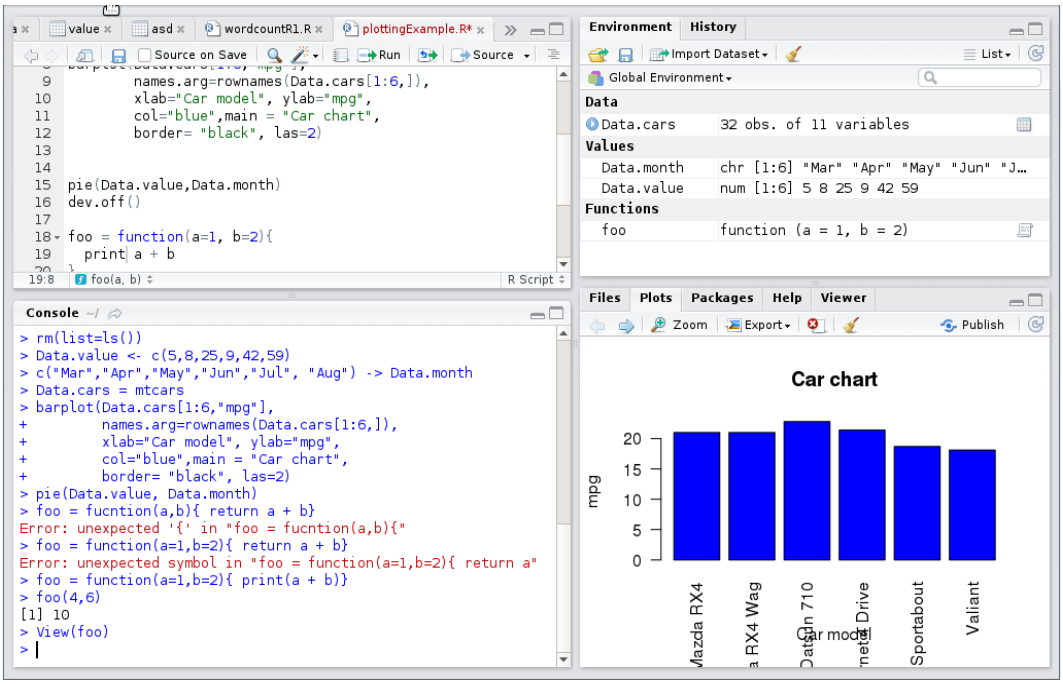


KUVIO 21. Pistekaavion muodostus ja pistekaavio

Kuviossa 21 on muodostettu pistekaavio R:ssä. Scatterplot3d on paketti, joka ladataan komennolla install.packages(”scatterplot3d”) CRAN-pakettivarastosta.

4.2.5 RStudio

Kuviossa 22 on RStudio, joka on R-ympäristöön kehitetty työkalu. Se sisältää konsolin, ohjelmointiympäristön, debuggaus työkalut sekä visualisointityökalut samassa näkymässä. RStudiolla komennot voidaan suorittaa suoraan konsolista tai ne voidaan kirjoittaa scripti-tiedostoon. Käyttöliittymässä pääsee myös tarkastelemaan tarkemmin, mitä ympäristössä olevat eri muuttujat ja datajoukkiot pitävät sisällään.



KUVIO 22. RStudion käyttöliittymä

4.3 Hadoop ja R

RHadoop on yksi Hadoop projekti, jonka tarkoituksena on yhdistää Big Datan säilöntä ja muokkaus Hadoop-ympäristössä R-kielen analysointi työkaluihin käyttäen hyväksi Hadoopin Streaming rajapintaa. RHadoop on R-paketti, joka sisältää funktiota ja toimintoja tämän mahdollistamiseksi.

RHadoop on viiden R-paketin kokoelma. Rhdfs sisältää toiminnot, joiden avulla voidaan olla yhteydessä HDFS:ään. Näitä toimintoja ovat järjestelmän tiedostojen selaaminen, lukeminen, kirjoitus sekä muokkaus. Rhbase mahdollistaa yhteyden HBASE-hajautettuun tietokantaan Thrift-palvelinta käyttäen. Tämä paketti mahdollistaa tietokannan taulujen käsittelyn. Plyrmr mahdollistaa yleiset data manipulaatio -operaatiot Hadoopiin varastoiduille suurille tietovarastoille. Rmr2-paketti mahdollistaa MapReduce-toiminnot R-kieleen integroituna. Ravro-paketin avulla

voidaan lukea ja kirjoittaa avro-tiedostoja paikallisesta järjestelmästä sekä HDFS-järjestelmästä.

RHadoopin avulla voidaan suorittaa Hadoopin MapReduce -operaatioita R-kielellä sekä yhdistää tähän R:stä löytyviä analysointi ja datan manipulointi työkaluja. Tämän ansiosta samalla ohjelmalla voidaan suorittaa sekä datan keräys, manipulointi että analysointi operaatioita.

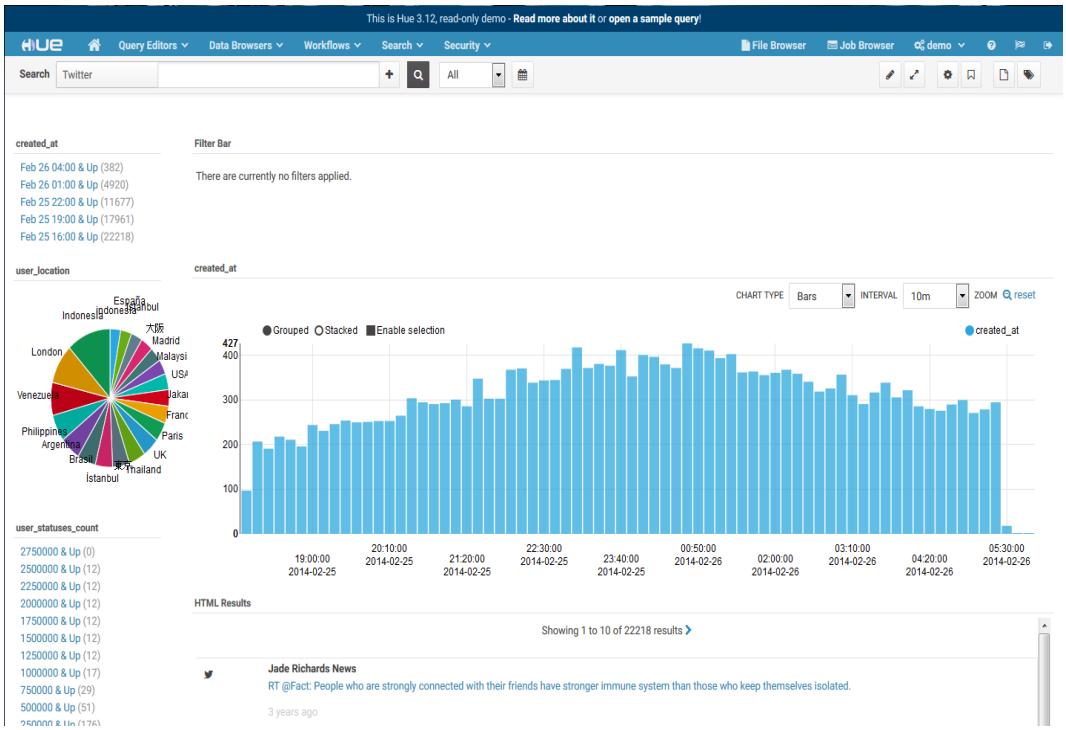
RHadoop sisältää myös perustoimintoja HDFS:ssä sijaitsevien tiedostojen hallinnointiin. Näitä toimintoja ovat tiedoston manipulointi, tiedoston luku ja kirjoitus, kansioiden hallinta sekä HDFS:n tiedostojen tarkastelutyökalut. Tiedostojen manipulointi mahdollistaa HDFS:n tiedostojen kopioinnin, siirtämisen, uudelleen nimeämisen, poistamisen sekä tiedostojen tuomisen ja viemisen HDFS-järjestelmästä.

4.4 Visualisointi työkaluja

Datan visualisoimiseen on olemassa eri työkaluja. Datan visualisointiin voidaan käyttää joko valmiita työkaluja, kuten Excel tai Power BI, tai sitten ohjelmoimalla. R-kieli sisältää lukuisia paketteja datan visualisoimiseen. Tämän lisäksi esimerkiksi Javascriptille on useita kirjastoja, kuten Chart.js.

4.4.1 Hadoop HUE

Hue (Hadoop user experience) on verkkokäyttöliittymä Hadoopin ekosysteemiin. Sen tehtävänä on yksinkertaistaa Hadoop jobien luomista, ylläpitämistä ja ajamista. Hue toimii kaikissa eri Hadoopin jakeluversioissa. Näitä ovat esimerkiksi Hortonworks, Cloudera ja MapR.



KUVIO 23. Huen käyttöliittymä (gethue 2017)

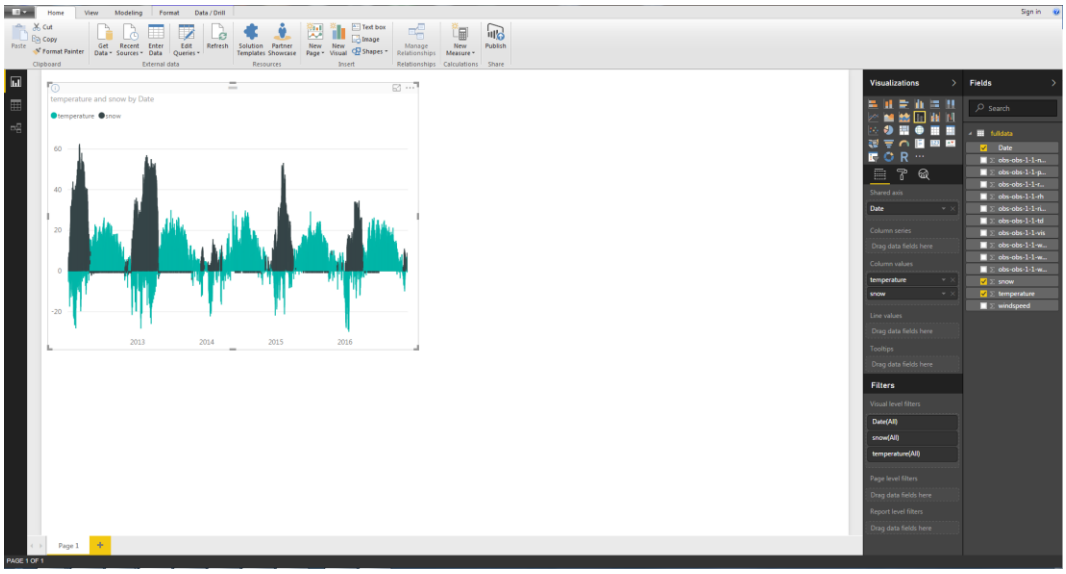
Kuviossa 23 näkyy Huen graafinen käyttöliittymä, joka yhdistää Hadoop ekosysteemin eri projektit. Siihen on integroituna eri tietokantakysely menetelmät, datan selaus työkalut, työnkulunhallinta sekä tiedonhaku HDFS-järjestelmästä. Tietokantakyselyt voidaan toteuttaa Hiven, Impalan ja Pigin avulla.

Hue mahdollistaa erilaisten graafisten havainnointityökalujen käytön interaktiivisesti HDFS:n ja muiden tiedostojärjestelmien kanssa. Se mahdollistaa erilaisten widgettejen käytön omien dashboardien luomiseen. Sillä on mahdollista esimerkiksi karttojen, erilaisten diagrammien ja heat mappejen käyttö datan havainnollistamiseen.

4.4.2 Microsoft Power BI

Microsoft Power BI on raportointi- ja analysointipalvelu. Se mahdollistaa interaktiivisten visualisointien, raporttien ja dashboardien luomisen. Se

toimii lukuisien eri datatyyppien kanssa, kuten Excel-tiedostot, CSV-tiedostot, GitHub ja Google Analytics. Power BI:lla on myös mahdollista suorittaa R scriptejä.



KUVIO 24. Power BI:n työpöytä-näkymä

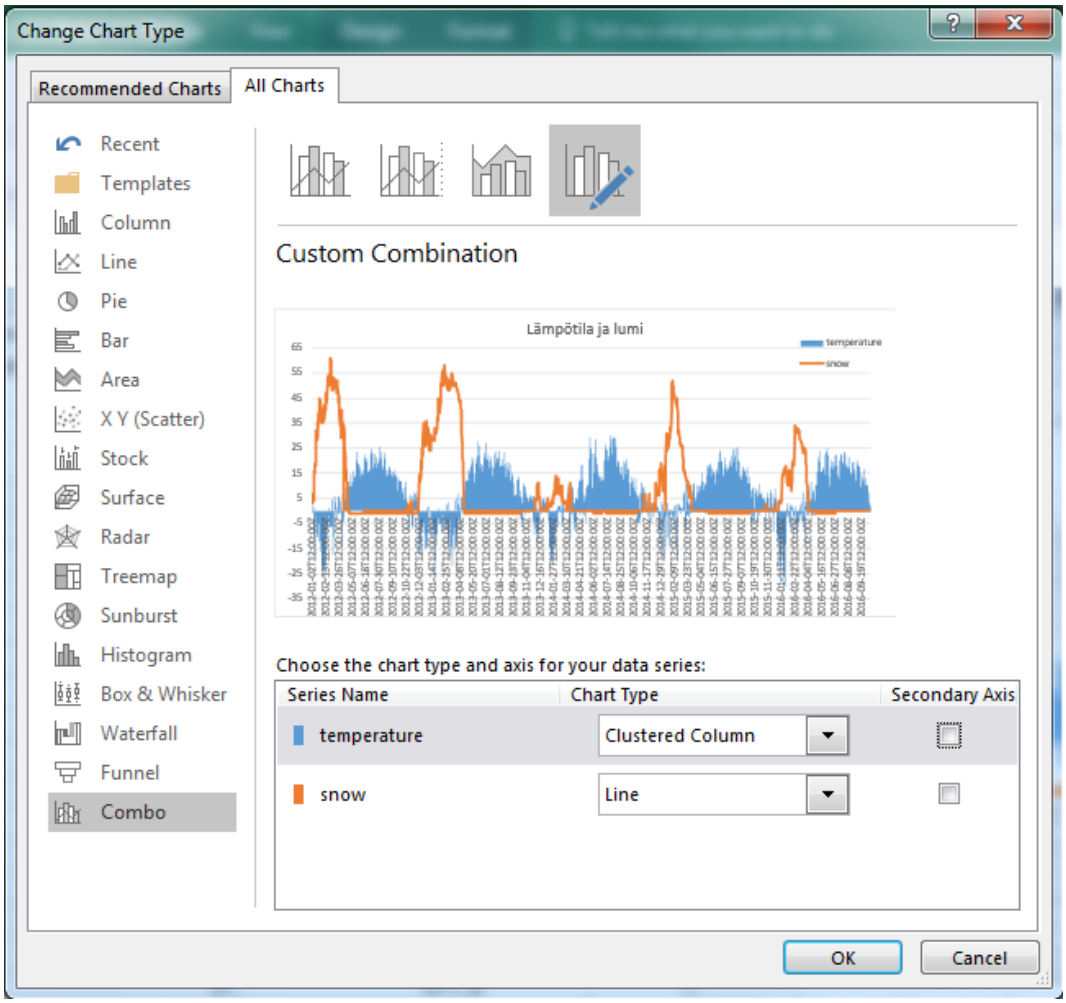
Power BI:sta on sekä työpöytä-versio että pilvipalvelu. Molemmat versiot jakavat ominaisuuksia keskenään, mutta sisältävät myös versiokohtaisia ominaisuuksia. Kuviossa 24 näkyy Power BI:n työpöytä-versio, joka mahdollistaa datan latauksen ja käsittelyn. Datalle on mahdollista tehdä erilaisia muunnos- ja muokkaustoimenpiteitä, kuten sarakkeiden pilkkomista eri osiin, termien muuttamista ja desimaalipisteiden muuttamista. Työpöytä versiossa on myös mahdollista laatia tietomalleja eli tauluja voidaan yhdistää toisiinsa esimerkiksi id-tiedon perusteella. Pilvipalvelu taas mahdollistaa tehtyjen raporttien julkaisun pilvipalvelussa muiden nähtäväksi. Raporttien sisältöä voi myös muokata pilvipalvelussa. Raporttien tarkastelu onnistuu joko selaimella tai mobiilisovelluksille.

Power BI osaa itse jaotella esimerkiksi .csv -muotoisen datan kentiksi. Visualisoiminen onnistuu helposti valitsemalla halutut datakentät. Tämän jälkeen valitaan haluttu graafin tyyppi. Graafeille voidaan määrittää erilaisia suodattimia virheellisten arvojen poistamiseksi.

4.4.3 Excel

Microsoft Excel on taulukkolaskentaohjelma, joka sisältää datan analysointiin- ja visualisointiin käytettäviä työkaluja. Excelin ominaisuudet eivät ole yhtä kattavat kuin tehtävään suunnitelluilla työkaluilla, mutta se soveltuu silti tehtävään. Excelillä on mahdollista tehdä niin heat mappeja kuin pistekaavioihin. Excelillä pystyy luomaan graafeja suoraan syötetystä datasta esimerkiksi Pivot-taulun avulla. Graafit myös päivittyvät arvojen muutosten mukaan.

Graafejen luonti tapahtuu ensin valitsemalla halutut sarakkeet ja rivit Excel-taulukosta. Tämän jälkeen graafin luominen tapahtuu yksinkertaisimmillaan yhdellä painalluksella. Graafeja pystyy myös muokkaamaan haluamansa näköiseksi esimerkiksi ulkoasua tai akseleiden nimiä muuttamalla.



KUVIO 25. Graafejen muokkaus

Excelissä on lukuisia erilaisia graafimalleja, joita se osaa myös ehdottaa valitulle datalle. Kuviossa 25 näkyy myös mahdollisuus yhdistää eri datasarjoja samaan graafiin. Tämä kombo-graafi mahdollistaa myös toissijaisen akselin käytön graafeissa, jos tarvitsee havainnollistaa eri tyyppistä dataa samassa graafissa.

Excel soveltuu parhaiten valmiin datan analysointiin ja esittämiseen. Datan manipulointi Excelillä on myös mahdollista, mutta se on työlästä. Excel tukee Visual Basic -ohjelmointikieltä, joka mahdollistaa datan käsittelyn.

5 CASE: ILMATIETEENLAITOKSEN AVOIN DATA

Työssä tutkittiin avoimen datan käytön mahdollisuuksia big data -ratkaisujen tutkimisessa. Työssä käytettynä testidatana oli Suomen ilmatieteenlaitoksen avoin data. Data saatiin ilmatieteenlaitoksen omaa avoimen datan rajapintaa hyväksi käyttäen.

Työn suorittamiseen käytettiin Virtualboxille asennettua CentOs 7 -käyttöjärjestelmää. Käyttöjärjestelmälle asennettiin Apache Hadoop 2.7.2 sekä RHadoop.

Data tulee palvelimelta XML:ään pohjautuvassa Geography Markup Language (GML) muodossa. Datan saamiseksi rajapinnasta vaaditaan ilmaista rekisteröitymistä ilmatieteenlaitoksen rajapintaan. Rekisteröitymisen jälkeen saadaan henkilökohtainen API-avain, jonka avulla rajapintaa pääsee käyttämään. Ilmatieteenlaitoksen palveluissa on API-avain kohtaiset datan pyyntörajoitukset. Latauspalveluun voi tehdä 20000 pyyntöä päivässä. Palvelu rajoittaa myös pyyntöjen tiheyden 600 pyyntöön 5 minuuttia kohden.

Ilmatieteenlaitokselta haettiin dataa väliltä 1.1.2012 – 1.1.2017, ja otoksen aikaväli oli kolme tuntia. Palvelusta haettiin kaikki mahdollinen data Lahdesta kyseiseltä aikaväliltä big dataa simuloimiseksi. Tämä kuitenkin muodosti ongelmaksi sen, että kyseisellä pyynnöllä oli rajoituksena 168h aikaväli. Ongelma ratkaistiin tekemällä silmukassa pyyntöjä yhdelle viikolle kerralla, ja kirjoittamalla saatu data viikkokohtaisesti omaan .xml-tiedostoonsa. Ajan hallintaan käytettiin Joda-Time -Java-kirjastoa.

Datan noutamiseen tehtiin Java-ohjelma, joka hakee datan ilmatieteenlaitoksen palvelimelta. Ohjelmassa luotiin Joda-Timen DateTime objektit viikon alulle, lopulle sekä haun viimeiselle päivämäärälle. DateTimeFormatterilla muutettiin päivämäärät haun tarvitsemaan yyyy-mm-dd -muotoon.

```
while (endDate.isBefore(finalDate)) {
    URL fmiAPI = new URL("http://data.fmi.fi/fmi-apikey/"+"/"
            + api
            + "wfs?request=getFeature&storedquery_id="
            + "fmi::observations::weather::timevaluepair"
            + "&place=lahti"
            + "&timestep=180"
            + "&starttime=" + formatter.print(startDate) + "T01:00:00Z"
            + "&endtime=" + formatter.print(endDate) + "T01:00:00Z");
```

KUVIO 26. URL-osoitteen muodostaminen

Itse yhteys palvelimelle otettiin Javan URLConnectionin avulla (KUVIO 26). Sieltä saatu sivu luettiin BufferedReaderiin. Tämän jälkeen se muutettiin merkkijonoiksi StringBuilderin avulla. Tämä kirjoitettiin tiedostoon PrintWriterin avulla. Tiedostoja muodostui jokaista viikkoa kohden yksi.

```
<wml2:MeasurementTimeseries gml:id="obs-obs-1-1-t2m">
    <wml2:point>
        <wml2:MeasurementTVP>
                <wml2:time>2012-01-29T03:00:00Z</wml2:time>
                <wml2:value>-22.2</wml2:value>
        </wml2:MeasurementTVP>
    </wml2:point>
```

KUVIO 27. Yhden mittausarvon gml-syntaksi

Muodostetut tiedostot ovat GML-syntaksilla varustettuja XML-tiedostoja. Kuviossa 27 gml:id -attribuutti on mitatun asian otsikko. Kuvion 27 tapauksessa obs-obs-1-1-t2m tarkoittaa lämpötilaa. Elementti wm12:point sisältää mittausarvot wm12:time, joka on mittaus ajankohta, sekä wm12:value, joka on mitattu arvo.

Toisella Java-ohjelmalla kerättiin kaikki muodostuneet .xml-tiedostot ja muutettiin ne yhdeksi .txt-tiedostoksi, joka oli sisällöllisesti .csv:n kaltainen. Ohjelma latasi kansion sisältä kaikki tiedostot yhteen muuttujaan. Tämän jälkeen tiedostojen sisältö ajettiin xml-parseriin. Ohjelmassa käytettiin

hyväksi javax-kirjaston xml-parseria. NodeListiin kerättiin kaikki wm12:MeasurementTVP-tagilla varustetut elementit. Tämä elementti pitää sisällään sekä wm12:time että wm12:value elementit. Lista käytiin läpi yksi node kerrallaan ja kerättiin kaikki eri arvot jokaista mittausajankohtaa kohden.

```
if (elementMap.containsKey(eElement.getElementsByTagName("wml2:time").item(0).getTextContent())) {

    tempString = elementMap.get(eElement.getElementsByTagName("wml2:time").item(0).getTextContent())
     + "," + eElement.getElementsByTagName("wml2:value").item(0).getTextContent();

    elementMap.put(eElement.getElementsByTagName("wml2:time").item(0).getTextContent(), tempString);

} else {
    elementMap.put(eElement.getElementsByTagName("wml2:time").item(0).getTextContent(), eElement.
    getElementsByTagName("wml2:value").item(0).getTextContent());
}
```

KUVIO 28. Arvojen sijoittaminen HashMapiin

Apuna käytettiin HashMapia, jonka avaimen ja arvon tyyppinä oli String. Tämän mapin avaimeksi tuli wm12:time-elementistä saatu arvo, ja arvoksi tuli wm12:value-elementin arvot yhdistettynä yhdeksi Stringiksi (Kuvio 28). Lopuksi mappi iteroitiin läpi, niin että jokainen avain-arvopari Stringi yhdistettiin yhdeksi Stringiksi pilkulla erotettuna. Nämä kaikki kerättiin yhteen Stringbuilderiin ja kirjoitettiin levylle .txt -tiedostoksi.

```
5  2012-01-01T09:00:00Z,-9.5,0.0,0.4,0.0,93.0,-10.5,NaN,NaN,2.0,1011.4,21370.0,0.0,0.0
6  2012-01-04T09:00:00Z,0.2,2.4,5.1,141.0,98.0,-0.1,NaN,NaN,10.0,982.5,18740.0,7.0,74.0
7  2012-01-05T00:00:00Z,1.5,4.1,9.6,207.0,93.0,0.5,NaN,NaN,6.0,971.9,50000.0,7.0,0.0
8  2012-01-02T12:00:00Z,-0.5,4.0,8.2,138.0,85.0,-2.6,NaN,NaN,3.0,1006.8,50000.0,8.0,0.0
```

KUVIO 29. Datan muoto .txt -tiedostossa

Data siirrettiin HDFS-tiedostojärjestelmään jatkokäsittelyä varten. Data sisältää mittaushetken aikaleiman sekä eri arvot pilkuilla erotettuina kuvion 29 osoittamalla tavalla. R:llä tehtiin MapReduce-työ RHadoopia hyväksi käyttäen.

```
11  ac.map = function(.,lines){
12
13    spl <- strsplit(lines,split="\n")
14    key <- substr(spl,0,10)
15    for (ln in spl) {
16      value <- sub(pattern = substr(ln,0,21),
17                   replacement = "",x = ln)
18    }
19
20    keyval(unlist(key), value)
21  }
```

KUVIO 30. Map toteutettuna R:llä

Data ajettiin MapReducen läpi RHadoopin avulla, ja laskettiin datan arvoille päivien keskiarvot. Map-luokassa sisään tullut data jaettiin riveittäin merkkijonoiksi. Kuvion 30 rivillä 14 merkkijonosta erotettiin alusta aikaleima mapin avaimeksi. Aikaleimasta poistettiin kellonaika, jolloin avaimeksi muodostui pelkkä päivämäärä. Tämän jälkeen arvo-merkkijonosta poistettiin aikaleima kokonaan, jolloin jäljelle jäi mittaustulokset (Kuvio 30, rivi 16). Tämä merkkijono muodosti arvon. Avain-arvopari syötetään kuvion 30 rivillä 20 olevaan keyval()-funktioon, joka muodostaa annetuista arvoista avain-arvopari -listan.

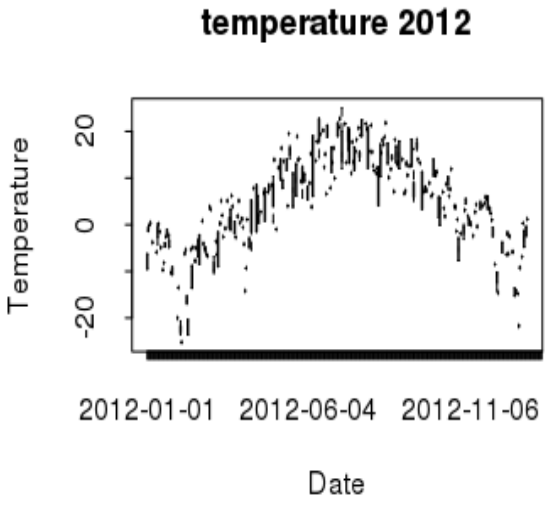
```
23  ac.reduce = function(key,value){
24
25    for (val in value) {
26      valu <- data.frame(unlist(strsplit(val, split=" ")))
27    }
28    valu[] <- lapply(valu[], function(x) as.numeric(as.character(x)))
29    cmean <- colMeans(valu,na.rm = TRUE)
30    keyval(key, cmean)
```

KUVIO 31. Reducen R toteutus

Reducer keräsi saadut avaimet arvoineen. Reducer luokka oli yksinkertainen. Ensin kuviossa 31 rivillä 26 se muutti silmukan sisällä kaikki saadut arvot datakehys-muotoon ja pilkkoi merkkijonon tyhjien merkkien mukaan eri vektoreiksi. Koska reduceriin tullut arvo on merkkijonomuotoa, se täytyi muuttaa numeeriseksi keskiarvojen laskemiseksi. Rivillä 28 oleva lapply-funktio kutsuu parametrinaan määriteltyä funktiota annetun muuttujan jokaiselle alkioille x. Tässä tapauksessa se käy kaikki alkiot läpi ja muuttaa ne numeerisiksi. Tämän jälkeen näille vektoreille laskettiin keskiarvo sarakkeittain. Tämä data kirjoitettiin takaisin HDFS-tiedostojärjestelmään.

Saatu data haettiin HDSFS-tiedostojärjestelmästä loppukäsittelyä varten. Data ladattiin datakehykseen, joka mahdollistaa manipulaation. Datalle lisättiin sarakkeiden otsikot sekä tehtiin havainnollistavia diagrammeja.

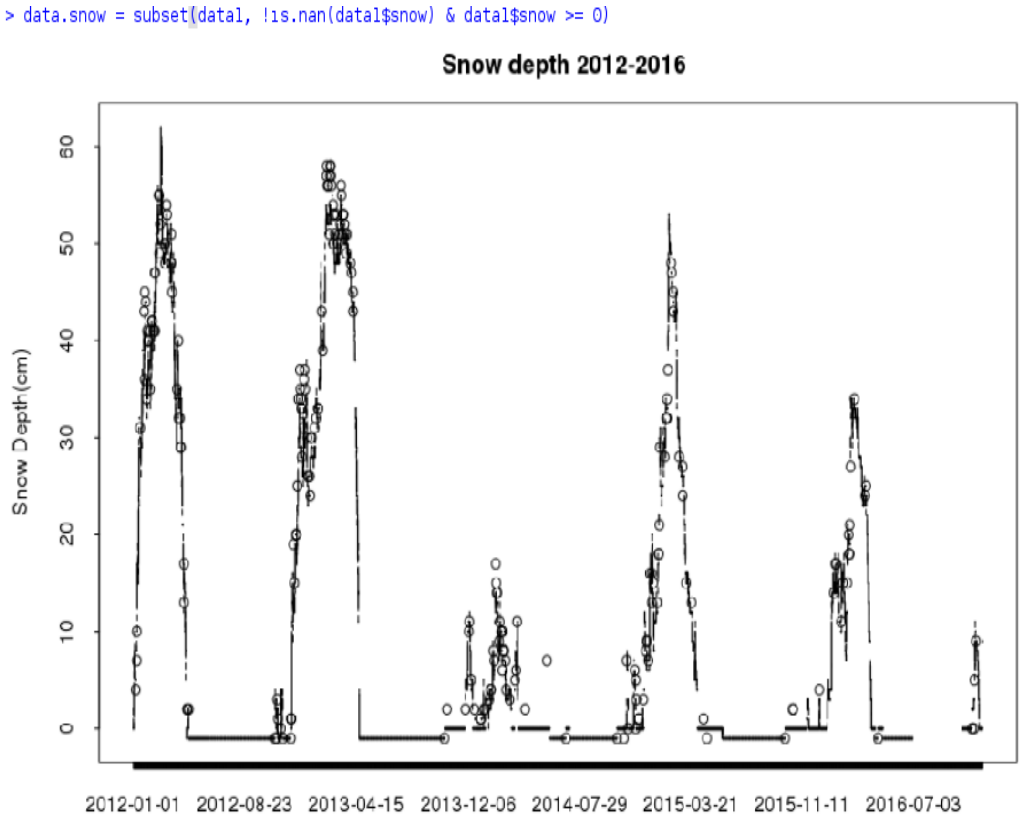
```
> plot(sdata.year$Date, sdata.year.temp$Temperature, type="l", xlim=c(sdata.
year[1,],sdata.year[360,]),ylim(c(-30,20)))
> title(main="temperature 2012", xlab="Date",  ylab="Temperature")
```



KUVIO 32. viivadiagrammi plot()-funktiolla

Kuviossa 32:ssa on esitettynä vuoden 2012 lämpötilat aika-akselilla. Graafia varten lämpötilat sekä päivämäärät erotettiin täydestä datasta

pienemmiksi osajoukoiksi omiin muuttujiinsa. Näiden muuttujien avulla piirrettiin graafi.



KUVIO 33. Lumen syvyys vuosina 2012 – 2016

Kuviossa 33 Näkyy vuosien 2012-2016 lumen syvyydet. Datasta poistettiin tyhjät arvot, jolloin mittauksesta ei ollut arvoa sekä negatiiviset arvot kuvion 33 ylälaidassa olevalla komennolla. Komento suodattaa sarakkeesta kaikki ei-numerot sekä luvut, jotka ovat alle nollan.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli tutustua Big Dataan sekä sen käsittelyyn ja analysointiin Hadoop-ympäristössä. Työssä käytettiin Ilmatieteen laitoksen tarjoamaa avointa dataa simuloimaan Big Dataa. Datan hakemiseen, tallentamiseen sekä käsittelemiseen tehtiin erilliset ohjelmat.

Työn teoriaosuudessa käytiin läpi mitä Big Data pitää sisällään ja miten se eroaa muusta datasta. Työssä myös käytiin läpi Hadoop-ympäristön toimintaperiaatetta sekä hiukan sen ympärille kasvaneita projekteja. Viimeisenä teoriaosuutena käytiin läpi datan analysointiin käytettäviä vaiheita sekä muutamaa datan visualisoimiseen käytettävää työkalua.

Käytännönosuudessa käytiin läpi työn kulku. Työssä kuvailtiin datan hakuun tehtyä ohjelmaa sekä datan muuttamista xml-tiedostosta txt-tiedostoksi. Työssä myös kuvaillaan manipuloimista R-ympäristön avulla.

Opinnäytetyön päätteeksi säädataa saatiin onnistuneesti analysoitua sekä visualisoitua. Työn tekemiseen tarvittavien Hadoop- ja R-ympäristöjen pystyttäminen onnistui lähes ongelmitta. Myös datan hakeminen ilmatieteen laitoksen palvelimelta onnistui. Suurimpana ongelmana työn tekemisessä oli testiympäristön liian vähäinen keskusmuistin määrä, joka rajoitti työn tekemistä.

Jatkokehityksenä työn kulkua voisi automatisoida sekä käyttöliittymää kehittää. Tällä hetkellä tämä prosessi koostuu kahdesta eri Java-ohjelmasta sekä R-scripteistä. Datan siirto ohjelmien välillä on nykyisessä versiossa toteutettu manuaalisesti. Datan hakevaan Java-ohjelmaan voisi toteuttaa konsolikäyttöliittymän, koska tämän hetkisessä koodissa kaikki tarvittava data, kuten kansiopolut ja päivämäärät, on kovakoodattuna lähdekoodiin.

LÄHTEET

A4academics 2016. Hadoop Map Reduce Architecture and Example [viitattu 9.4.2017]. Saatavissa: http://a4academics.com/tutorials/83-hadoop/840-map-reduce-architecture

Apache Software 2017a. Introduction [viitattu 9.4.2017]. Saatavissa: https://ambari.apache.org/

Apache Software 2017b. What Is Apache Hadoop? [viitattu 9.4.2017]. Saatavissa: https://hadoop.apache.org/

Bappalige, S. 2014. An introduction to Apache Hadoop for big data [viitattu 10.4.2017]. Saatavissa: https://opensource.com/life/14/8/intro-apache-hadoop-big-data

Beal, V. 2017. Structured data [viitattu 9.4.2017]. Saatavissa: http://www.webopedia.com/TERM/S/structured_data.html

BrightPlanet 2012. Structured vs. Unstructured data [viitattu 20.10.2016]. Saatavissa: https://brightplanet.com/2012/06/structured-vs-unstructured-data/

Bernard, M. 2015. 10 Big Data Use Cases Everyone Must Read [viitattu 9.4.2017]. Saatavissa: http://www.datasciencecentral.com/profiles/blogs/spark-or-hadoop-which-is-the-best-framework

Bernard, M. 2016. The Most Practical Big Data Use Cases of 2016 [viitattu 9.4.2017]. Saatavissa: https://www.forbes.com/sites/bernardmarr/2016/08/25/the-most-practical-big-data-use-cases-of-2016/#6a8e6fed3162

Bernard, M. 2017. How is Big Data Used in Practice? 10 Use Cases Everyone Must Read [viitattu 9.4.2017]. Saatavissa: https://www.ap-

institute.com/big-data-articles/how-is-big-data-used-in-practice-10-use-cases-everyone-should-read

Diadmin 2015. Characteristics of Big Data – Part One [viitattu 15.10.2016]. Saatavissa: http://www.dataintensity.com/characteristics-of-big-data-part-one/

Gethue 2017. gethue demo [viitattu 9.4.2017]. Saatavissa: gethue.com

Gewirtz, D. 2016. Volume, velocity, and variety: Understanding the three V's of big data [viitattu 15.10.2016]. Saatavissa: http://www.zdnet.com/article/volume-velocity-and-variety-understanding-the-three-vs-of-big-data/

Gorakala, S. 2014. Data Analysis Steps [viitattu 9.4.2017]. Saatavissa: https://www.r-bloggers.com/data-analysis-steps/

Hadoop Team 2015. Hadoop Modes [viitattu 9.4.2017]. Saatavissa: http://hadoopinrealworld.com/hadoop-modes/

hadoopbigdata 2016. Hadoop Racks in 6 steps [viitattu 9.4.2017]. Saatavissa: http://hadoopbigdata.org/hadoop/hadoopracks6steps/

Hedlund, B. 2011. Understanding Hadoop Clusters and the Network [viitattu 9.4.2017]. Saatavissa: http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/

IBM 2017. About Hive [viitattu 9.4.2017]. Saatavissa: https://www-01.ibm.com/software/data/infosphere/hadoop/hive/

Kaycee, J. 2016. Hadoop Ecosystem Overview [viitattu 9.4.2017]. Saatavissa: http://thebigdatablog.weebly.com/blog/the-hadoop-ecosystem-overview

Marr, B. 2017. How is Big Data Used in Practice? 10 Use Cases Everyone Must Read [viitattu 9.4.2017]. Saatavissa: http://www.ap-institute.com/big-

data-articles/how-is-big-data-used-in-practice-10-use-cases-everyone-should-read.aspx

Menon, R. 2014. hadoop in 5 minutes [viitattu 9.4.2017]. Saatavissa: https://techvital.wordpress.com/2014/07/01/hadoop-in-5-minutes/

Merrian-Webster 2014. Definition of BIG DATA [viitattu 15.10.2016]. Saatavissa: http://www.merriam-webster.com/dictionary/big%20data

Miller, R. 2015. Inside Facebook's Blu-Ray Cold Storage Data Center [viitattu 15.10.2016]. Saatavissa: http://datacenterfrontier.com/inside-facebooks-blu-ray-cold-storage-data-center/

Oxford 2013. Definition of big data in English [viitattu 15.10.2016]. Saatavissa: https://www.oxforddictionaries.com/definition/english/big-data

Rouse, M. 2014. semi structured data [viitattu 9.4.2017]. Saatavissa: http://whatis.techtarget.com/definition/semi-structured-data

R-project 2017. What is R? [viitattu 9.4.2017]. Saatavissa: https://www.r-project.org/about.html

Salo, I. 2013. Big Data tiedonvallankumous. Jyväskylä: Docendo

Shon, P. 2014. apache hbase explained in 5 minutes or less [viitattu 9.4.2017]. Saatavissa: https://www.credera.com/blog/technology-insights/java/apache-hbase-explained-5-minutes-less/

Singh, V. 2013. Hadoop and HDFS [viitattu 9.4.2017]. Saatavissa: https://www.dezyre.com/article/hadoop-components-and-architecture-big-data-and-hadoop-training/114

Wikispace 2017. MapReduce [4.9.2017]. Saatavissa: https://hadooptutorial.wikispaces.com/MapReduce