OULUN AMMATTIKORKEAKOULU

Salla Paso

PILVIPOHJAISEN OHJELMISTO- JA LAITEYMPÄRISTÖN KEHITTÄMINEN TILASTOLLISEEN LASKENTAAN

PILVIPOHJAISEN OHJELMISTO- JA LAITEYMPÄRISTÖN KEHITTÄMINEN TILASTOLLISEEN LASKENTAAN

Salla Paso
Opinnäytetyö
Kevät 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, langattomat laitteet

Tekijä: Salla Paso
Opinnäytetyön nimi: Pilvipohjaisen ohjelmisto- ja laiteympäristön kehittäminen tilastolliseen laskentaan
Työn ohjaajat: Kari Jyrkkä, Jari Hyttinen, Jukka Lämsä
Työn valmistumislukukausi ja -vuosi: Kevät 2017
Sivumäärä: 29 + 3 liitettä

Opinnäytetyön aiheena oli pilvipohjaisen ohjelmisto- ja laiteympäristön kehittäminen tilastolliseen laskentaan ja työn tilaajana toimi Nokia Oyj. Tavoitteena oli ympäristöjen kehittämisen lisäksi niiden toimivaksi todentaminen Matlabilla toteutetulla demo-sovelluksella.

Työn toteutus alkoi pilvi- ja klusterijärjestelmiin tutustumisella ja siihen liittyvien asioiden opiskelulla. Eri järjestelmien tutkimisen ja testauksen jälkeen laiteympäristö päädyttiin toteuttamaan klusterina Linux-alustalle ja pilvipalveluympäristö toteutettiin Apache Hadoopilla. Järjestelmien rakentamisen jälkeen toteutettiin Matlab-sovellus, jolla analysoitiin yrityksen tuotannosta saatavaa dataa.

Työn tuloksena saatiin luotua toimiva kolmen PC:n muodostama klusteri pilvipalvelujärjestelmällä. Lisäksi luotiin toimiva laskentasovellus Matlabia käyttäen ja yrityksen omaan käyttöön jäävä dokumentaatio, josta löytyy ohjeet klusterin rakentamiseen ja pilvipalvelun luomiseen. Yrityksen käyttöön löytyi myös monia jatkokehitysideoita.

Asiasanat: klusteri, Matlab, pilvipalvelu, Linux, Apache Hadoop

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Wireless Devices

Author: Salla Paso
Title of thesis: Creation of a Cloud-based Software and Hardware Environment for Statistical Computing
Supervisors: Kari Jyrkkä, Jari Hyttinen, Jukka Lämsä
Term and year when the thesis was submitted: Spring 2017
Pages: 29 + 3 appendices

The subject of this thesis was to create a cloud-based software and hardware environments for statistical computing. The customer was Nokia Oyj. The objectives of this thesis were to create the environments and confirm those environments with Matlab application.

The first phase of this work was to study cloud and cluster computing systems. After researching and testing few different systems, it was decided to implement the environments as a cluster computation system with Linux and Apache Hadoop.

As a result of this thesis, a cluster was made of three PC's with cloud computing system and a successful application was created using Matlab to process manufacturing-related big data. Also a document was written for the company, which describes how the cluster and cloud computing systems were made. Along with those results, there were many new development ideas for the cluster.

Keywords: cluster, Matlab, cloud, Linux, Apache Hadoop

ALKULAUSE

Haluan kiittää Nokia Oyj:tä tämän opinnäytetyön mahdollistamisesta ja mielenkiintoisesta aiheesta sekä opinnäytetyöni yhdyshenkilöinä ja valvojina toimineita Jari Hyttistä ja Jukka Lämsää työni ohjaamisesta ja palautteen antamisesta. Isot kiitokset haluan antaa myös kannustaville työkavereilleni.

Työni ohjaajana OAMK:n puolelta toimi Kari Jyrkkä, jota haluan kiittää työn aikana saamastani palautteesta ja ohjauksesta.

Suuret kiitokset tuesta ja kannustuksesta kevään 2017 aikana kuuluvat myös avopuolisolleni, ystävilleni ja perheelleni.

Oulussa 9.5.2017

Salla Paso

SISÄLLYS

SANASTO

Big data	Yleisnimitys erittäin suurille datamäärille, voidaan käyttää myös suomenkielistä versiota ’massadata’.
Cpk	Cpk kuvaa suorituskykyindeksinä laskentaprosessin potentiaalista suorituskykyä ja sillä lasketaan toleranssialueen keskikohdasta keskiarvon poikkeama.
Hadoop	Avoimen lähdekoodin ohjelmistokehys. Käytetään palvelinklusterin luomiseen.
Klusteri	Tietokonejoukosta muodostettu looginen tietokone.
Klusterilaskenta	Sovellusten suorittamista klusterilaskentajärjestelmässä hajautetusti ja rinnakkain.
KVM-kytkin	Keyboard, Video, Mouse -switch. Laite, jolla voi hallinnoida useaa tietokonetta yhden näppäimistön, näytön ja hiiren avulla.
LCL	Lower Control Limit. Alempi tilastollinen kontrolliraja.
Linux	Yleisin palvelinkäyttöjärjestelmä.
Live-USB	USB-tikulla oleva käyttöjärjestelmä, joka on käyttövalmis, kun tietokoneen käynnistää käyttämällä kyseistä USB-tikkua.
LSL	Lower Specification Limit. Alempi spesifikaatioraja.
Node	Klusterin osa eli noodi.
Pilvilaskenta	Sovellusten suorittamista laskentapilvessä hajautetusti ja rinnakkain.
Rufus	Ohjelma, jonka avulla voi alustaa ja luoda käynnistyviä Live-USB -laitteita.
SSH	Secure Shell. Salattuun tiedonsiirtoon tarkoitettu tietoliikenneprotokolla.
UCL	Upper Control Limit. Ylempi tilastollinen kontrolliraja.

USL	Upper Specification Limit. Ylempi spesifikaatioraja.

1 JOHDANTO

Tietotekniikan ala kehittyy jatkuvasti ja tallennettavan tiedon määrä kasvaa massadatan (engl. big data) myötä. Täten myös tallennustilan tarve kasvaa. Etenkin nykyaikaisissa tehtaissa, joissa tulee päivittäin suuria määriä mittaus- ja testausdataa, olisi järkevää käyttää sitä tehtaan prosessien ohjaukseen, kuitenkin luovuttamatta tietoja tehtaan ulkopuolelle. Lisää tallennustilaa voikin saada esimerkiksi pilvipalveluista, jotka skaalautuvat nopeasti ja helposti käyttäjänsä tarpeiden mukaisesti (1, s. 17).

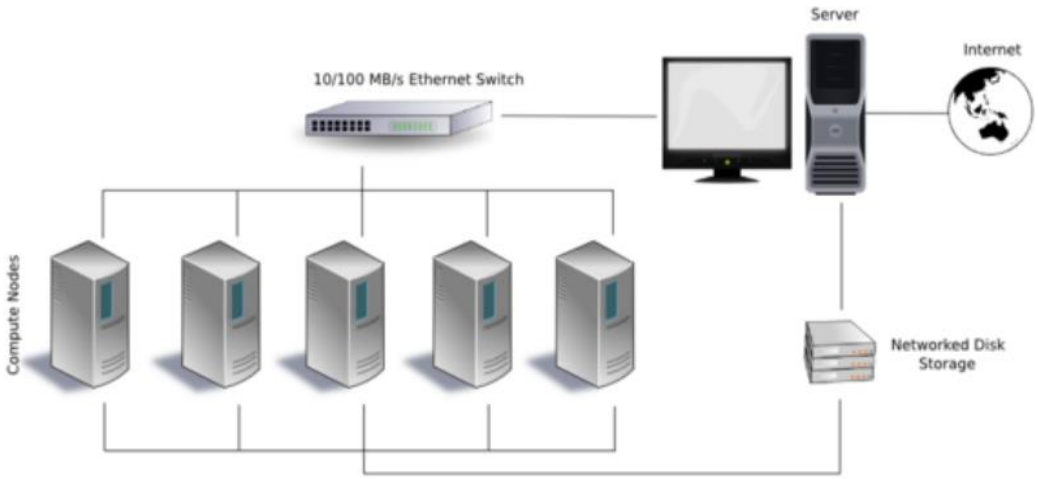
Tavoitteena tässä opinnäytetyössä on kehittää ohjelmisto- ja laiteympäristö pilvipohjaiseen tilastolliseen laskentaan. Toteutetun laite- ja ohjelmistoympäristön toiminta todennetaan Matlabilla tehtävän demo-sovelluksen avulla. Demo-sovelluksen ymmärtämisen avuksi tässä työssä on käyty myös lyhyesti teoriaa tilastollisesta laskennasta ja suorituskykyanalyysista.

Työn käytännön osuus toteutetaan Linux-alustalle klusterina. Yrityksen oma sisäinen pilviratkaisu haluttiin valita ulkoistetun sijaan, koska yrityksessä oli tarve lisätä osaamista ja tietoa aiheesta. Lisäksi käytettyjä laskentamenetelmiä ei haluttu luovuttaa kolmansille osapuolille.

Tämä opinnäytetyö on osa Oulun ammattikorkeakoulun tietotekniikan koulutusohjelman langattomien laitteiden suuntautumisvaihtoehdon opintoja. Työn tilaajana toimii Nokia Oyj ja työ suoritetaan kevään 2017 aikana. Nokia on yli sadassa maassa toimiva maailmanlaajuinen teknologiakehittäjä, joka tarjoaa asiakkailleen kattavan valikoiman tuotteita, palveluita sekä lisensointimahdollisuuksia (2).

2 KLUSTERI

Klusteri-termiä (engl. cluster) käytetään talouden, tieteen ja teollisuuden alalla, mutta pääperiaate on aina sama. Tarkka klusterin määritelmä vaihtelee riippuen käytettävästä klusterista. Tietotekniikassa klusteri tarkoittaa yleensä samankaltaisten tai identtisten tietokoneiden joukkoa, joka toimii yhtenäisesti ja toteuttaa sille annettuja tehtäviä. Klusterissa olevat tietokoneet voivat olla myös virtuaalikoneita. Klusterin jäsentietokoneiden fyysiset laitteet sijaitsevat samassa paikassa ja niiden väliset yhteydet ovat erittäin nopeita. (3, s.167–168.) Kuvassa 1 on esitetty klusterin tyypillinen fyysinen konfiguraatio.



KUVA 1. Klusterin tyypillinen fyysinen konfiguraatio (4)

Klusterin muodostavia palvelimia kutsutaan noodeiksi (engl. node) (5, s. 11). Klustereita käytettäessä saadaan enemmän suoritustehoa verrattuna yhteen tietokoneeseen. Suoritustehon nousu perustuu hajautukseen. (6, s. 7.)

Niiden avulla voidaan saavuttaa monia hyödyllisiä asioita, kuten helppo päivitettävyys, suurien järjestelmien hyvä hintasuhde, riippumattomuus laitevalmistajista sekä avoimeen lähdekoodiin perustuvien alustojen hyödyntäminen. Lisäksi saadaan varmuutta palveluiden saatavuuteen ja ympäristöä voidaan skaalata tarpeen mukaan. (7, s. 2.)

Klusteroinnin tarkoituksena on käyttää useampaa käyttöjärjestelmää yhdessä, jolloin yksi koneista toimii palvelimena. Klusterointia voidaan käyttää esimerkiksi tietokannoissa, palvelimissa sekä laskennassa. (8, s. 5.)

Virtualisointi tarkoittaa fyysisen palvelimen jakamista useaksi virtuaalipalvelimeksi ohjelmistokerroksella. Jokaisessa palvelimessa on oma käyttöjärjestelmänsä. (9.)

Virtualisoimalla palvelimia voidaan säästää resursseja sekä tila- ja sähkövaatimuksia. Suurena hyötynä virtualisoinnissa on myös vikasietoisuus ja datan saumaton siirto virtuaalikoneiden välillä. (10.)

2.1 Klusterityypit

Klusterityyppejä voidaan kehittää moneen tarkoitukseen ja tarpeeseen. Niihin voi luoda myös erilaisia palveluita ja palvelimia, kuten tiedosto-, tietokanta- ja tulostuspalvelimia. (11.) Tässä opinnäytetyössä luotu klusteri oli tyypiltään laskentaklusteri.

2.1.1 Laskentaklusterit

Hyödyntämällä klusterin kaikkia koneita yhtä aikaa voidaan muodostaa laskentaklusteri, jolla laskennan suorittaminen on huomattavasti tehokkaampaa. Tämän voi toteuttaa esimerkiksi jakamalla työtaakkaa noodien kesken asentamalla haluttu palvelu tai ohjelmisto kaikkiin klusterin noodeihin, jolloin ohjelmisto hoitaa tehtävien hajauttamisen. (12, s. 7.)

Hajautettua laskentaa käytettäessä kaikki klusterin koneet suorittavat samaa algoritmia. Hajautettu järjestelmä suorittaa diskreettejä synkronisia kierroksia, jotka kaikki sisältävät kolme kohtaa, jotka ovat viestin lähettäminen viereiselle klusterille, viestin vastaanottaminen sekä paikallisen laskennan suorittaminen. Algoritmin suorittaminen alkaa ensimmäisellä kierroksella ja kierroksia toteutetaan, kunnes kaikki koneet ovat suorittaneet laskennan ja saaneet tuloksen. (13, s. 19.)

2.1.2 Korkean käytettävyyden klusterit

Korkean käytettävyyden klusterit ovat klustereita, jotka on suunniteltu toimimaan yksittäisistä verkkoyhteyksien katkeamisista tai laitteiden rikkoutumisista huolimatta (14). Tällaisia klusteripalveluita tuotetaan, kun tavoitteena on taata käyttäjälle palveluiden maksimaalinen saatavuus (5, s. 14). Etuna korkean käytettävyyden klustereissa on vikasietoisuus, joka mahdollistaa klusterin yksittäisten koneiden korjaamisen ja päivittämisen ilman palveluiden alasajoa. Jos joku klusterin koneista esimerkiksi sammuu hallitsemattomasti, siirtyvät sen tehtävät automaattisesti toiselle klusterin koneelle. (15.)

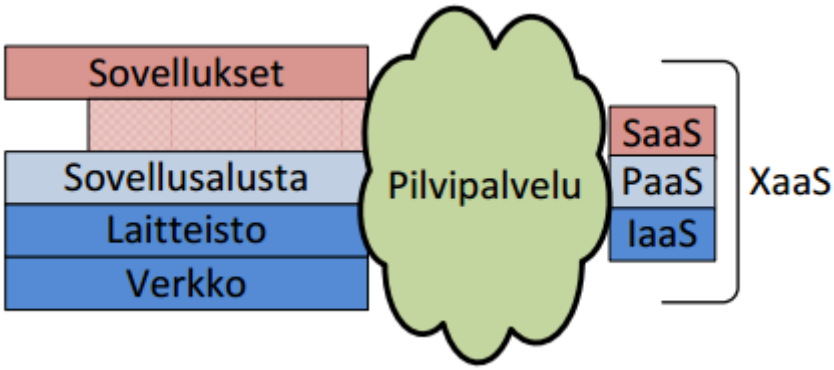
2.2 Klusterin rakentaminen

Klusterin rakentaminen alkaa fyysisestä osuudesta, eli hankitaan klusterissa käytettävät tietokoneet. Jo kahdesta fyysisestä tai virtuaalisesta tietokoneesta voi rakentaa klusterin. Tietokoneiden tulisi olla mahdollisimman samanlaiset, jollei jopa identtiset IP-osoitteita lukuun ottamatta, ja niiden tulee olla yhteydessä toisiinsa esimerkiksi Ethernet-kaapeleilla. Koneille asennetaan käyttöjärjestelmä sekä käyttöjärjestelmän kanssa yhteensopiva klusterointiohjelmisto, jonka jälkeen klusterille annetaan oma IP-osoite ja nimi. Toimintaa kannattaa testata erilaisilla yhteystesteillä verkon yli, kuten esimerkiksi SSH-yhteydellä. (5, s.12.)

3 PILVIPALVELUT

Pilvipalvelu, tutummin pilvi, on yleisesti käytetty termi, jolle ei kuitenkaan ole vakiintunutta määritelmää tai rajausta (16, s. 16). Normaalisti pilvellä tarkoitetaan helposti ylläpidettävää palvelua, joka tarjoaa virtuaalisia resursseja näyttämättä loppukäyttäjälle sen fyysisiä ominaisuuksia. Yleisimpiä pilvipalveluita on esimerkiksi Google Drive, Microsoft OneDrive sekä Facebook. (5, s. 8.)

Nykyisin verkkopalveluita varten voidaan ostaa resursseja jonkun palveluntarjoajan pilvestä, jolloin yrityksen ei tarvitse itse investoida pilvipalvelun ylläpitämiseen (5, s. 8). Kuten kuvasta 2 käy ilmi, pilvipalvelumallit jaetaan yleensä kolmeen eri malliin, jotka ovat infrastruktuuri palveluna (IaaS, Infrastructure as a Service), sovellusalusta palveluna (PaaS, Platform as a Service) ja verkkosovellukset palveluna (SaaS, Software as a Service). Näitä yhdistämällä voidaan muodostaa myös malli, joka voi sisältää minkä tahansa palvelun (XaaS, Anything as a Service). (16, s. 22–23.)



KUVA 2. Pilvipalvelumallit (5)

Pilvipalvelutyyppejä ovat yksityinen, yhteisöllinen, julkinen sekä hybridipilvi (16, s. 19). Yksityinen pilvi on tarkoitettu nimensä mukaisesti yksityiseen käyttöön yksittäisessä yrityksessä. Yksityisen pilven ylläpito voidaan hoitaa joko kyseisen yrityksen sisällä tai ulkoistaa, jolloin pilvipalvelun laitteisto voi sijaita palveluntarjoajan tiloissa. Tällöin on kuitenkin huomioitava riittävä tietoturvan ja hyödyntämisen suunnittelu. Jaettua yksityistä pilveä kutsutaan yhteisöpilveksi. Yhteisöpilvi on hyvä keino jakaa kustannuksia, mutta samalla tietoturvariskit kasvavat. (17, s. 55)

Julkista pilveä käytetään yleensä Internetin kautta. Julkinen pilvi on hyvin samankaltainen kuin yksityinen pilvi, mutta tietoturvan tulee olla tarkempi. Hybridipilvi rakennetaan useasta eri pilvityypistä, jotka yleensä ovat julkinen ja yksityinen pilvi. Yritys voi itse yhdistää omaan yksityiseen pilveensä julkisen pilvipalvelun resursseja, ja täten pilvien virtuaalikoneiden yhdistäminen onnistuu. (17, s.54–56.)

Tässä opinnäytetyössä luotiin yksityinen PaaS-tyyppinen pilvipalvelumalli, eli kehitimme alustan, jonka päällä voidaan suorittaa, testata, kehittää ja ylläpitää erilaisia sovelluksia.

4 TILASTOLLINEN LASKENTA JA SUORITUSKYKYANALYYSI

Tämä luku käsittelee teoriaa tilastollisesta laskennasta ja suorituskykyanalyysista helpottamaan ymmärtämistä klusterissa käytettävästä Matlabilla luotavasta sovelluksesta. Sovelluksessa on tavoitteena käsitellä tuotannon normaalijakaumaa noudattavaa testidataa ja mitata sen laskennallista suorituskykyä, joten sitä tarkemmin tilastotieteitä ei ole tarvetta käsitellä. Luvussa 6 kerrotaan tarkemmin sovelluksen sisällöstä.

4.1 Tilastollinen jakauma

Useimmiten numeerisessa muodossa oleva empiirinen tutkimusaineisto on järkevintä käsitellä tilastollisesti (18). Laskennassa tulee käyttää tilastollisia kontrollirajoja, jotta mittaus pysyy tilastollisesti kontrollissa ja tulossarjan jakauma olisi normaali. Kontrollissa pysyminen tarkoittaa sitä, että tunnistetaan kaikki virhetilanteet ja muut tuloksiin vaikuttavat tekijät. Mitatessa laskennallista suorituskykyä mittaustapahtumasta tulee mittauksen olla stabiili ja datan tulee olla vähintään lähes normaalisti jakautunut. Kontrollirajoina voidaan käyttää alempaa ja ylempää kontrollirajaa (eli Lower Control Limit, LCL ja Upper Control Limit, UCL). (19, s. 22.)

Tilastollista jakaumaa voidaan tarkastella sellaisesta tulosjoukosta, joka ei sisällä mittausvirheitä ja jonka on todettu olevan kontrollissa. Laskelmien luotettavuuden takaamiseksi tulee tietää datan jakautuneisuus ennen laskelmien tekemistä. Esimerkiksi suorituskyvyn mittaaminen pystytään tekemään vain normaalisti jakautuneelle datalle. (19, s. 23.)

Normaalisti jakautuneesta datasta piirretty kuvaaja muotoutuu Gaussin käyrän, eli kellokäyrän, mukaisesti. Tämä tarkoittaa sitä, että datan havaintoarvot asettuvat otosjoukon keskiarvon ympärille likimain tasaisesti. Jos data ei ole jakautunut normaalisti, on jakauman kuvaaja erityisen litteä, vino tai huipukas. Jakauman selvittämisessä voidaan käyttää hyväksi myös goodness-of-fit-testejä, esimerkiksi kun halutaan tietää toleranssirajojen näkökulmasta datapisteiden sijoittuminen jakaumalle. (19, s. 23–24.)

Jakauman tiheysfunktion eli Gaussin kellokäyrän kaava voidaan laskea, mikäli satunnaismuuttuja noudattaa normaalijakaumaa, ja sitä merkitään kuten kaavassa 1.

KAAVA 1. Normaalijakauman laskukaava (20)

$$X \sim N(\mu, \sigma)$$

missä
μ = jakauman sijainti
σ = jakauman muoto.

Jakauman muodosta voidaan tutkia tulosten sopivuutta normaalijakaumaan. Normaalisti jakautunut data on symmetrinen keskiarvoonsa nähden, eli datapisteet jakautuvat likimain tasaisesti keskiarvon kummallekin puolelle. Normaalijakauman vinous- ja huipukkuuskertoimen arvot ovatkin 0. (19, s. 24.)

Normaalijakauman ollessa kaavan 1 mukainen, lasketaan tiheysfunktio kuten kaavassa 2 esitetään.

KAAVA 2. Tiheysfunktion laskukaava (20)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^1}}$$

missä
μ = jakauman odotusarvo
σ = jakauman keskihajonta.

Normaalijakauman todennäköisyydet voidaan määrittää kertymäfunktion avulla. Tiheysfunktion integroiminen mahdollista tehdä vain numeerisesti, joten usein todennäköisyydet otetaan esimerkiksi MAOL- tai jonkin muun taulukkokirjan normaalijakauman kertymäfunktion taulukosta. Taulukot on kuitenkin laadittu vain sellaisille satunnaismuuttujille, joiden odotusarvot ovat $\mu = 0$ ja $\sigma = 1$, sillä taulukoiden tekeminen kaikilla μ:n ja σ:n 1 arvoilla on mahdotonta. Tällöin jakaumaa kutsutaan normitetuksi normaalijakaumaksi. (20.)

4.2 Suorituskykyindeksit ja Cpk

Cpk kuvaa suorituskykyindeksinä laskentaprosessin ns. potentiaalista suorituskykyä. Tuotekehityksessä suorituskykylukujen avulla voidaan kuvata tuoteominaisuuksien toimivuutta suhteessa spesifikaatiorajoihin. Ennen indeksin laskemista tulee määrittää mittaukselle tavoitearvo ja ylempi sekä alempi spesifikaatioraja (eli Upper Specification Limit ja Lower Specification Limit). USL ja LSL määritetään mitattavan datan speksien perusteella. Cpk:lla lasketaan toleranssialueen keskikohdasta keskiarvon poikkeama (19, s. 33). Cpk ottaa siis huomioon vain poikkeaman toleranssivälin keskikohdasta otetut mittaustulosten keskiarvot. (21, s. 176.) Kaavassa 3 on esitetty Cpk:n laskukaava.

KAAVA 3. Cpk:n laskukaava (19)

$$C_{pk} = \min(\frac{USL-\bar{x}}{3\hat{\sigma}}, \frac{\bar{x}-LSL}{3\hat{\sigma}})$$

missä
$\bar{x}$ = mittaustulosten keskiarvo
$\hat{\sigma}$ = mittaustulosten keskihajonta
USL = ylempi spesifikaatioraja
LSL = alempi spesifikaatioraja.

5 KÄYTÄNNÖN TOTEUTUS

Tämä luku kertoo opinnäytetyössä suoritetusta käytännön toteutuksesta. Käytännön toteutukseen kuuluivat klusterin ja pilvipalvelun rakentaminen sekä järjestelmän testaus.

Käyttöjärjestelmänä päätettiin käyttää Linuxia, joka on täysin ilmainen avoimen lähdekoodin käyttöjärjestelmä. Linux on yleisesti suosittu käyttöjärjestelmä verkkosovelluksien ajamiseen, verkkoliikenteen ohjaamiseen ja muihin taustaprosesseihin eri organisaatioissa (22). Linuxille ei myöskään ole tehty juurikaan viruksia tai haittaohjelmia, ja siihen on hankala hyökätä (8, s. 4).

5.1 Klusterijärjestelmän rakentaminen

Opinnäytetyössä luotu klusteri rakennettiin käyttämällä kolmea Teito 4U -tietokonetta, jotka yhdistettiin Belkin SoHo KVM -kytkimeen sekä Ethernet-kaapeleilla reitittimeen. Kytkimen avulla pystyi helposti vaihtamaan käytettävää tietokonetta napin painalluksella, joten klusterin käyttämiseksi tarvittiin vain yksi näyttö, näppäimistö ja hiiri. Tietokoneille asennettiin yrityksen oma muokattu Red Hat Enterprise Linux 7.1 -versio tietoturvan optimoimiseksi. Linuxin asennus tapahtui Live-USB:lla, joka luotiin asentamalla ISO-levykuva Rufuksen avulla USB-tikulle. Linuxin asennuksessa noudatettiin yrityksen omaa asennusopasta (23).

Kuvassa 3 näkyy toteutettu kolmen tietokoneen klusteri, jossa alin tietokone toimii isäntänä ja samalla ohjaa muita tietokoneita. Ylimmän tietokoneen päällä on kaikkiin tietokoneisiin yhdistetty KVM-kytkin, jonka päällä reititin, johon kaikki kolme tietokonetta on yhdistetty Ethernet-kaapeleilla. Lisäksi reititin on kytketty testauslaboratorion paikalliseen verkkoon.

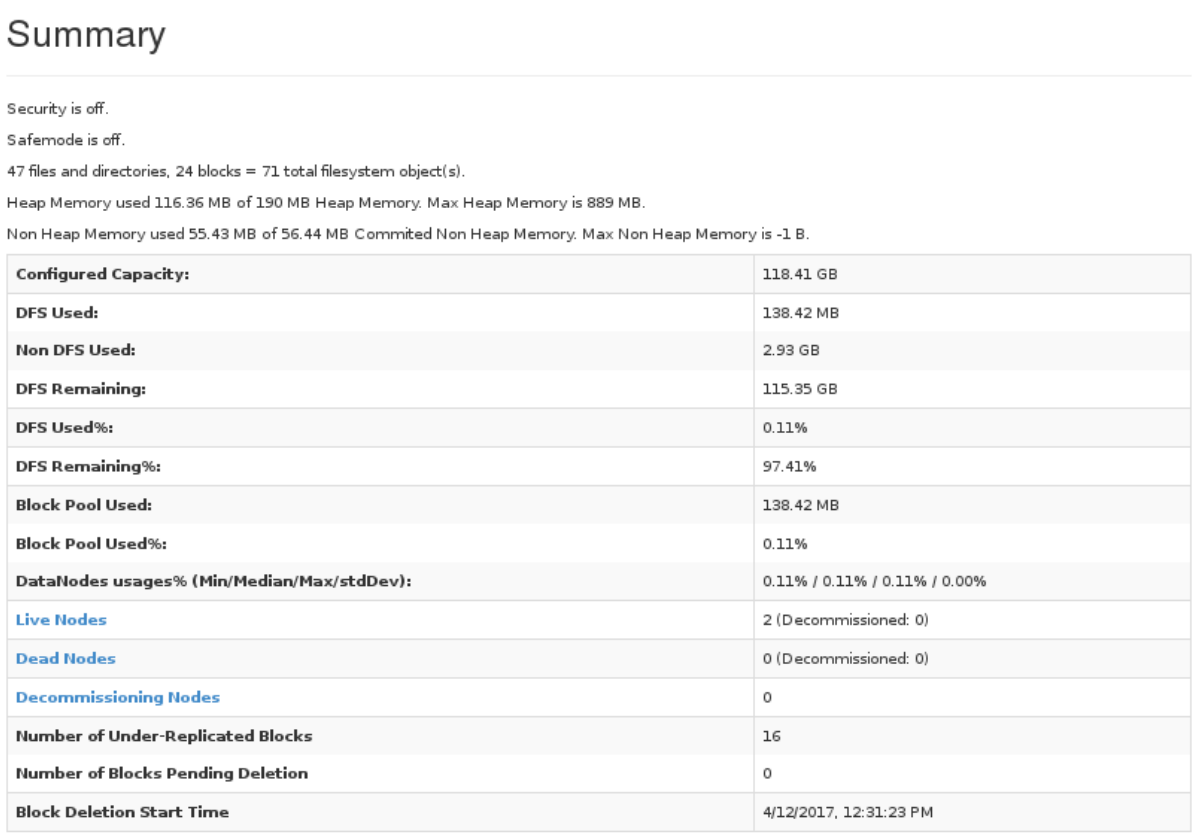
KUVA 3. Fyysinen klusteri (24)

Eri järjestelmien tutkimisen ja testausten jälkeen alustaksi päädyttiin asentamaan Apache Hadoop -ohjelmistokehys. Valinta tuli tehdä nopeasti yrityksen sisäisen projektin aikatauluvaatimuksista johtuen. Hadoop vaatii toimiakseen Javan, joten asennus aloitettiin siitä ja asennus tapahtui asennusohjeen mukaisesti (25). Tämän jälkeen alkoi Hadoop 2.6.0 -version asennus asennusohjetta mukaillen (26). Asennuksessa käytetyt komennot löytyvät liitteestä 1. Yhden tietokoneen toimivaksi toteamisen jälkeen koneen kovalevyn sisältö kloonattiin muiden tietokoneiden kovalevyille. Valmiissa klusterissa oli kolme tietokonetta, joista yksi määriteltiin toimimaan isäntänä (engl. master) ja kaksi orjana (engl. slave). Määrittely tapahtui konfiguraatiotiedostojen avulla, jotka ovat liitteessä 2.

5.2 Järjestelmän testaus

Järjestelmää testattiin luomalla Javalla WordCount-ohjelma, joka laski valitun tiedoston sanamäärät ja ilmoitti ne käyttäjälle. Lisäksi ohjelma ilmoitti laskemiseen kuluneen ajan. Laskentaan käytettävä tiedosto vietiin HDFS-tiedostojärjestelmään ja koodi ajettiin niin yhdellä PC:llä kuin klusterilla. Klusterin todettiin olevan yhtä PC:tä huomattavasti nopeampi. Käytetty koodi on nähtävillä liitteessä 3.

Myös Hadoopin HDFS-tiedostojärjestelmästä voidaan käydä tarkastamassa, ovatko konfiguroidut noodit käynnissä (kuva 4). Live Nodes -kohdassa näkyy 2 kappaletta noodeja, koska isäntänä toimiva kone toimii NameNodena ja kyseiset kaksi noodia ovat DataNodeja. Noodien ollessa käynnissä, voi niille antaa erilaisia tehtäviä suoritettavaksi.



Summary

Security is off.

Safemode is off.

47 files and directories, 24 blocks = 71 total filesystem object(s).

Heap Memory used 116.36 MB of 190 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 55.43 MB of 56.44 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

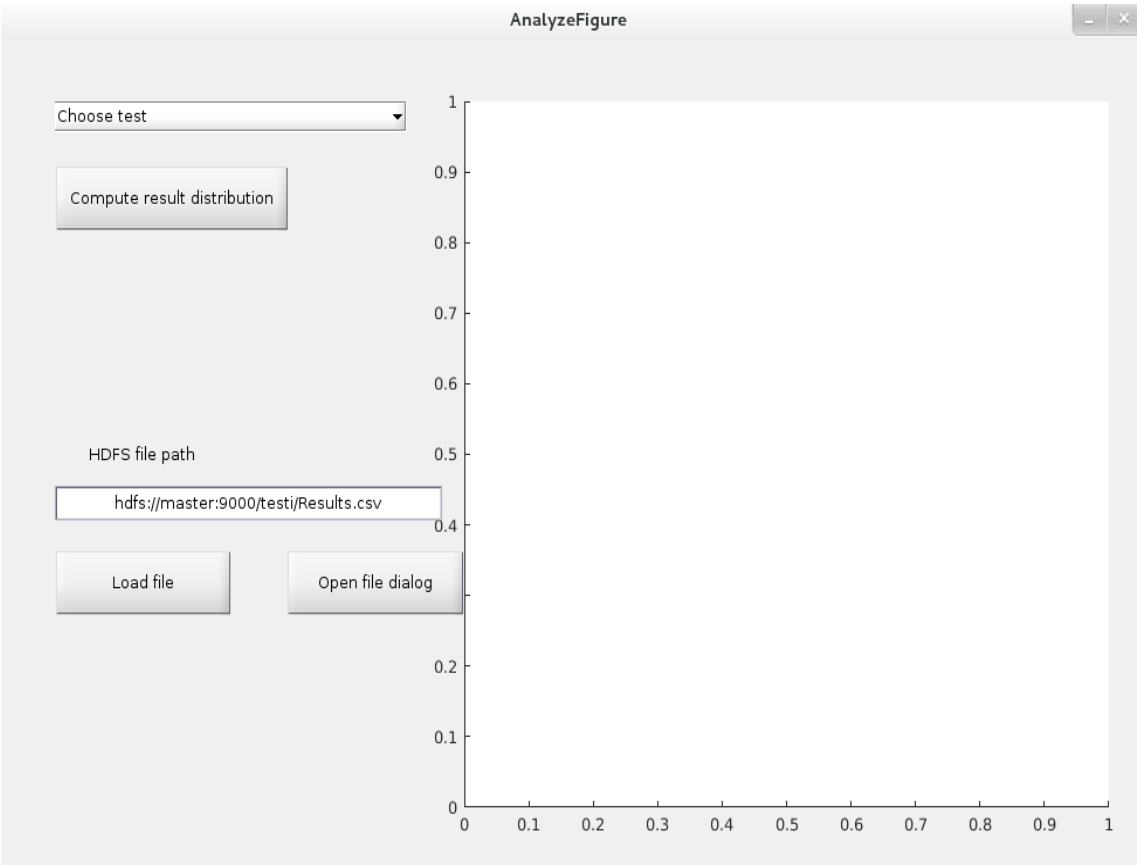
Configured Capacity:	118.41 GB
DFS Used:	138.42 MB
Non DFS Used:	2.93 GB
DFS Remaining:	115.35 GB
DFS Used%:	0.11%
DFS Remaining%:	97.41%
Block Pool Used:	138.42 MB
Block Pool Used%:	0.11%
DataNodes usages% (Min/Median/Max/stdDev):	0.11% / 0.11% / 0.11% / 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	16
Number of Blocks Pending Deletion	0
Block Deletion Start Time	4/12/2017, 12:31:23 PM

KUVA 4. Yhteenveto Hadoopin noodien tilasta

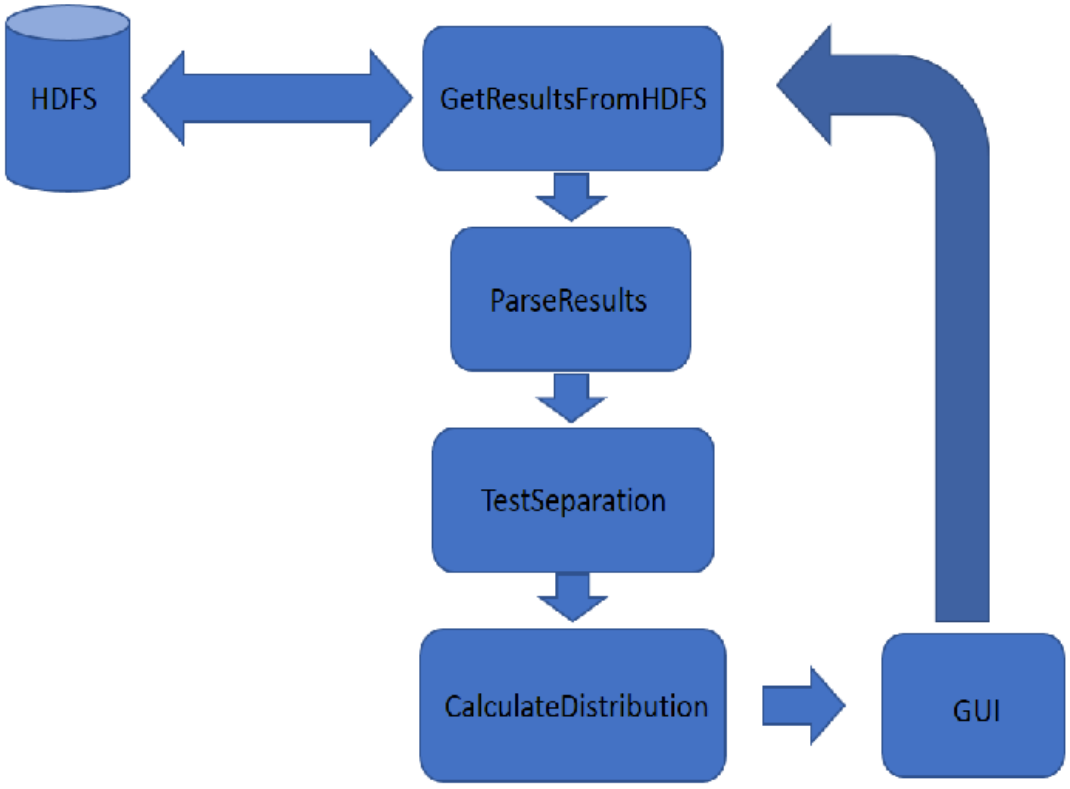
6 KLUSTERIN HYÖDYNTÄMINEN MATLABISSA

Tämä luku kertoo klusterin hyödyntämisestä Matlabilla luotavassa sovelluksessa. Matlab on laajalti käytetty työkalu numeerisessa laskennassa (27). Matlabista asennettiin versio 9.1 ja tarvittavat työkalupaketit isäntänä toimivalle koneelle.

Matlabilla luotiin erilaisia funktioita, kuten tiedostonlukufunktio, tulosfunktio, testien erottelufunktio sekä funktio, joka muuttaa tiedoston solut numeroiksi, jotta tiedostoa pystyttiin käsittelemään laskennallisesti. Nämä kaikki funktiot koottiin korkean tason funktioon, jota hyödyntämällä luotiin käyttöliittymä, jolla voitiin valita mitä testiä halutaan analysoida. Käyttöliittymä on esitetty kuvassa 5 ja koodeista tehty lohkokaavio näkyy kuvassa 6.



KUVA 5. Käyttöliittymän näkymä ennen testin valintaa HDFS-tiedostojärjestelmästä haetulle tiedostolle



KUVA 6. Matlabilla luotujen koodien lohkokaavio

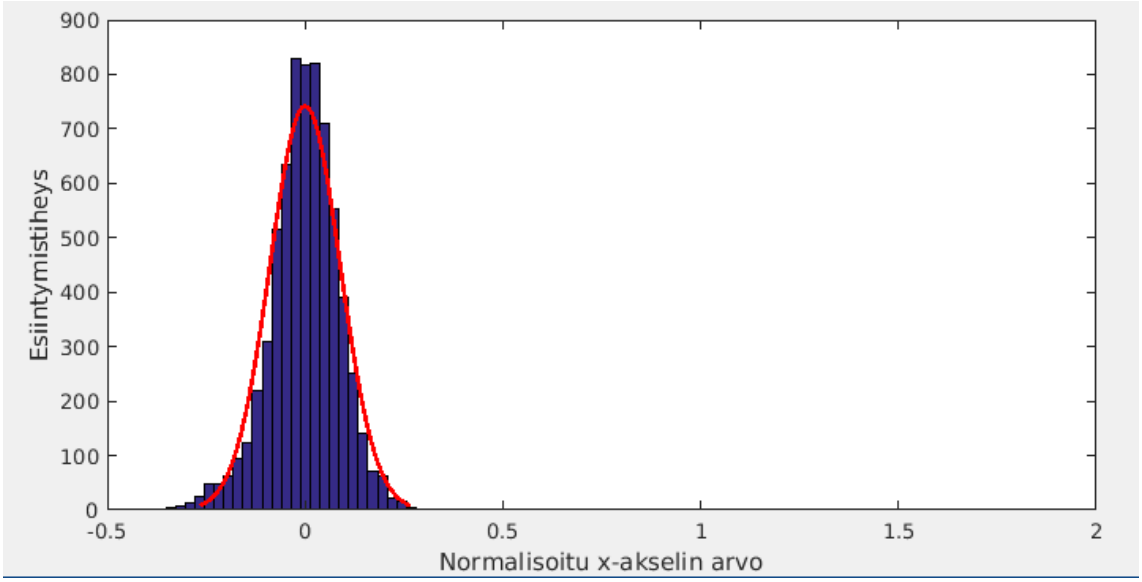
Ohjelma jäsentää halutusta tiedostosta käsiteltävät kohteet, kuten tässä tapauksessa testin nimen, mittaustuloksen, USL:n, LSL:n ja testin lopputuloksen. Jäsennellyistä mittaustuloksista laskettiin sekä keskiarvo että keskihajonta, joiden tuloksien avulla pystyttiin laskemaan Cpk. Lisäksi ohjelma piirtää tuloksista kuvaajan, josta käy ilmi tulosten hajautuneisuus.

Tiedosto, jota haluttiin käsitellä, siirrettiin Hadoopin HDFS-tiedostojärjestelmään ja koodia muokattiin hakemaan tiedosto tiedostojärjestelmästä sen sijaan, että se olisi ollut tietokoneen kovalevyllä. Aikataulullisista syistä rinnakkaislaskentaan tarvittavia lisenssejä ei ehditty hankkia, joten demo-sovellus käyttää HDFS-tiedostojärjestelmää vain tiedon säilyttämiseen, eikä työssä käytetä MapReduce-toimintoa. MapReduce-toimintoa käyttämällä voidaan jakaa työtaakka klusterin kaikkien noodien kesken, ja se otettaneen käyttöön klusterissa myöhemmin. Tässä työssä laskenta tapahtui kuitenkin vielä yhdellä noodilla. Kuvassa 7 näkyy komento, jolla tiedosto siirrettiin HDFS-tiedostojärjestelmään testi-nimiseen kansioon.

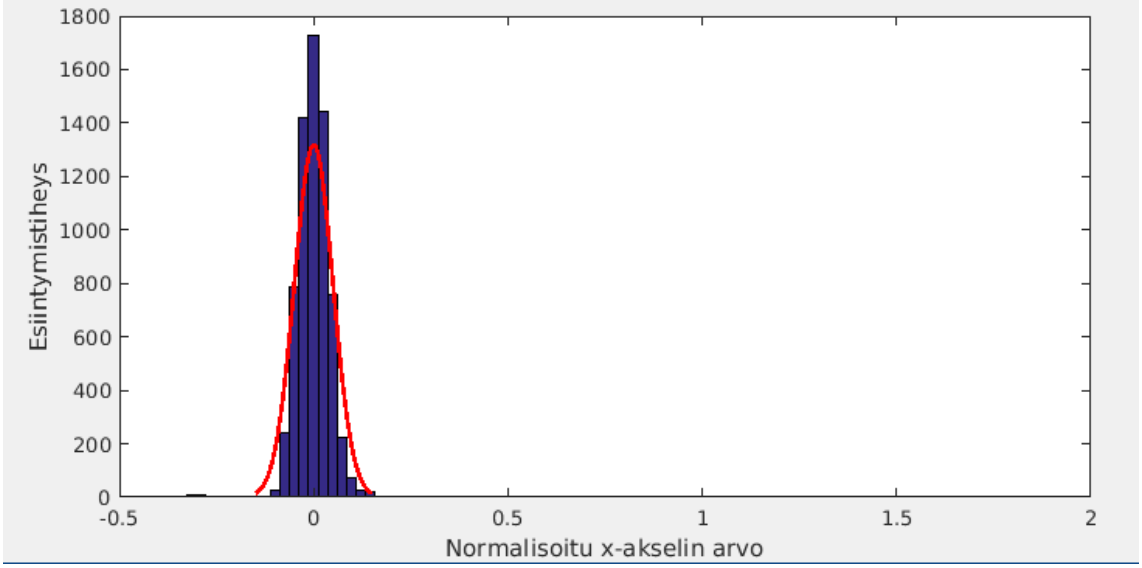
```
[hadoop@master ~]$ hadoop fs -copyFromLocal /home/hadoop/Documents/MATLAB/Results.csv hdfs://master:9000/testi/
```

KUVA 7. Tiedoston siirto tietokoneen kovalevyltä HDFS-tiedostojärjestelmään

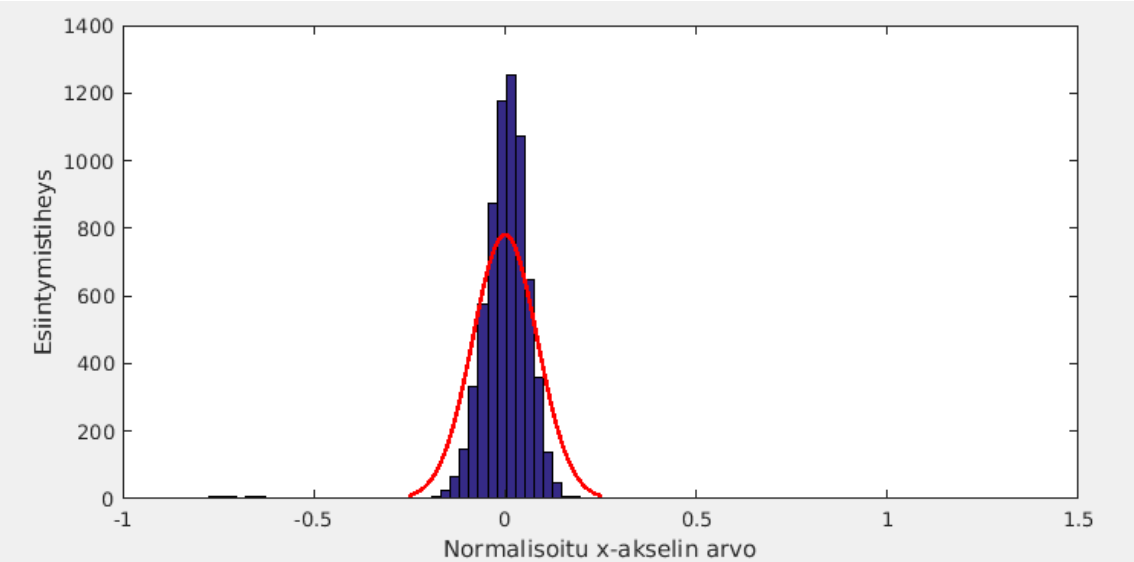
Tässä tapauksessa ohjelmaa käytettiin analysoimaan tehtaan tuotannossa toteutettujen kolmen eri suorituskykytestin testituloksia, joiden kuvaajat ovat nähtävissä kuvissa 8, 9 ja 10. Kuvissa siniset palkit kuvaavat toteutunutta hajontaa ja punainen kuvaaja laskennallista hajontaa Gaussin käyrän mukaisesti.



KUVA 8. Ensimmäisen testin tulosten hajautuneisuus



KUVA 9. Toisen testin tulosten hajautuneisuus



KUVA 10. Kolmannen testin tulosten hajautuneisuus

7 YHTEENVETO

Tässä opinnäytetyössä oli tavoitteena kehittää ohjelmisto- ja laiteympäristö pilvipohjaiseen tilastolliseen laskentaan, joka tuli todentaa toimivaksi Matlabilla tehtävällä sovelluksella. Laiteympäristö tuli toteuttaa klusterina. Opinnäytetyölle oli asetettu oppilaitoksen puolesta aikataululliset rajat, ja yrityksen puolelta joitakin valintoja nopeuttivat muiden käynnissä olevien projektien aikataululliset syyt.

Työn tulokseksi saatiin toimiva kolmen PC:n klusteri PaaS-tyyppisellä pilvipalvelulla, jolle pystyi antamaan erilaisia tehtäviä, kuten esimerkiksi jo mainittu, liitteestä 3 löytyvä toiminnan testauskoodi. Myös Matlabilla toteutettu demo-sovellus on yksi työn tuloksista, mutta sovelluksen ohjelmistokoodeja ei voida jakaa yrityksen ulkopuolisille niiden sisällön vuoksi. Sovellus onnistui kuitenkin odotetusti ja koodien toiminnasta tehtiin lohkokaavio selkeyttämään sovelluksen toiminnan hahmottamista.

Tämän opinnäytetyön lisäksi yritykselle luotiin dokumentti, josta löytyy ohjeet klusterin rakentamiseen ja pilvipalvelun luomiseen. Dokumentti luotiin englannin kielellä, jotta sitä voidaan hyödyntää mahdollisimman laajasti yrityksen sisällä.

Opinnäytetyö oli erittäin mielenkiintoinen ja koin oppivani työtä tehdessä paljon. Työ oli ajankohtainen ja laaja, ja se onnistui kokonaisuutena hyvin. Myös asetetut aikataulutavoitteet saavutettiin. Tuloksena syntyneelle klusterille keksittiin monia yrityksen sisäiseen käyttöön jääviä jatkokehityskohteita. Työn mielekkyyttä lisäsi niin oma kiinnostus pilvipalveluiden ja klustereiden toiminnasta kuin työkavereiden tuki. Myös tilastollinen laskenta oli uutta ja mielenkiintoista asiaa.

LÄHTEET

1. Salo, Immo 2014. Big data & pilvipalvelut. Jyväskylä: Docendo Oy.

2. Ohjelmoitavan maailman globaali teknologiajohtaja. Nokia Oyj. Saatavissa: http://www.nokia.com/fi_fi/tietoa-meista/keita-olemme. Hakupäivä 15.2.2017.

3. Carvalho, L. 2012. Windows Server 2012 Hyper-V Cookbook. Packt Publishing: Olton, Birmingham, GBR.

4. Cloud Computing 2015. Wikimedia. Saatavissa: https://upload.wikimedia.org/wikiversity/en/0/00/Cloud_Computing.wiki.20150310.pdf. Hakupäivä 4.5.2017.

5. Hakala, Anni 2014. Korkeasti käytettävän yksityisen pilvipalvelun toteuttaminen. Opinnäytetyö. Tampere: Tampereen ammattikorkeakoulu, tietojenkäsittelyn koulutusohjelma. Saatavissa: https://www.theseus.fi/bitstream/handle/10024/85283/Hakala_Anni.pdf?sequence=2. Hakupäivä 3.2.2017.

6. Sakkara, Jyri 2016. Kubernetes ja klusteroitava verkkosovellus. Opinnäytetyö. Jyväskylä: Jyväskylän ammattikorkeakoulu, tekniikan ja liikenteen ala. Saatavissa: https://www.theseus.fi/bitstream/handle/10024/120976/Opinnayte_jyri_sakkara_final.pdf?sequence=1. Hakupäivä 21.2.2017.

7. Buyya, Rajkumar; Eskicioglu, Rasit; Graham, Peter; Pourreza, Hossein; Sommers, Frank; Yeo, Chee Shin. Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers. Saatavissa: http://www.cloudbus.org/papers/ic_cluster.pdf. Hakupäivä 21.2.2017.

8. Paavola, Jussi 2010. Klusterin toiminnan tutkiminen Linux-ympäristössä. Opinnäytetyö. Ylivieska: Keski-Pohjanmaan ammattikorkeakoulu, tieto-

tekniikan koulutusohjelma. Saatavissa: https://publications.theseus.fi/bitstream/handle/10024/15791/Jussi_Paavola.pdf?sequence=1. Hakupäivä 15.2.2017.

9. Mäntylä, Juha-Matti 2008. Virtualisointi mullistaa tietotekniikan. Tivi. Saatavissa: http://www.tivi.fi/CIO/2008-11-30/Virtualisointi-mullistaa-tietotekniikan-3158514.html. Hakupäivä 15.2.2017.

10. Briggs, Martin 2014. The Benefits of Server Virtualization. Techopedia. Saatavissa: https://www.techopedia.com/2/29064/trends/virtualization/the-benefits-of-server-virtualization. Hakupäivä 15.2.2017.

11. TechNet 2012. Failover Clustering Overview. Saatavissa: https://technet.microsoft.com/en-us/library/hh831579.aspx. Hakupäivä 3.2.2017.

12. Waldén, Aleksi 2014. Virtualisoitu tulostinpalvelinklusteri. Opinnäytetyö. Lahti: Lahden ammattikorkeakoulu, tietotekniikan koulutusohjelma. Saatavissa: https://www.theseus.fi/bitstream/handle/10024/71133/walden_aleksi.pdf?sequence=1. Hakupäivä 15.2.2017.

13. Rybicki, Joel 2011. Exact bounds for distributed graph colouring. Pro gradu. Helsinki: Helsingin yliopisto, tietotekniikan tiedekunta. Saatavissa: https://helda.helsinki.fi/bitstream/handle/10138/26560/exactbou.pdf?sequence=1. Hakupäivä 3.3.2017.

14. Christensen, Elden 2010. Clustering and High-Availability. Failover Clustering and Network Load Balancing Team Blog. Saatavissa: https://blogs.msdn.microsoft.com/clustering/2010/10/05/evaluating-high-availability-ha-vs-fault-tolerant-ft-solutions/. Hakupäivä 3.2.2017.

15. Alila, Antti 2013. Hyper-V:n käyttöönotossa huomioitavat asiat ja parhaat käytännöt. Hyper-V ja korkea käytettävyys. TechNet IT Pro Blog Finland. Saatavissa: https://blogs.technet.microsoft.com/fiitpro/2013/10/25/hyper-vn-kyttnotossa-huomioitavat-asiat-ja-parhaat-kytnnt-hyper-v-ja-korkea-kytettvyys/. Hakupäivä 3.2.2017.

16. Salo, Immo 2010. Cloud Computing: palvelut verkossa. Porvoo: WSOYpro Oy.

17. Heino, P. 2010. Pilvipalvelut. Hämeenlinna: Talentum Media Oy.

18. Virtuaaliammattikorkeakoulu 2007. Tilastollisen analyysin periaatteet. Ylemmän AMK-tutkinnon metodifoorumi -nettisivu. Saatavissa: http://www2.amk.fi/digma.fi/www.amk.fi/opintojaksot/0709019/1193463890749/1193464131489/1194289328583/1194289853960.html. Hakupäivä 21.3.2017.

19. Härmä, Antti 2010. Tilastollisten menetelmien soveltaminen tuotekehitysympäristössä. Opinnäytetyö. Oulu: Oulun seudun ammattikorkeakoulu, tietojenkäsittelyn koulutusohjelma. Saatavissa: https://www.theseus.fi/bitstream/handle/10024/20923/Harma_Antti.pdf?sequence=1. Hakupäivä 21.3.2017.

20. Internetix. Luku 3 Todennäköisyyslaskennan perusteita. Harjoitusmateriaali. Saatavissa: http://materiaalit.internetix.fi/fi/opintojaksot/5luonnontieteet/matematiikkal/mb3/normaalijakauma. Hakupäivä 4.5.2017.

21. Salomäki, Rauno 1999. Suorituskykyiset prosessit – Hyödynnä SPC. Helsinki: MET.

22. Anttila, Ville. 2016. Docker-ympäristön klusterointi. Opinnäytetyö. Jyväskylä: Jyväskylän ammattikorkeakoulu, tekniikan ja liikenteen ala. Saatavissa: http://www.theseus.fi/bitstream/handle/10024/112021/Anttila_Ville.pdf?sequence=1. Hakupäivä 19.2.2017.

23. NOKIA Enterprise Linux 7. Yrityksen sisäinen dokumentti.

24. NOKIA Cluster Computation Document 2017. Yrityksen sisäinen dokumentti.

25. Kumar, Rahul 2017. How to Install JAVA 8 (JDK/JRE 8u121) on CentOS/RHEL and Fedora. TecAdmin. Asennusohje. Saatavissa:

https://tecadmin.net/install-java-8-on-centos-rhel-and-fedora/#. Hakupäivä 15.2.2017.

26. Kumar, Rahul 2015. How to Setup Hadoop 2.6.0 (Single Node Cluster) on CentOS/RHEL and Ubuntu. TecAdmin. Asennusohje. Saatavissa: https://tecadmin.net/setup-hadoop-2-4-single-node-cluster-on-linux/. Hakupäivä 15.2.2017.

27. MATLAB - MathWorks 2017. The Language of Technical Computing. Saatavissa: https://se.mathworks.com/products/matlab.html. Hakupäivä 4.5.2017.

```
$ su root
```

```
# cd /opt/
```

```
# wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-
securebackup-cookie" "http://download.oracle.com/otn-
pub/java/jdk/8u121-b13/e9e7ea248e2c4826b92b3f075a80e441/jdk-
```

```
# tar xzf jdk-8u121-linux-x64.tar.gz
```

```
# java -version
```

```
# adduser hadoop
```

```
# passwd hadoop
```

```
# su - hadoop
```

```
$ ssh-keygen -t rsa -P ””
```

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
$ chmod 0600 ~/.ssh/authorized_keys
```

```
$ ssh localhost
```

```
$ exit
```

```
$ cd ~
```

```
$ wget http://apache.claz.org/hadoop/common/hadoop-2.6.0/hadoop-
2.6.0.tar.gz
```

```
$ tar xzf hadoop-2.6.0.tar.gz
```

```
$ sudo mv hadoop-2.6.0 hadoop
```

```
$ sudo chown -R hadoop /home/hadoop/hadoopdata/
```

```
$ cd $HADOOP_HOME/sbin/
```

```
$ source ~/.bashrc
```

```
$ nano /home/hadoop/hadoop/etc/core-site.xml
```

```
$ nano /home/hadoop/hadoop/etc/hdfs-site.xml
```

```
$ nano /home/hadoop/hadoop/etc/mapred-site.xml
```

```
$ nano /home/hadoop/hadoop/etc/yarn-site.xml
```

```
$ hdfs namenode -format
```

```
$ start-dfs.sh
```

```
$ start-yarn.sh
```

```
$ jps
```

core-site.xml

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://master:9000/</value>
                <description>NameNode URI</description>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/hadoop/hadoop/tmp/hadoop-
${user.name}</value>
        </property>
</configuration>
```

hdfs-site.xml

```
<configuration>
        <property>
                 <name>dfs.datanode.data.dir</name>
                 <value>file:///home/hadoop/ha-
doopdata/hdfs/datanode</value>
                 <description>DataNode directory</description>
        </property>

        <property>
                 <name>dfs.namenode.name.dir</name>
                 <value>file:///home/hadoop/ha-
doopdata/hdfs/namenode</value>
                 <description>NameNode directory</description>
```

```
        </property>

        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>
        <property>
                <name>dfs.permissions</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.datanode.use.datanode.hostname</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.namenode.http-address</name>
                <value>master:50090</value>
                <description>Your Secondary NameNode hostname
                for http access.</description>
        </property>
    </configuration>
```

yarn-site.xml

```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
```

```
</property>

<property>

        <name>yarn.nodemanager.aux-services.mapre-
        duce_shuffle.class</name>

        <value>org.apache.hadoop.mapred.Shuffle-
        Handler</value>

</property>

<property>

        <name>yarn.resourcemanager.resource-tracker.ad-
        dress</name>

        <value>master:8031</value>

</property>

<property>

        <name>yarn.resourcemanager.scheduler.ad-
        dress</name>

        <value>master:8030</value>

</property>

<property>

        <name>yarn.resourcemanager.address</name>

        <value>master:8032</value>

</property>

<property>

        <name>yarn.resourcemanager.webapp.ad-
        dress</name>

        <value>master:8088</value>

</property>

<property>

        <name>yarn.resourcemanager.admin.address</name>

        <value>master:8033</value>
```

```
        </property>
</configuration>
```

mapred-site.xml

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

```
package WordCount;

import java.io.File;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.commons.lang3.time.StopWatch;
import java.util.StringTokenizer;
import java.util.concurrent.TimeUnit;
import javax.swing.JFileChooser;

public class WordCount1 {

    private static File getFile(){

        JFileChooser chooser = new JFileChooser(".");
        chooser.setDialogTitle("Select File To Count Words:");
        int retval = chooser.showOpenDialog(null);
        File f =null;
        chooser.grabFocus();
        if (retval == JFileChooser.APPROVE_OPTION)
            f = chooser.getSelectedFile();
        return f;
    }
    public static void main(String [] args) throws Exception{
    Configuration c = new Configuration();
    Path output =  new Path("hdfs://master:9000/words.txt");
    Path outputResults =  new Path("hdfs://master:9000/results.txt");
    StopWatch timer = new StopWatch();
    c.set("fs.default.name", "hdfs://master:9000");
    File inputFile = getFile();
    Path input = new Path(inputFile.getAbsolutePath());
    FileSystem hdfs = FileSystem.get(c);
    timer.start();
    hdfs.copyFromLocalFile(input, output);
    Job j= Job.getInstance(c);
    j.setJobName("WordCount");
    j.setJarByClass(WordCount.class);
    j.setMapperClass(MapForWordCount.class);
    j.setReducerClass(ReduceForWordCount.class);
    j.setOutputKeyClass(Text.class);
    j.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(j, output);
    FileOutputFormat.setOutputPath(j, outputResults);
    timer.stop();
    System.out.println("Computation time: " + timer.getTime(TimeUnit.SECONDS) + " s");
    System.exit(j.waitForCompletion(true)?0:1);

    }
```

```
    public static class MapForWordCount
        extends Mapper<LongWritable, Text, Text, IntWritable>{

    public void map(LongWritable key, Text value, Context con
                    ) throws IOException, InterruptedException{
    final IntWritable one = new IntWritable(1);
        Text word = new Text();
            StringTokenizer itr = new StringTokenizer(value.toString().toLowerCase());
            while (itr.hasMoreTokens()){
                word.set(itr.nextToken());
                con.write(new Text (word), one);
        }
    {
      Text outputKey = new Text(word.toString().toUpperCase().trim());
      IntWritable outputValue = new IntWritable(1);
      con.write(outputKey, outputValue);
    }
    }
    }
    public static class ReduceForWordCount
        extends Reducer<Text, IntWritable, Text, IntWritable>{

    public void reduce(Text word, Iterable<IntWritable> values, Context con
                       ) throws IOException, InterruptedException{
        int sum = 0;
        for(IntWritable value : values){
            sum += value.get();
        }
        con.write(word, new IntWritable(sum));
        System.out.println(word.toString() + " " + sum);
        }
    }
}
```