PALVELINKLUSTERIN RAKENTAMINEN DEBIAN-YMPÄRISTÖÖN

Tero Ratilainen

Opinnäytetyö
Marraskuu 2010

Tietojenkäsittely
Luonnontieteiden ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Tekijä(t)
RATILAINEN, Tero

Julkaisun laji
Opinnäytetyö

Päivämäärä
15.11.2010

Sivumäärä
66

Julkaisun kieli
Suomi

Luottamuksellisuus
( ) saakka

Verkkojulkaisulupa myönnetty
(X)

Työn nimi
PALVELINKLUSTERIN RAKENTAMINEN DEBIAN-YMPÄRISTÖÖN

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Työn ohjaaja(t)
TUIKKA, Tommi

Toimeksiantaja(t)
LemonEntry Network Ay

Tiivistelmä
LemonEntry Network Ay:lle tehdyssä opinnäytetyössä tutkittiin palvelinklustereissa käytettyjä tekniikoita ja näiden soveltamista ns. HA-klusterin luomiseksi hosting-palveluille. Tavoitteena oli luoda vikasietoinen ja skaalautuva klusteri, joka toiminnallaan minimoi palvelukatkokset. Ympäristöllä tulisi olemaan toiminnallisuus tasolla, jossa yksittäisen palvelimen vikatilanne ei tuota häiriötä palvelun toimintaan käyttäjän näkökulmasta. Ympäristö toteutettiin Unix-pohjaisella Debian Squeeze -käyttöjärjestelmällä, joka työn tekovaiheessa oli vielä kehitysvaiheensa loppusuoralla. Tämä tekee työn hyödyntämisen mahdolliseksi huomattavasti pienemmillä muutoksilla verrattaen esim. Debian Lennyyn lähitulevaisuudessa, kun uusia palvelimia otetaan käyttöön uudemmalla käyttöjärjestelmäversiolla.

Työstä on tehty käytännössä tekniikkariippumaton, jolloin klusterin yksittäiset komponentit eivät ole riippuvaisia tietystä alla olevasta tekniikasta. Tämä mahdollistaa yksittäisen komponentin vaihtamisen toiseen vastaavaan tekniikkaan niin, että muihin komponentteihin ei tarvitse tehdä suuria muutoksia. Komponentit on myös tarvittaessa mahdollista vaihtaa vaiheittain käyttöön, jolloin muutoksesta ei oikein toteutettuna koidu huoltokatkoksia.

Lopputuloksena työstä saatiin testiympäristöön palvelinklusteri, joka suorittaa kuormantasauksen palvelinten kesken sekä tarkkailee klusterin tilaa automaattisesti. Vikatilanteessa pyynnöt ohjautuvat pelkästään toiminnassa oleville palvelimille ja vikatilanteen päättyessä palvelimet otetaan automaattisesti takaisin käyttöön. Normaalitilanteessa palvelimet synkronoivat tiedot keskenään, jolloin on mahdollista suorittaa palvelinten välillä kuormantasausta sekä saavutetaan tilanne, jossa tiedot ovat vikatilanteen sattuessa saatavilta muilta palvelimilta. Ympäristö on myös skaalautuva – kapasiteetin nostamiseksi riittää uusien palvelimien lisääminen järjestelmään.

Avainsanat (asiasanat)
Palvelimet, klusterit, Linux, Debian, hosting, kuormantasaus

Muut tiedot

Author(s)
RATILAINEN, Tero

Type of publication
Bachelor´s Thesis

Date
15.11.2010

Pages
66

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
DEBIAN-BASED HIGHLY-AVAILABLE CLUSTER

Degree Programme
Business Information Systems

Tutor(s)
TUIKKA, Tommi

Assigned by
LemonEntry Network Ay

Abstract
This Bachelor's thesis for LemonEntry Network Ay researched technologies used in Highly-Available clusters and their implementation in clusters used in Hosting service production. The goal was to produce a Highly-Available cluster, which is scalable and not easily prone to hardware failure. The environment was set up in a manner where a single failure will not cause interruption to the service. The environment was built with Unix-based Debian Squeeze, which has a proposed release date around Christmas 2010 and so will be likely used when the system goes live. This will lessen the work in near future as the thesis is tailored for new software revisions.

The system has been made technology-independent, which makes it possible to change single components of the cluster to utilize different technologies without major changes to other components. The components can also be changed or replaced one at a time without a need for downtime when properly performed.

As a result, a server cluster was built into a testing environment, which performs load balancing and monitors the health of the cluster. Should a server go down, the queries get redirected to live servers and the faulty server stays being monitored for when it returns to service. When the situation has been resolved and the cluster notices that the server is back in service, the server will be automatically connected back to the cluster. In a normal situation, the servers synchronize with each other, making it possible to perform load balancing and having a dependable environment in case of failure. The environment is scalable – all that is needed is to add more servers to the cluster.

Keywords
Server, Cluster, Linux, Debian, Hosting, Highly-Available, Load-Balanced

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

1 LYHENTEET JA TERMIT

Termi	Kuvaus
Bootloader	Ohjelma, jota käytetään käyttöjärjestelmän käynnistykseen tietokoneen käynnistyessä.
Distributed computing	Laitteiden tai resurssien jakaminen useisiin eri sijainteihin ja niiden hyödyntäminen halutun tuloksen saavuttamiseksi.
DRBD	Distributed Replicated Block Device. Tekniikka tiedon synkronointiin verkon ylitse.
GFS2	Global File System 2. Palvelinklustereissa käytetty tekniikka tiedostojärjestelmien jakamiseen.
GlusterFS	Palvelinklustereissa käytetty tekniikka tiedostojärjestelmien jakamiseen.
Hosting	Palveluiden (esim. www-sivut, sähköpostit yms.) tarjoaminen palvelimelta toisille osapuolille.
iSCSI	Internet Small Computer System Interface. Tekniikka tietojen varmuuskopiointiin verkon yli.
Konfigurointi	Laitteen tai ohjelman saattaminen tilaan, jossa se on toimintakykyinen.
Locale	Käyttöjärjestelmässä tai ohjelmassa käytetty kieli sekä merkistökoodausmenetelmä.
Lokien kierrätys	Lokitiedostojen säännöllinen uudelleennimeäminen ja pakkaaminen. Suoritetaan ylläpidon työn helpottamiseksi, sillä tiedostoissa on tiedot tietyltä

	aikaväliltä. Lokien kierrätys myös rajoittaa lokitiedostojen koon inhimillisemmäksi.
MBR	Master Boot Record. Kovalevyn ensimmäinen varausyksikkö.
Metatieto	Tietoa toisesta tiedosta.
NFS	Network File System. Protokolla tiedostojen ja kansioiden jakamiseen tietoverkon ylitse.
NTP	Network Time Protocol. Käytetään tietokoneiden kellonajan ajantasaisena pitämiseen.
Pakettivarasto	Linux-käyttöjärjestelmässä käytetty yleensä internetissä sijaitseva paikka, josta on mahdollista asentaa ohjelmia tai ohjelmien tarvitsemia ohjelmistokomponentteja.
Palvelinklusteri	Ryhmä palvelimia, jotka toimivat yhdessä ennalta määritellyn tuloksen saavuttamiseksi.
Pingaus	Ping-komennolla verkkoyhteyden tai verkkolaitteen toiminnan testaaminen.
Redundanttinen	Suorittaa samaa tai vastaavaa toimintoa kuin joku tai jokin toinen osa organisaatiota. Yleensä nähdään turhana resurssien tuhlauksena, tässä yhteydessä kuitenkin luoden lisäresursseja sekä parantaen vikasietoisuutta.
Root-käyttäjä	Pääkäyttäjä *nix-pohjaisissa järjestelmissä.
SCSI	Small Computer System Interface
Single-point-of-failure	Yksittäinen kohta teknisessä järjestelmässä, joka

	rikkoutuessaan tekee koko järjestelmän toimintakyvyttömäksi.
SWAP	Menetelmä tietokoneen muistin määrän keinotekoiseen lisäämiseen. Käytetään välttämään tilanteita, joissa fyysinen muisti ei riitä ohjelman suorittamiseksi.
Synkronointi	Tietojen ajantasaisuuden varmistaminen kahdessa tai useammassa eri paikassa ja niiden päivittäminen.
VirtualHost	Apachessa käytettävä ns. virtuaalipalvelin, jolla voidaan rajata eri www-sivut omiin lohkoihinsa.
Virtuaalisointi	Palvelinympäristön luominen useista eri laitteista, tai useiden eri palvelinympäristöjen luominen yksittäiseen laitteeseen, joka toimii kuten se toimisi ympäristöä vastaavassa laitteessa. Mahdollistaa esim. yksittäisen laitteen resurssien jakamisen tai useiden eri laitteiden resurssien yhdistämisen.

Taulukko 1 Lyhenteet ja termit

2 LÄHTÖKOHDAT

LemonEntry Network Ay on IT-alan yritys, joka tarjoaa palvelinratkaisuja sekä asiantuntijapalveluita. Asiakasmäärien kasvu aiheuttaa yrityksen palvelimille kasvupaineita ja siksi yritys on kehittämässä käyttöönsä uutta, vikasietoisempaa ja suorituskykyisempää järjestelmää nykyisten hosting-ratkaisujen korvaajaksi. Myös jatkossa nousevat resurssitarpeet tulevat vaatimaan suurempia resursseja, joten ratkaisun tarvitsee paitsi kattaa nykyiset ja lähitulevaisuuden tarpeet, sen tulee myös olla helposti päivitettävissä vastaamaan tulevaisuuden tarpeita. Tämän opinnäytetyön tarkoituksena on kehittää järjestelmä, jota on mahdollista käyttää nykyisen ratkaisun päivittämiseksi mahdollisimman pienellä lisätyöllä.

Opinnäytetyö rajattiin www- ja tietokantapalveluiden toteutukseen. Työssä tutkittiin eri malleja ja tekniikoita ratkaisun toteuttamiseksi pitäen kuitenkin järjestelmä mahdollisimman yksinkertaisena ylläpidon helpottamiseksi. Yhtenä tavoitteena työssä on myös saattaa palvelut erilleen toisistaan, jolloin on mahdollista hallita palveluita omina kokonaisuuksinaan sekä saattaa ne mahdollisuuksien mukaan riippumattomiksi toisistaan. Ratkaisun tulisi myös olla helposti käyttöönotettavissa nykyisten ratkaisujen näkökulmasta. Työn laajuutta tämä rajaa esimerkiksi siinä suhteessa, että nykyinen ympäristö rajaa käytettävät alustat yhteensopiviin vaihtoehtoihin sekä mahdollisesti on jo valmiiksi konfiguroitu käyttöön – työssä siis pidättäydytään ratkaisun luomiseen liittyvään tutkimukseen. Vikasietoisuuden saavuttamiseksi järjestelmästä luodaan redundanttinen. Käytännössä tämä tarkoittaa kaikkien palveluiden monistamista niin, että jokaista palvelua on tarjoamassa kaksi tai useampi palvelin. Palvelut myös hajautetaan erilleen toisistaan, jolloin palvelulle saadaan tuotettua korkeampi suorituskyky ja tietoturva. Lopullisena tavoitteena on siis käytännössä luoda klusteroitu hosting-ratkaisu, jolta löytyy sekä suorituskykyä että vikasietoisuutta. Tämän saavuttamiseksi opinnäytetyössä selvitettiin, millainen palvelinklusteri yrityksen tarpeisiin soveltuu parhaiten ja kuinka sellainen toteutetaan. Kyseessä on siis kehittämistutkimus, jossa käytetyt menetelmät ovat käytännössä olleet tekniikoiden tutkinta ja sovellus, joiden avulla kehitettiin toimiva järjestelmä saadun tiedon mukaisesti soveltamalla ja testaamalla järjestelmää.

3 PALVELINKLUSTERI

3.1 Mitä ovat palvelinklusterit

Oxford Dictionaries:n (n.d.) mukaan *”Klusteri on ryhmä samankaltaisia elementtejä tai ihmisiä, jotka ovat lähekkäin tai ilmenevät lähekkäin toisiaan”*. Palvelinklusterin tapauksessa tämä tarkoittaa sitä, että ryhmä palvelimia toimivat yhdessä luoden kokonaisuuden, joka näkyy käyttäjälle yhtenä palvelimena. Yleisesti ottaen klusterit voivat tarjota käyttäjille suuremman suorituskyvyn sekä paremman vikasietoisuuden yksittäisiin palvelinratkaisuihin verrattuna. Alla muutamia erilaisia kategorioita:

3.1.1 HA-klusteri

HA eli High Availability -klusterin tarkoitus on tarjota mahdollisimman suuri toimivuus järjestelmälle. Järjestelmä suunnitellaan alun pitäen sellaiseksi, että siinä ei ole ns. single-point-of-failure -kohtia lainkaan. Tämä käytännössä toteutetaan sillä, että käytössä on olevat järjestelmät ovat vähintäänkin kahdennettuja (jokaisesta toimenpiteestä vastaa kaksi identtistä palvelinta, jotka synkronoidaan keskenään reaaliajassa). Tämä parantaa palvelun luotettavuutta, saatavuutta sekä palvelualttiutta. Tällaisessa järjestelmässä yhden palvelimen rikkoutuminen ei vaikuta merkittävästi järjestelmän toimivuuteen, jolloin järjestelmä on toimintakykyinen myös vikatilanteessa. Vikatilanne ei myöskään näy asiakkaalle, sillä järjestelmä pysyy edelleen toimintakykyisenä ja ongelma on mahdollista korjata ilman palvelukatkoksia. (Quintero ym. 2004, 117.)

3.1.2 Kuormantasausklusterit

Kuormantasausklustereissa on tarkoituksena tasata liikenne tasaisesti useammalle palvelimelle yhden sijasta. Näin saavutetaan parempi toimivuus tilanteissa, joissa palveluun kohdistuu suuri määrä liikennettä samanaikaisesti. Tässä ratkaisumallissa useampi palvelin suorittaa samaa tehtävää samoilla taustaresursseilla, näkyen käyttäjälle yhtenä palvelimena. Tuloksena on suurempi suorituskyky, jolloin on mahdollista tarjota palveluita suuremmalle asiakasmäärälle. (Powers ym. 2002, 4.)

3.1.3 Laskennalliset klusterit

Laskennallisessa klusterissa useita laitteita yhdistetään suorittamaan suurempaa laskutoimitusta. Hyvänä esimerkkinä tällaisesta klusterista on Folding@Home, joka tutkii proteiinien taittumista ja sen liittymistä erilaisiin sairauksiin, kuten syöpiin, BSE:hen, Alzheimerin tautiin, Parkinsonin tautiin ja moneen muuhun sairauteen. Tutkimukseen tarvitaan erittäin suuri määrä laskentatehoa, eikä sellaisen saavuttaminen ole usein järkevää yhdellä laitteistolla teknisistä sekä rahallisista syistä. Hajauttamalla laskenta suuremmalle määrälle edullisempia laitteita voidaan saada riittävä määrä laskentatehoa käyttöön ilman kalliita yksittäisinvestointeja. (Topolsky, 2007.)

3.1.4 Grid computing

Grid computing mahdollistaa avointen standardien avulla itsenäiset ja fyysisesti erillään olevat resurssit toimimaan suuren virtuaalisen tietokoneen tavoin. Mallina se kykenee tarjoamaan dynaamisia virtuaalisia järjestelmiä, jotka tarjoavat turvallisen ja tarkasti koordinoidun tiedonkulun heterogeenisten resurssien välillä. Näitä resursseja voivat olla esimerkiksi ohjelmistot, data sekä fyysiset resurssit kuten suorituskykyä, verkkoliikennettä tai levytilaa. Käyttäjälle nämä resurssit näkyvät yhtenä suurena virtuaalisena tietokoneena. Tällöin eri toiminnot on jaoteltuna omiin laitteisiinsa, jolloin voidaan saavuttaa parempi tietoturva ja suorituskyky. (Ferreira ym. 2005, 4.)

Grid computing on yksi ns. distributed computing -malli, joka käytännössä tarkoittaa sitä, että käytetty laitteisto voi sijaita useissa eri paikoissa ja jopa useiden eri valtioiden ja organisaatioiden alueella. Nämä resurssit voivat myös olla virtualisoituna yhden laitteen sisällä, jolloin voidaan hyödyntää laitteen resurssit tehokkaammin. (Mts. 4-7.)

Grid computingia voidaan hyödyntää sekä palvelinpuolella (jolloin toiminnallisuus keskittyy hajauttamaan esim. Laskutoimituksia tai resursseja useammille laitteille), että työasemapuolella (jolloin toiminnallisuus keskittyy työasemien käyttämättömäksi jäävän laskentatehon hyödyntämiseen). Mallin mukaan suunniteltu järjestelmä voidaan myös tarvittaessa suunnitella toteuttamaan ratkaisu useaan eri tarpeeseen, jolloin se koostuu useammista eri komponenteista, jotka kukin hoitavat oman osaamisalueensa tehtävät. (Mts. 7-11.)

3.2 Tiedostojärjestelmäklusterointitekniikat

3.2.1 iSCSI

iSCSI eli **Internet Small Computer System Interface** on tekniikka, joka mahdollistaa SCSI-komentojen ajamisen IP-pohjaisten verkkojen yli. Käytettävänä kohteena voi olla joko verkkokiintolevy tai toinen tietokone. Tämä käytännössä mahdollistaa levyn jakamisen verkon ylitse niin kuin se olisi kytkettynä suoraan laitteeseen, sekä myös kloonaamaan SCSI-komennot verkon yli, jolloin saadaan kahdennettu levytila. Näin tieto on useammalla eri laitteella, eikä ole yhden laitteen toiminnasta riippuvainen. (Linux-iSCSI Project, 2004.) iSCSI:n kohdalla on kuitenkin otettava huomioon se, että levytilaan voidaan tallentaa kerrallaan ainoastaan yhdestä palvelimesta. Useammasta kohteesta tiedon samanaikainen tallentaminen saattaa korruptoida tietoja. (QNAP Systems, Inc., n.d.)

3.2.2 DRBD

DRBD on tekniikka, joka sijoittuu käyttöjärjestelmässä tiedostojärjestelmän ja levyohjaimen väliin. DRBD välittää levykomennot sekä paikalliselle kovalevylle, että verkon ylitse toissijaiselle levypalvelimelle. Tekniikka hyödyntää kahta palvelinta, joista toinen toimii ensisijaisena levypalvelimena ja toinen toissijaisena. Tiedot tallentuvat molempiin paikkoihin reaaliajassa, luoden kahdennetun levypalvelun. RAID-tekniikkaa hyödyntämällä yhdessä DRBD:n kanssa voidaan välttyä myös osalta ongelmista, jotka kohdistuvat kumpaan tahansa levypalvelimeen, sillä yksittäisen levyn tuhoutuessa kyseinen levy voidaan vaihtaa ilman, että koko palvelimen toiminta häiriintyy. Palvelimet käytännössä vahtivat toistensa tilaa ja toimintaa reaaliajassa, tietäen palvelun tilan ja reagoiden ongelmatilanteisiin välittömästi. Vikatilanteen tapahtuessa rikkoutunut palvelin siirtyy välittömästi pois käytöstä, jolloin toinen toiminnassa oleva palvelin siirtää palvelun itselleen kunnes vika on korjattu. Vikatilanteen selvittyä palvelimet kytketään taas yhteen ja synkronoidaan keskenään niin, että tilanne palautuu takaisin vikatilannetta edeltävään tilaan – eli kahteen palvelimeen, joilla on molemmilla samat tiedot. (LINBIT HA-Solutions GmbH, n.d.)

3.3 Yhteenveto klusterityypeistä

Tyyppi	Tarkoitus	Ympäristö
Highly Available -klusteri	Korkea tavoitettavuus, minimoidut katkokset	Palvelin
Kuormantasausklusteri	Suuri suorituskyky, resurssien tasainen käyttö	Palvelin
Laskennallinen klusteri	Suuri laskentateho, matalammat kustannukset	Palvelin, työasema
Grid computing	Suuri suorituskyky, resurssien hajauttaminen	Palvelin, työasema

Taulukko 2 Klusterityyppien vertailu

Kuten aiemmin todettu, toteutukseen tarvitaan hosting-käyttöön soveltuva palvelinklusteri. Laskentatehoa klusterissa tarvitaan, mutta varsinaista lukujen murskausta ei kuitenkaan suoriteta, joten laskennallinen klusteri ei ole tarpeellinen. Sen sijaan, klusterin tulisi olla mahdollisimman tavoitettavissa, tähän soveltuu HA-klusteri erinomaisesti. HA-klusterissa itsessään on liiketoiminnan näkökulmasta yksi huono puoli – käytössä olevat varapalvelimet ovat käytännössä käyttämättöminä normaalitilanteessa, joten ne eivät tuota mitään rahallisesta näkökulmasta. Ne ovat kuitenkin liiketoiminnan kannalta välttämättömiä, sillä ne tuottavat palvelun mikäli ensisijaiset palvelimet rikkoutuvat. Mikäli tähän liitetään kuormantasausklusteri, saadaan varapalvelimet hyötykäyttöön myös normaalitilanteessa. Näin nekin ovat tuottavia sekä vähentävät yksittäisen palvelimen kuormaa antaen klusterille paremman suorituskyvyn. Tällaisen klusterin kanssa huomioon tulee ottaa kuitenkin yksittäisen palvelimen rikkoutumisen seuraukset klusterille – klusterin tilaa tulee

valvoa jatkuvasti ja palvelimia lisätä jo ennen kuin yhden palvelimen merkityksellisyys klusterissa kasvaa liian suureksi. Kuormantasauksen vuoksi palvelut näkyvät yhtenä suurena järjestelmänä käyttäjälle – tässä suhteessa järjestelmä täyttää myös ilmenemisensä ja toimintansa muodon vuoksi grid computing:n tunnusmerkit. Ratkaisu on siis käytännössä hybridi – siihen yhdistetään usea eri klusterointitekniikka mahdollisimman suuren hyötysuhteen saavuttamiseksi. Tämä osaltaan lisää ylläpidon työtä (palvelinten valvonta, ylläpitotoimet jne.), mutta tarjoaa paremman suorituskyvyn palveluille sekä varmentaa palvelimien toiminnan (ongelmat palvelimilla havaitaan nopeasti, sillä ne ovat aktiivisessa käytössä).

3.4 Miksi klusteroida?

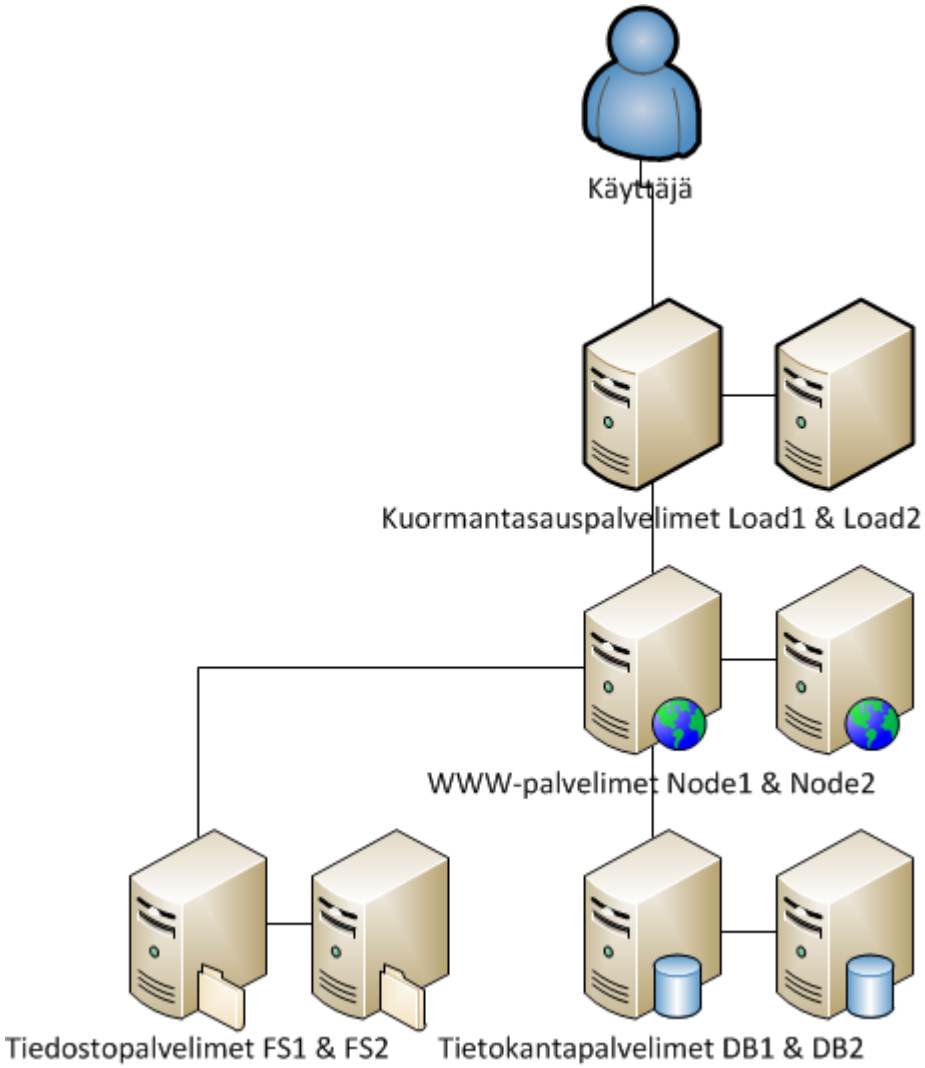
Vaikka palvelimet usein ovatkin tehokkaita, suuremmilla käyttäjämäärillä yksittäisen palvelimen tehot eivät välttämättä riitä antamaan palvelulle riittävää toimivuutta ja joustavuutta. Perinteinen ratkaisu tässä on ollut päivittää palvelin tehokkaammaksi tai siirtää osa palveluista toiselle palvelimelle, jolloin palvelimen kuorma vähenee. Yksittäisen palvelimen päivittämisessäkin tulee kuitenkin ennen pitkää raja vastaan, jolloin asiaan on keksittävä jokin toinen ratkaisu. Yksi tällainen ratkaisu on palvelimien klusterointi, jolloin samaa tehtävää suorittaa useampi palvelin. Tämä myös tuo järjestelmään paremman vikasietoisuuden, sillä yhden palvelimen rikkoutuminen ei vaikuta merkittävästi palvelun toimintaan. Palvelimia on myös helpompi päivittää, sillä usein yhden palvelimen päivittäminen voi olla riskialtista. Mikäli päivitys rikkoo ohjelmiston tai aiheuttaa ongelmia sen käytössä, palveluun tulee palvelukatko. Tämä taas on erittäin huono asia, sillä yritykset usein pyrkivät tarjoamaan mahdollisimman katkottoman palvelun – palvelinklusterilla nämäkin voidaan välttää. Samalla myös voidaan välttää vanhoista ohjelmistoista aiheutuvat uhat – kuten haavoittuvuudet, tietoturva-aukot ja jo korjatut bugit. Lyhyesti siis – järjestelmän palvelualttius kasvaa ja järjestelmän ylläpito helpottuu (vaikka sen määrä ei suinkaan vähene). (Powers ym. 2002, 4-5.)

4 TOTEUTUS

4.1 Vaatimukset

3. luvussa käsitellyistä eri klusterointimalleista tähän työhön soveltuu parhaiten kuormatasattu HA-klusteri, eli näiden kahden klusterimallin hybridi. Tämä siksi, että HA-klusterit tarjoavat korkean vikasietoisuuden ja näin ollen järjestelmä ei ole yhtä herkkä vikatilanteille. Tällaisessa järjestelmässä on kuitenkin huonona puolena se, että yksi tai useampi palvelin on tyhjän panttina sillä aikaa, kun yksi palvelin tekee kaiken työn. Mikäli klusteriin lisätään kuormantasaus, saadaan kaikki palvelimet käyttöön samanaikaisesti. Tämä hyödyntää palvelinten resurssit huomattavasti tehokkaammin sekä tekee järjestelmästä huomattavasti tehokkaamman. Tällöin yritys ei käytännössä maksa turhasta, sillä palvelimet ovat käytössä tarpeiden mukaan. Kuormantasaus mahdollistaa myös järjestelmän helpon päivittämisen, sillä päivitys on mahdollista palvelimia lisäämällä.

Ohjelmistotasolla palvelinklusteri tuotetaan Debianilla, sillä sen tulee olla unix-pohjainen, sekä monet käyttöön soveltuvat ohjelmistot vaativat Linux-ytimen, jolloin esim. BSD-pohjaiset järjestelmät eivät sovellu käyttöön. Yrityksessä on käytetty aiemmin Debian-pohjaisia Linuxeja, joten on järkevää rakentaa järjestelmä tälle pohjalle, jolloin mahdolliset koulutus- yms. kustannukset jäävät mahdollisimman pieniksi. Tässä työssä käytetään Squeezea eli Debianin kehitysversiota sillä se tullaan kirjoitushetken tietojen mukaan julkaisemaan joulun 2010 tienoilla, joten se on erittäin todennäköisesti käytössä varsinaisessa toteutuksessa.



Kuvio 1 Tiedon liikkuminen palvelinklusterissa

Klusterissa tulisi kaikki palvelut olla hajautettuna omiin laitteisiinsa parhaan mahdollisen vikasietoisuuden tavoittamiseksi, joten palvelimille jaetaan omat tehtävät: osa palvelimista toimii kuormantasaajina, toinen osa ns. nodeina, eli ajavat http-palvelinsovellusta. Tässä tapauksessa siis Apachea. Kolmas osa palvelimista ajaa MySQL-tietokantapalvelinsovellusta ja neljäs eli viimeinen osa toimii levypalvelimena. Kuormantasaajat käytännössä jakavat saapuvan liikenteen sille palvelimelle, jolla on pienin kuorma kyseisellä hetkellä. Näin varmistetaan se, että palvelimet jakavat käytön tasaisesti antaen käyttäjille parhaan mahdollisen käyttökokemuksen myös ruuhka-aikoina sen sijaan, että suurin osa kyselyistä päätyisi yhdelle palvelimelle.

Käyttäjältä menee siis http-pyyntö internet-selaimesta kummalle tahansa nodeista, jotka taas käyttävät tietokantoja ja levytilaa omilta erillisiltä palvelimiltaan.

Tietokanta- ja levypalvelimet synkronoivat tietonsa keskenään samaa tehtävää ajavien palvelinten kanssa taaten tiedon ajantasaisuuden sekä toimivuuden myös vikatilanteissa.

4.2 VirtualBox:n asetukset

Toteutus tehdään käyttämällä VirtualBox-virtualisointiohjelmistoa. Samaan laitteeseen luodaan kahdeksan kappaletta virtuaalikoneita, joihin asennetaan Debian Squeeze. Toteutukseen käytetään 64-bittistä versiota, sillä sama prosessoriarkkitehtuuri tulee olemaan varsinaisissa palvelimissa käytössä. Kaikki virtuaalikoneet tullaan asentamaan liitteen 1 mukaisella asetuskokoonpanolla.

Virtuaalikoneet tullaan nimeämään seuraavalla tavalla: Load1, Load2, Node1, Node2, DB1, DB2, FS1 ja FS2. Näistä load1-2 tulevat ajamaan kuormantasausohjelmaa, node1-2 Apache www-palvelinta, DB1-2 MySQL-tietokantapalvelinta ja FS1-2 tarjoamaan levytilaa. IP-osoitteet osoitetaan palvelimille ja palveluille liitteen 2 mukaisella tavalla.

4.3 Debianin asennus

Aloitetaan asennus normaalisti ja valitaan tavallinen asennus, eli ei graafista asennusta. Valitaan kieleksi englanti ja maaksi ”**other -> Europe -> Finland**”. ja Localeksi asetetaan ”**United States**” eli ”**en_US.UTF-8**” ja näppäinkartaksi ”**Finnish**”. Tässä vaiheessa asennus suorittaa laitteiston tunnistuksen sekä lataa lisää ohjelmakomponentteja esim. verkkoon yhdistämisen mahdollistamiseksi. Asennus tulee seuraavaksi kysymään ensisijaista verkkolaitetta, tähän valitaan *ensimmäinen* kohta valikosta, eli ”**eth0**”. Koneen nimeksi syötetään koneen nimi 3.2:ssa määritetyllä tavalla, sekä verkkotunnukseksi unksi.info.

Osiointi suoritetaan *tiedostopalvelimille* seuraavasti: ”**Manual -> tyhjä levyosio -> Create a new partition**”. Seuraavaksi täytyy asettaa osion koko, syötetään ensimmäiselle osiolle kooksi ”**10GB**”. Tyypiksi asetetaan ”**Primary**”. Use as -kohtaan vaihdetaan arvo ”**ext4**”. Lopuksi hyväksytään osio valitsemalla ”**Done setting up the partition**”. Tämän jälkeen luodaan kaksi muuta osiota samalla tavalla kuin edellinen. Kooksi asetetaan ensimmäiselle ”**500MB**” ja toiselle ”**10GB**”. Molemmille asetetaan ”Use as”-kohtaan arvoksi ”**do not use**”. Lopuksi valitaan ”**Finish partitioning and**

write changes to disk”. *Muille palvelimille* osiointi tapahtuu muuten samalla tavalla, paitsi että niille tulee vain yksi osio, joka on koko kovalevyn kokoinen. Seuraaviin kysymyksiin vastataan seuraavasti: ei luoda swap-osiota (tuotantokäyttöön tuleviin palvelimiin on kuitenkin syytä luoda sellaiset, ettei palvelin jumiudu siinä tapauksessa, että muisti syystä tai toisesta loppuu kesken) ja alustetaan luodut osiot. Rootin salasanaksi laitetaan **1234**, uudeksi käyttäjäksi luodaan **unksi** salasanalla **1234**.

Asennus aloittaa seuraavaksi asentamaan itse järjestelmää. Alkuun kysytään muita asennuslevyjä sekä internet-pakettivarastoja, molempiin kysymyksiin vastataan kieltävästi. Pakettienkäyttökyselyyn ei osallistuta. Asennetaan järjestelmä ”**standard system”** ja ”**SSH server”** optioilla. Bootloader asennetaan suoraan MBR:ään. Järjestelmä asentuu nyt loppuun ja käynnistyy hetken kuluttua uudelleen.

4.4 Debianin peruskonfiguraatio

Aluksi poistetaan asennuksessa tullut exim4 -sähköpostipalvelin komennolla ”**apt-get purge exim4***”. Seuraavaksi tarkistetaan pakettivarastot tiedostosta ”**/etc/apt/sources.list**”. Lisätään käyttöön ”**contrib**” -pakettivarastot muuttamalla tiedosto liitteen 3 mukaiseksi. Tämän jälkeen päivitetään järjestelmä komennolla ”**apt-get update && apt-get dist-upgrade**”.

Asetetaan toinen verkkosovitin eli ”**eth1**” aktiiviseksi seuraavalla komennolla: ”**ifconfig eth1 up && ifconfig eth1 192.168.56.120/24**”. Tarkistetaan sovittimen toiminta pingaamalla isäntäkonetta komennolla: ”**ping 192.168.56.1**”. IP-osoitteet on syytä asettaa asennettavalle palvelimelle osoitettuihin osoitteisiin edellisessä ja seuraavassa kohdassa, kuten kuvattu liitteessä 2. Tämän jälkeen ajetaan liitteen 4 mukaiset asetukset tiedostoon ”**/etc/network/interfaces**”, jotta verkko toimii myös uudelleenkäynnistyksen jälkeen.

4.5 Debianin levypalvelinympäristön asentaminen

Kaikki komennot ajetaan molemmille palvelimille (sekä fs1:lle, että fs2:lle), ellei toisin määritetä. Aluksi asennetaan tarvittavat paketit komennolla ”**apt-get install ntp ntpdate drbd8-utils drbdlinks heartbeat nfs-kernel-server**”. NTP tarvitaan, jotta palvelimet ovat varmasti samassa ajassa ja näin kyetään varmistamaan, että asiat

tapahtuvat samanaikaisesti. NFS taasen tarvitaan levytilan jakamiseksi muille palvelimille. Otetaan NFS:n automaattinen käynnistys pois päältä seuraavilla komennoilla: ”**update-rc.d -f nfs-kernel-server remove && update-rc.d nfs-kernel-server stop 20 2 3 4 5 && update-rc.d -f nfs-common remove && update-rc.d stop 20 2 3 4 5**”. DRBD tulee hallitsemaan NFS:n käytön, joten ongelmien välttämiseksi on parempi antaa sen hallita NFS:n tila kokonaan.

Tässä vaiheessa järjestelmässä on perusasennus ja järjestelmä ovat osioituna seuraavalla tavalla:

Osio 1: /dev/sda1, tiedostojärjestelmänä ext4, koko 10 GB, liitetty kohtaan /

Osio 2: /dev/sda2, ei tiedostojärjestelmää, koko 500 MB, tulee DRBD:n metatieto-osioksi

Osio 3: /dev/sda3, ei tiedostojärjestelmää, koko 10 GB, tulee DRBD:n dataosioksi

(LINBIT HA-Solutions GmbH 2010)

Seuraavaksi otetaan DRBD:n moduuli käyttöön komennolla ”**modprobe drbd**” ja luodaan ohjelmien konfigurointi seuraavasti:

Jotta sallitaan verkkojako halutuille palvelimille ja rajataan se kyseiseen verkkoon, lisätään tiedoston ”**/etc/exports**” loppuun seuraava rivi: ”**/data/export 192.168.56.0/255.255.255.0(rw,no_root_squash)**”.

Tiedosto ”**/etc/drbd.conf**” liitteen 5 mukaiseksi. Tästä on syytä huomata, että asetukset rajaavat tiedonsiirron 100 Mbps:ään. Asetus tulee nostaa suuremmaksi palvelinten kapasiteetin mukaan. (Reisner & Ellenberg 2008a)

Tiedosto ”**/etc/ha.d/ha.cf**” liitteen 6 mukaiseksi. On syytä varmistaa, että node -kohdassa on arvo ”**fs1 fs2**”, jotta käytössä on oikeat palvelimet.

Tiedosto ”**/etc/heartbeat/authkeys**” liitteen 7 mukaiseksi. Tähän tiedostoon tulee sama arvo molemmille tiedostopalvelimille, mutta eri kuin muille Heartbeatia käyttäville palvelimille. Eri avaimilla varmistetaan, etteivät palvelimet vahingossakaan aiheuta ristiriitoja keskenään.

Tiedostoon ”**/etc/ha.d/haresources**” seuraavat rivit:

”fs1 IPaddr::192.168.56.130

fs1 drbddisk::r0 Filesystem::/dev/drbd0::/data::ext4 nfs-kernel-server”

Tämä tiedosto tulee molemmille palvelimille täysin samanlaisena, jotta fs1-palvelimesta tulee ensisijainen tiedostopalvelin. (Haas 2010)

Lopuksi asetetaan tiedosto-oikeudet kohdilleen komennolla ”**chmod 600 /etc/heartbeat/authkeys**”, jotta palvelun salasanaa ei voi lukea muut käyttäjät kuin ylläpitäjä. Näillä asetustiedostoilla määritettiin palvelun IP-osoitteeksi 192.168.56.130, joka tulee vaihtumaan tiedostopalvelimien välillä sen mukaan, mikä palvelin on kulloinkin ensisijaisena palvelimena. Näin varmistetaan, että muut palvelimet hakevat aina tiedon oikealta palvelimelta ilman uudelleenkonfiguroinnin tarvetta.

Seuraavaksi luodaan varsinainen levyjärjestelmä. Aluksi luodaan DRBD:n metatieto-osio komennolla ”**drbdadm create-md r0**” ja otetaan se käyttöön komennolla ”**drbdadm up all**” (Reisner & Ellenberg 2008b). Komennolla ”**cat /proc/drbd**” voidaan tarkistaa palvelun toiminta. Tässä vaiheessa molemmat palvelimet näkyvät toissijaisina ja synkronoimattomina, sillä niitä ei ole vielä konfiguroitu eikä synkronoitu.

Seuraavat komennot ajetaan vain FS1-palvelimelle. Asetetaan FS1-palvelin ensisijaiseksi komennolla ”**drbdsetup /dev/drbd0 primary -o**” ja alustetaan sekä liitetään se omalle paikalleen seuraavalla komennolla: ”**mkfs.ext4 /dev/drbd0 && mkdir /data && mount -t ext4 /dev/drbd0 /data**” (Reisner & Ellenberg 2008c). Tämän jälkeen muutetaan NFS:n asetukset niin, että ne siirtyvät FS2:lle vikatilanteessa: ”**mv /var/lib/nfs/ /data/ && ln -s /data/nfs/ /var/lib/nfs**”. Luodaan myös kansio liitettäväksi muille palvelimille komennolla **”mkdir /data/export**”. Varmistetaan kansion oikeudet komennolla ”**chmod 777 /data/export**” ja irroitetaan tiedostojärjestelmä järjestelmästä komennolla **”umount /data**”. Lopuksi tarkistetaan tilanne ja todetaan palvelimen olevan ensisijainen tiedostopalvelin komennolla ”**cat /proc/drbd**”.

Seuraavat komennot ajetaan vain FS2-palvelimelle. Valmistellaan FS2-palvelin vikatilannetta varten luomalla tarvittavat kansiot ja linkitykset komennolla: ”**mkdir /data && rm -rf /var/lib/nfs && ln -s /data/nfs /var/lib/nfs**”. Tämä komento luo kansion, johon tiedostojärjestelmä liitetään tarvittaessa sekä ohjaa NFS:n käyttämään FS1:ltä saatuja asetuksia.

Lopuksi ajetaan seuraavat komennot molemmille palvelimille palveluiden käynnistämiseksi: ”**/etc/init.d/drbd start**” ja ”**/etc/init.d/heartbeat start**”.

4.6 Node-palvelimien asennus ja konfigurointi

4.6.1 NFS:n käyttöönotto

Ensinnä kytketään käyttöön NFS, jotta saadaan levypalvelimet käyttöön ja sivustot sekä asetukset jaettua eri palvelimien kesken. Tämä saavutetaan asettamalla ”**/etc/fstab**” -tiedoston loppuun seuraava pätkä: ”**192.168.56.130:/data/export /share nfs rw,soft,intr,rsize=8192,wsize=8192**” (Sean 2008; SourceForge 2006). On myös syytä luoda jaon käyttämä kansio komennolla ”**mkdir /share**”. Toimivuus varmistetaan käynnistämällä palvelin uudelleen ja toteamalla, että kyseinen kansio on liitettynä tiedostojärjestelmään uudelleenkäynnistyksen jälkeen komennolla ”**ls -la /share**”.

4.6.2 Apachen ja PHP:n asennus

Seuraavaksi asennetaan tarvittavat ohjelmistot eli Apache -http-palvelin ja PHP. Asennus tapahtuu kätevästi suorittamalla komento ”**apt-get install apache2 apache2-suexec libapache2-mod-php5 php5 php5-cgi php5-mysql php5-suhosin php5-mcrypt**”. Seuraavaksi suoritetaan Apachen ja PHP:n konfigurointi vain Node1-palvelimella, sillä Node2:n konfigurointi suoritetaan myöhemmässä vaiheessa käyttämällä Node1:n asetuksia.

Jotta lokit muodostuvat oikein kuormantasaajien kanssa, on LogFormat -kohdista vaihdettava pätkä ”**%h**” pätkään ”**%{X-Forwarded-For}i**”. Tämä asetus löytyy asetustiedostosta ”**/etc/apache2/apache2.conf**”. Kyseinen tiedosto löytyy liitteestä 8. Ilman tätä muutosta lokeissa näkyy aina vierailijan kohdalla kuormantasaajan tiedot muuttaen lokit osittain hyödyttömiksi, sillä kävijöitä ei voida erottaa toisistaan.

Seuraavaksi testataan toimivuus. Aluksi käskytetään Apachea lataamaan asetukset uudestaan, jotta saadaan PHP käyttöön komennolla ”**/etc/init.d/apache2 reload**”. Oletuksena Apache luo oletussivun kansioon ”**/var/www**”. Poistetaan kansiossa oleva ”**index.html**”-tiedosto ja korvataan se tiedostolla ”**index.php**”, johon laitetaan seuraava lause: ”**<? phpinfo(); ?>**”. Tämän jälkeen selaimeen laitettaessa osoite http://192.168.56.100 aukeaa PHP:n versiotietosivu, josta näkee PHP:n asetukset ja versiotiedot. Tämä tarkoittaa sitä, että kaikki toimii.

4.6.3 Konfiguraation jakaminen verkon ylitse

Kohdassa 3.6.1 otettiin käyttöön verkkojako sekä Node1- että Node2-palvelimille liitoskohtaan /share. Käytännössä siirretään tiedostot verkkojaolle sekä luodaan linkitykset niihin vanhasta sijainnista. Seuraavaksi kyseiset tiedostot poistetaan Node2:lta ja linkitetään jaosta löytyviin tiedostoihin.

Tämän saavuttamiseksi siirretään Node1:ltä Apachen ja PHP:n asetukset, joka tapahtuu komennolla ”**mv /etc/apache2 /share/etc && ln -s /share/etc/apache2 /etc/apache2 && mv /etc/php5 /share/etc && ln -s /share/etc/php5 /etc/php5**” ja toiseksi www-sivut komennolla ”**mv /var/www /share && ln -s /share/www /var/www**”. Tämän jälkeen käynnistetään Apache varmuuden vuoksi uudestaan komennolla ”**/etc/init.d/apache2 restart**” ja testataan www-selaimella toimivuus em. tavalla. Tämän jälkeen asetetaan Node2 käyttämään näitä tiedostoja komennolla ”**rm -rf /etc/apache2 && ln -s /share/etc/apache2 /etc/apache2 && rm -rf /etc/php5 && ln -s /share/etc/php5 /etc/php5 && rm -rf /var/www && ln -s /share/www /var/www**” ja käynnistetään Apache uudestaan komennolla ”**/etc/init.d/apache2 restart**”. Todennetaan toimivuus www-selaimella osoitteessa http://192.168.56.101.

4.6.4 SSL:n konfigurointi

Tähän tarvitsee kaksi asiaa, SSL-moduulin Apachelle ja sertifikaatin. Moduuli on jo asennettuna, se vain tarvitsee ottaa käyttöön. Tämä tapahtuu komennolla ”**a2enmod ssl**”. Sertifikaatin luonti onnistuu komennolla ”**openssl genrsa -out cert.key && openssl req -new -key cert.key -out cert.csr && openssl x509 -req -days 365 -in cert.csr -signkey cert.key -out cert.crt**”. Tämä käytännössä luo ns. ”private key”:n, sertifikaatin ja allekirjoittaa sertifikaatin luomallaan avaimella. Asetetaan nämä

tiedostot kansioon ”**/etc/apache2**”. Tietoturvasyistä on tärkeää, että nämä tiedostot ovat luettavissa vain root-käyttäjälle. Seuraavaksi kopioidaan Apachessa oleva VirtualHost ja vaihdetaan sen portiksi 443. Tähän uuteen VirtualHost:iin lisätään seuraavat rivit:

”SSLEngine on

SSLCertificateFile /etc/apache2/cert.crt

SSLCertificateKeyFile /etc/apache2/cert.key”.

Nämä asetukset löytyvät tiedostosta ”**/etc/apache2/sites-enabled/000-default**”, joka löytyy liitteestä 9. Tämän jälkeen käynnistetään Apache uudestaan molemmilta node-palvelimilta komennolla ”**/etc/init.d/apache2 restart**”. Tämän jälkeen https://192.168.56.100 toimii selaimessa.

4.7 Tietokantapalvelimien asennus ja konfigurointi

Tietokantapalvelimelle riittää MySQL:n asentaminen ja konfigurointi. Asennus tapahtuu komennolla ”**apt-get install mysql-server**”. Tämän jälkeen tulee sallia tietokantayhteydet muilta palvelimilta. Ensimmäinen askel on asettaa palvelinprosessi kuuntelemaan IP-osoitetta, joka on muiden palvelimien tavoitettavissa. Tämä tapahtuu asettamalla asetustiedostoon **/etc/mysql/my.cnf** seuraava rivi: ”**bind-address = 192.168.56.110**”. Tähän tulee tietysti palvelinkohtainen IP-osoite. Tämän jälkeen palvelin tulee käynnistää uudelleen komennolla **/etc/init.d/mysql restart**. Toiseksi tulee antaa root-käyttäjälle oikeus kirjautua muualtakin kuin palvelimelta itseltään, tämä tapahtuu seuraavasti: Kirjoitetaan komento ”**mysql -p**”. Tässä muodossa ohjelma kysyy rootin salasanaa sen sijaan, että se kirjoitetaan komentoriville ja päätyy lokitiedostoihin. Salasanan syötettyä annetaan ohjelmalle seuraava rivi: ”**GRANT ALL PRIVILEGES ON *.* TO root@’%’ IDENTIFIED BY ’1234’ WITH GRANT OPTION;**”. Tässä on huomattava, että tämä antaa mahdollisuuden kirjautua roottina mistä tahansa, myös muualta, kuin käytössä olevilta palvelimilta. Tämä on tietoturvariski, joka on suositeltavaa korjata joko määrittelemällä tähän IP-osoitteet erikseen tai estämällä palvelimelle pääsy muualta kuin käytössä olevilta palvelimilta. Seuraavaksi syötetään komento ”**FLUSH PRIVILEGES;**”, joka vahvistaa edellisellä komennolla tehdyt muutokset. Nyt

palvelimelle on mahdollista kirjautua root-käyttäjällä palvelimen ulkopuolelta esim. phpMyAdminin kautta. Tässä vaiheessa on suositeltavaa asentaa phpMyAdmin jommalle kummalle node-palvelimelle ja testata rootilla kirjautumista molempien node-palvelimien kautta. Tietojen siirtoa varten täytyy tehdä asiaa varten erillinen käyttäjä. Tämän voi suorittaa esim. phpMyAdminista samalla, kun palvelinten toiminta testataan. Laitetaan käyttäjälle nimeksi ”**replication_user**” ja ”**replication slave**” sekä ”**replication client**” -oikeudet. (Maxia 2006)

Seuraavaksi asetetaan palvelimet synkronoimaan tietokannat keskenään, jolloin saavutetaan vikasietoisuus. Kyseessä on siis ns. Master-Master -asennus, jossa molemmat tietokantapalvelimet ottavat muutoksia vastaan ja päivittävät ne keskenänsä. Tällaisessa tilanteessa ongelmatilanteessa tarvitsee vain vaihtaa käytettävän palvelimen osoite, mikäli yksi palvelimista poistuu käytöstä. Paremman tuloksen tähän saa lisäämällä väliin kuormantasaajan, jolloin ongelmatilanne ei edes näy käyttäjälle asti.

Aluksi varmistetaan, että molemmilla palvelimilla on samat tiedot – eli kopioidaan DB1 -palvelimelta kaikki tiedot DB2 -palvelimelle. Tämä tapahtuu suorittamalla DB1-palvelimella komento ”**mysqldump --all-databases --master-data > dbdump.db -p**”, jolloin saadaan varmuuskopio kaikesta palvelimen sisällöstä, mukaanlukien palvelimen hallinnollinen data. Tämä tiedosto tulee siirtää DB2-palvelimelle esim. SFTP:n kautta, jossa suoritetaan komento ”**mysql < dbdump.db -p**”, joka ajaa tiedot sisään. Koska käytössä on Debian, on muistettava ottaa huomioon järjestelmän huoltotoimenpiteet, jotka suoritetaan erillisellä käyttäjällä. Tiedostosta ”**/etc/mysql/debian.cnf**” täytyy kopioida salasana DB1-palvelimelta DB2-palvelimelle. Mikäli tätä ei tehdä, tulee DB2-palvelimelle ongelmia, kuten esim. MySQL-palvelimen käynnistys, sammutus ja lokien kierrätys eivät tule toimimaan. DB1-palvelimelle tulee asettaa seuraavat rivit ”**[mysqld]**”-kohtaan liitteen 10 mukaiset rivit. DB2-palvelimelle tulevat vastaavasti liitteen 11 mukaiset rivit samaan paikkaan. MySQL:n täysi asetustiedosto palvelimelta DB1 löytyy liitteestä 12. (Maxia 2006)

Tämän jälkeen molempien palvelinten MySQL-prosessit käynnistetään uudestaan komennolla ”**/etc/init.d/mysql restart**”. Tämän jälkeen käydään phpMyAdminista

katsomassa molemmilta palvelimilta tiedot ”**Replication**”-tabilta. Molemmat palvelimet näyttävät itsensä masterina sekä toisen palvelimen slavena. Mikäli jompi kumpi tai molemmat palvelimet eivät näy vielä muutaman minuutin odottelun jälkeen slavena tai antavat virheilmoituksia, on syytä painaa ”**Full start**”-linkkiä tai syöttää toisen palvelimen tiedot uudestaan.

Palvelinten toimintaa on hyvä testata esim. asentamalla SMF -keskustelupalstaohjelmisto. Onnistuneessa asennuksessa ohjelman tietokanta löytyy molemmilta palvelimilta samanlaisena, käytti sitten kumpaa palvelinta tahansa.

4.8 Kuormantasauspalvelimien konfigurointi

Kuormantasaukseen valittiin käytettäväksi ohjelmaksi Pound, sillä se on kevyt ja pieni ohjelma, joka tarjoaa myös SSL- ja istuntotuen. Näin ollen esim. verkkokauppasovellukset eivät hajoa kuormantasauksen vuoksi. Poundin asennus tapahtuu helposti komennolla ”**apt-get install pound**”. Asennuksen jälkeen avataan Poundin asetustiedosto ”**/etc/pound/pound.cfg**”. Käytännössä ainoat asiat, mitä tiedostosta tarvitsee muuttaa, on vaihtaa palvelimen sisäisten osoitteiden tilalle palvelimen oma ulkoinen osoite sekä node-palvelimien osoitteet. Mikäli halutaan asettaa https käyttöön, tarvitsee myös kopioida ”**ListenHTTP**” kohta ”**ListenHTTPS**” -kohdaksi sekä asettaa tarvittavat sertifikaatit käyttöön. (Apsis Gmbh, n.d.)

Tässä on syytä huomata, että Pound purkaa https-liikenteen ja se jatkaa salaamattomana eteenpäin. Asetustiedosto tulee identtisenä molemmille palvelimille. Tiedosto on liitteessä 13. Seuraavaksi on syytä avata tiedosto ”**/etc/default/pound**” ja vaihtaa sinne arvo ”**startup=1**” tiedoston loppuun. Tämän jälkeen Pound on mahdollista käynnistää komennolla ”**/etc/init.d/pound start**”. Juuri vaihdettu vipu on varmistuskeino siihen, ettei kuormantasauspalvelin tahattomasti käynnisty ilman konfigurointia.

Jotta molemmat kuormantasaajat saadaan toimimaan vikasietoisesti, tarvitaan Heartbeat, aivan kuten tiedostopalvelinten kanssa. Heartbeatin saa asennettua komennolla ”**apt-get install heartbeat**”. Tämän jälkeen luodaan asetukset seuraavasti:

Tiedosto luodaan ”**/etc/ha.d/ha.cf**” liitteen 6 mukaisesti, vaihtamalla asetuksen ”**node**” arvoksi ”**load1 load2**”.

Tiedosto ”**/etc/heartbeat/authkeys**” liitteen 7 mukaiseksi. Tähän tiedostoon on syytä luoda erillinen avain tiedostopalvelimiin nähden, jottei palvelinten välille voi vahingossakaan tulla ristiriitaa.

Tiedosto **/etc/ha.d/haresources** seuraavanlaiseksi:

load1 IPaddr::192.168.56.131

Tämä tiedosto tulee molemmille palvelimille täysin samanlaisena, jolloin load1-palvelimesta tulee ensisijainen tiedostopalvelin. Muutoin palvelimet katsovat itsensä ensisijaisiksi palvelimiksi molemmissa tapauksissa. (Haas 2010)

Lopuksi asetetaan tiedosto-oikeudet kohdilleen komennolla ”**chmod 600 /etc/heartbeat/authkeys**”, jotta palvelun salasanaa ei voi lukea muut kuin ylläpitäjä. Näillä asetustiedostoilla määritettiin palvelun IP-osoitteeksi 192.168.56.131, joka tulee vaihtumaan kuormantasauspalvelimien välillä sen mukaan, mikä palvelin on kulloinkin ensisijaisena palvelimena. Näin varmistuu se, että palvelu toimii, vaikka kumpi tahansa palvelimista olisi epäkunnossa. Tähän IP-osoitteeseen tulee ohjata mahdolliset palvelussa käytettävät domain-nimet, jotta ne ohjautuvat järjestelmässä oikein. Yksittäisen kuormantasauspalvelimen IP-osoitteen käyttö saattaa aiheuttaa ongelmia tapauksessa, jossa viitattu palvelin on tavoittamattomissa.

5 TESTAUS

Järjestelmää on testattu järjestelmän toteutusvaiheessa. Tämä on varmistanut yksittäisten komponenttien toiminnan sekä todentanut asetuksiin tehtävät muutokset välittömästi muutosten astuessa voimaan. Näin säästetään aikaa, sillä tilanteita ei pääse tapahtumaan, joihin on tehty suuri määrä muutoksia ja mahdollisessa ongelmatilanteessa ei välttämättä ole selvää, mikä tai mitkä muutoksista aiheuttavat ongelmat. Sen sijaan kyseessä on aina yksittäinen muutos, jolloin ongelman löytäminen, tutkiminen ja ratkaisu nopeutuu ja helpottuu. Toteuttaessa on myös testattu aina arkkitehtuurissa seuraavan järjestelmän toimintaa edellisten kanssa, jolloin järjestelmien yhteistoiminta varmistuu.

Lopuksi valmiilla järjestelmällä on testattu SMF-keskustelupalstaohjelmiston asennusta sekä käyttöä, jolloin testatessa tulevat yhdellä kertaa kaikki käytössä olevat palvelimet ja palvelut käyttöön. Tämä on myös testinä lähinnä sitä käyttöä, missä järjestelmä tulee olemaan käyttöönoton jälkeen. Testauksen aikana on myös kokeiltu sammuttaa yksittäisiä palvelimia kesken kaiken, jolloin nähdään palvelimen katoamisen, eli vikatilanteen vaikutus palveluun. Testeissä huomattiin muutamien sekuntien viiveitä, kun palvelimia ajettiin alas – ohjaukset palvelimien välillä siis toimivat ripeästi havaittuaan vikatilanteen eikä tästä aiheudu merkittävää haittaa palvelun toiminnalle. Palvelut myös havaittiin palautuvan muutaman sekunnin viiveellä normaaliksi sitä mukaa, kun palvelimet nostettiin ylös.

6 KEHITYSIDEOITA

Apache-palvelimien tietoturva ja toiminnallisuus on oletusasetusten tasolla. Niiden konfigurointi on suoritettu ainoastaan tasolle, jolla niiden toimivuus voidaan todeta – on siis varsin suositeltavaa siirtää asetukset jo olemassa olevista, optimoiduista ja suojatuista palvelimista näihin palvelimiin kokoonpanoa käyttöönotettaessa. Tämä on tehty tarkoituksenmukaisesti, jotta vältytään tekemästä sama työ useampaan kertaan.

Palvelimille ei myöskään ole laitettu palomuureja – tämä on välttämätöntä tuotantoympäristössä. Palvelimille tulee laittaa yrityksessä käytössä olevat palomuuriratkaisut ennen käyttöönottoa. Palomuurissa on otettava huomioon se, että tutkimuksessa ei ole rajattu MySQL-palvelimeen pääsyä. Tämä on erittäin suositeltavaa tehdä palomuurilla – tämä estää samalla myös käyttäjän omat virheet niissä tapauksissa missä oikeuksia ei rajata riittävän tehokkaasti.

Tietokantapalvelimista saadaan parempi hyöty irti asettamalla ne kuormantasaajan taakse. Näin saadaan käytettyä palvelinten resurssit paremmalla hyötysuhteella sekä minimoidaan palveluille aiheutuvat häiriöt. Tällä kokoonpanolla asettaisin node-palvelimien hosts-tiedostoon määrätyn osoitteen, esim. db.domain.com osoittamaan db1-palvelimelle, jolloin kaikki tietokantapyynnöt kohdistuvat vain toiseen tietokantapalvelimeen. Vikatilanteessa osoite voidaan tarvittaessa kääntää db2-palvelimelle, joka käytännössä toimii identtisesti db1-palvelimeen nähden. Näihin palvelimiin ei ole järkevää asentaa Heartbeatia, sillä mikäli tarvitaan parempaa ratkaisua, on järkevämpi sijoittaa esim. kuormantasaaja node-palvelimille.

Tiedostopalvelinten suhteen tulee ottaa huomioon, että NFS:ää ei ole tässä työssä juurikaan optimoitu. Tämä johtuu siitä, että NFS käyttäytyy eri tavoin eri laitteisto-, verkko- ja ohjelmistokokoonpanoissa, eikä sitä ole näin ollen mahdollista optimoida järkevästi luodussa ympäristössä. Optimointityö tulee suorittaa palvelimilla, joissa suorituskyky on mahdollista testata. Resurssien tarpeen kasvaessa voi myös olla ennen pitkää hyödyllistä ottaa tehokkaampi ratkaisu käyttöön tiedostojen jakamiseen, kuten esim. GlusterFS tai GFS2.

7 POHDINTA

Työssä luotiin HA-palvelinklusteri Debian 6.0 Squeeze -alustalle. Klusteri tarjoaa käytännössä perusluontoiset webhosting-palvelut, eli www- ja tietokantapalvelut, sekä erillisen tietovaraston. Aluksi selvitettiin, millainen järjestelmä on sopivin tarvittuun toteutukseen. Tulokseksi päädyttiin eräänlaiseen hybridiratkaisuun, jossa sovelletaan sekä Highly Available -klusteria sekä kuormantasausklusteria, muodostaen tavallaan eräänlaisen grid computing-ratkaisun. Näin saadaan hyödynnettyä molempien klusterimallien hyvät puolet sekä saadaan palvelinten resurssit paremmin hyödynnettyä, sillä varapalvelimia ei käytännössä ole eikä tarvita sillä ne ovat jatkuvassa käytössä. Palvelun tavoitettavuus ei kärsi järjestelyistä mikäli palvelinten käyttöastetta seurataan ja suorituskykyä lisätään jo ennen kuin kaikista palvelimista muodostuu palvelulle välttämättömiä resurssitarpeiden vuoksi.

Lopputuloksena saatu järjestelmä toimii juuri niin kuin pitääkin – klusterille tulevat pyynnöt jakautuvat eri palvelimille tasaisesti, mutta kuitenkin niin, että esim. istunnot toimivat yksittäisten palvelinten ja käyttäjien välillä. Klusteri myös reagoi vikatilanteisiin suunnitellulla tavalla – vikatilannetta simuloitaessa, esim. sammuttamalla yksittäisiä palvelimia, palvelu jatkaa toimintaansa käyttäjän näkökulmasta normaalilla tavalla. Palvelinten päästä tutkittaessa huomataan lokitiedostoista, että palvelupyynnöt ohjautuvat ainoastaan toiminnassa oleville palvelimille välittömästi, kun palvelin on havaittu kadonneeksi klusterista. Lokitiedostoista myös ilmenee, että palvelimet tarkkailevat jatkuvasti toisiaan suunnitellulla tavalla ja suorittavat oikeat toimenpiteet palvelinten kadotessa ja ilmaantuessa toimintaan. Palvelimet myös synkronoivat tietonsa keskenään vikatilanteiden jälkeen, jolloin palvelimet palautuvat vikatilannetta edeltäneeseen tilaan – eli tilanteeseen, jossa molemmilla palvelimilla on identtiset tiedot. Ympäristöstä ilmenee, että järjestelmä on mahdollista toteuttaa suhteellisen yksinkertaisesti ja toimivasti. Järjestelmää on myös mahdollista laajentaa tarpeiden mukaan palvelimia lisäämällä ja/tai päivittämällä. Helppo laajentaminen myös helpottaa vastaavalla tavalla vanhan laitteiston poistoa – se voidaan vain napata pois käytöstä uusien laitehankintojen jälkeen sillä kaikki palvelimella olevat tiedot löytyvät muualta järjestelmästä eikä palvelin näin ollen ole kriittinen toiminnalle.

Toteutuksen luomiseksi on tarjolla useita erilaisia tekniikoita ja tämä on vain yksi esimerkki siitä, kuinka tällaisen järjestelmän pystyy toteuttamaan. Järjestelmänä tämä on kuitenkin hyvä siinä suhteessa, että haluttaessa tästä järjestelmästä on mahdollista vaihtaa mikä tahansa komponentti käyttämään eri tekniikkaa ilman, että muihin klusterin komponentteihin tarvitsee tehdä suuria muutoksia. Tämä tekee järjestelmästä suhteellisen helpon päivittää, sillä päivitys on mahdollista suorittaa parhaimmassa tapauksessa vaiheittain, jolloin huoltokatkoja ei tarvita. Tämä helpottaa suuresti asiakkaiden elämää, sillä palveluihin ei tule katkoksia, sekä palveluntarjoajan elämää, sillä palvelut voidaan pitää toiminnassa myös järjestelmähuoltojen ja -päivitysten aikana.

Mikäli palvelut tuotetaan virtualisoituna, voidaan päivityksiä suorittaa vieläkin helpommin – palvelut voidaan helposti siirtää pienitehoisemmalta palvelimelta tehokkaampaan tai tarvittaessa käyttää useampaa palvelinta yhdellä laitteistolla. Samalla myös palvelinten kloonaaminen helpottuu sillä palvelin voidaan kopioida toiselle laitteistolle ja muuttaa pelkästään tarvittavat palvelinkohtaiset asetukset, jolloin vältytään uusilta asennuksilta. Tämä myös säästää testaukseen käytettävää aikaa, sillä palvelimen toiminta on ennalta tiedossa ja näin on tarpeellista testata vain muuttuneet osat – eli käytännössä palvelimen perustoiminnallisuus. Virtuaalisointia käyttäessä myös helpottuu vanhan palvelinlaitteiston poisto järjestelmästä vieläkin helpommin, sillä palvelimella pyörivät palvelinohjelmat voidaan siirtää suoraan toiselle laitteelle, jolloin ainut palvelulle näkyvä ero on palvelimen muuttuneet resurssit.

8 LÄHTEET

Apsis Gmbh. n.d. Pound – Reverse proxy and load-balancer. Viitattu 25.10.2010. http://www.apsis.ch/pound.

Ferreira, L., Batista, M., Fibra, S., Lee, C. Y., Silva, C. A. Q., Almeida, J., Lucchese, F. & Keung, N. 2005. Grid Computing Products and Services. San José, CA: IBM.

Haas, F. 2010. Chapter 3. Creating an initial Heartbeat configuration. Viitattu 10.10.2010. http://www.linux-ha.org/doc/ch-initial-config.html.

LINBIT HA-Solutions GmbH. 2010. Chapter 5. Configuring DRBD. Viitattu 10.10.2010. http://www.drbd.org/users-guide-emb/ch-configure.html.

LINBIT HA-Solutions GmbH. n.d. DRBD. What is DRBD. Viitattu 21.9.2010. http://www.drbd.org/.

Linux-iSCSI Project. 2004. Linux-iSCSI Project. Viitattu 21.9.2010. http://linux-iscsi.sourceforge.net/.

Maxia, G. 2006. Advanced MySQL Replication Techniques. Viitattu 20.10.2010. http://onlamp.com/pub/a/onlamp/2006/04/20/advanced-mysql-replication.html.

Oxford Dictionaries, n.d. Definition of Cluster from Oxford Dictionaries Online. Viitattu 1.9.2010. http://oxforddictionaries.com/view/entry/m_en_gb0157260.

Powers, S., Andersen, E. D., Nee, S., Salmon, D., Sethi, S. & Walkky, L. 2002. Clustering and IASPs for Higher Availability on the IBM eServer iSeries Server. Rochester, MN: IBM.

QNAP Systems, Inc. n.d. How to create and use the iSCSI target service on the QNAP NAS. Viitattu 21.9.2010. http://www.qnap.com/pro_application.asp?ap_id=135.

Quintero, D., Braunbeck, T., Thye, O. S., Vlad, A. & Zutenis, P. Cluster Systems Management Cookbook for pSeries. Austin, TX: IBM.

Reisner, P. & Ellenberg, L. 2008. drbd.conf – Configuration file for DRBD's services. Man-sivu. Kyseessä on man-sivu DRBD.CONF(5).

Reisner, P. & Ellenberg, L. 2008. drbdadm – Administration tool for DRBD. Man-sivu. Kyseessä on man-sivu DRBDADM(8).

Reisner, P. & Ellengerg, L. 2008. drbdsetup – Setup tool for DRBD. Man-sivu. Kyseessä on man-sivu DRBDSETUP(8).

Topolsky, J. 2007. Folding@Home recognized by Guinness World Records. Viitattu 1.9.2010. http://www.engadget.com/2007/10/31/folding-home-recognized-by-guinness-world-records/.

Sean, 2008. Re: nfs mounts don’t work through fstab. Sähköpostiviesti 20.2.2008. Vastaanottaja FreeBSD-current -sähköpostilista. http://unix.derkeiler.com/Mailing-Lists/FreeBSD/current/2008-02/msg00608.html.

SourceForge, 2006. 5. Optimizing NFS Performance. Viitattu 15.10.2010. http://nfs.sourceforge.net/nfs-howto/ar01s05.html.

Zend Technologies Ltd, 2010. Zend Server Cluster Manager. Viitattu 1.9.2010. http://www.zend.com/products/server-cluster-manager/.

9 LIITTEET

9.1 Liite 1: VirtualBox:n asetukset virtuaalipalvelimille

- Keskusmuisti: 256 MB
- Kovalevy: 20 GB, SATA, Dynaamisesti kasvava levykuva, host I/O -välimuisti käytössä molemmilla levyohjaimilla
- Prosessori: 1 kpl
- Näytönohjaimen muisti: 9 MB
- Äänikortti: ei käytössä
- Verkkokortti: Adapteri 1 sillattuna, Intel PRO/1000 T Server 82543GC, kaapeli kytketty, Adapteri 2 Host-only -asetuksella, Intel PRO/1000 T Server 82543GC, kaapeli kytketty
- Sarjaportit: ei käytössä
- USB-portit: ei käytössä
- IO APIC käytössä
- Järjestelmän kello UTC-aikavyöhykkeellä
- VT-x/AMD-V käytössä
- Nested Paging käytössä

9.2 Liite 2: Virtuaalipalvelinten verkko-osoitteet

DB1: 192.168.56.110

DB2: 192.168.56.111

FS1: 192.168.56.120

FS2: 192.168.56.121

Load1: 192.168.56.50

Load2: 192.168.56.51

Node1: 192.168.56.100

Node2: 192.168.56.101

NFS-palvelu: 192.168.56.130

Kuormantasauspalvelu: 192.168.56.131

Aliverkon peite: 255.255.255.0

9.3 Liite 3: Asetustiedosto /etc/apt/sources.list

#

deb cdrom:[Debian GNU/Linux testing _Squeeze_ - Official Snapshot amd64 CD Binary-1 20100920-04:41]/ squeeze main

deb http://ftp.fi.debian.org/debian/ squeeze main contrib

deb-src http://ftp.fi.debian.org/debian/ squeeze main contrib

deb http://security.debian.org/ squeeze/updates main contrib

deb-src http://security.debian.org/ squeeze/updates main contrib

9.4 Liite 4: Asetustiedosto /etc/network/interfaces

```
# This file describes the network interfaces available on your system

# and how to activate them. For more information, see interfaces(5).


# The loopback network interface

auto lo

iface lo inet loopback


# The primary network interface

allow-hotplug eth0

iface eth0 inet dhcp


allow-hotplug eth1

iface eth1 inet static

address 192.168.56.51

netmask 255.255.255.0
```

9.5 Liite 5: DRBD:n asetustiedosto /etc/drbd.conf

```
global {
usage-count no;
}
common {
 syncer { rate 100M; }
}

resource r0 {
 protocol C;
 handlers {
  pri-on-incon-degr "echo o > /proc/sysrq-trigger ; halt -f";
  pri-lost-after-sb "echo o > /proc/sysrq-trigger ; halt -f";
  local-io-error "echo o > /proc/sysrq-trigger ; halt -f";
 }

 startup {
  degr-wfc-timeout 120;    # 2 minutes.
 }

 disk {
  on-io-error   detach;
```

```
  }

  net {
  }

  syncer {
    rate 100M;
    al-extents 257;
  }

on fs1 {
    device /dev/drbd0;
    disk /dev/sda3;
    address 192.168.56.120:7788;
    meta-disk /dev/sda2[0];
  }

on fs2 {
    device /dev/drbd0;
    disk /dev/sda3;
    address 192.168.56.121:7788;
    meta-disk /dev/sda2[0];
```

```
    }

}
```

9.6 Liite 6: Heartbeat:n asetustiedosto /etc/ha.d/ha.cf

```
logfacility local0

keepalive 1

deadtime 10

bcast eth1

auto_failback on

node fs1 fs2
```

9.7 Liite 7: Heartbeat:n autentikointitiedosto /etc/heartbeat/authkeys

```
auth 3

3 md5 81dc9bdb52d04dc20036dbd8313ed055
```

9.8 Liite 8: Apachen asetustiedosto /etc/apache2/apache2.conf

```
#

# Based upon the NCSA server configuration files originally by Rob McCool.

#

# This is the main Apache server configuration file.  It contains the

# configuration directives that give the server its instructions.

# See http://httpd.apache.org/docs/2.2/ for detailed information about

# the directives.

#

# Do NOT simply read the instructions in here without understanding

# what they do.  They're here only as hints or reminders.  If you are unsure

# consult the online docs. You have been warned.

#

# The configuration directives are grouped into three basic sections:

#  1. Directives that control the operation of the Apache server process as a

#     whole (the 'global environment').

#  2. Directives that define the parameters of the 'main' or 'default' server,

#     which responds to requests that aren't handled by a virtual host.

#     These directives also provide default values for the settings

#     of all virtual hosts.

#  3. Settings for virtual hosts, which allow Web requests to be sent to
```

```
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "foo.log"
# with ServerRoot set to "/etc/apache2" will be interpreted by the
# server as "/etc/apache2/foo.log".
#


### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#


#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
```

```
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation (available
# at <URL:http://httpd.apache.org/docs/2.2/mod/mpm_common.html#lockfile>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
#ServerRoot "/etc/apache2"


#
# The accept serialization lock file MUST BE STORED ON A LOCAL DISK.
#
LockFile ${APACHE_LOCK_DIR}/accept.lock


#
# PidFile: The file in which the server should record its process
# identification number when it starts.
# This needs to be set in /etc/apache2/envvars
#
PidFile ${APACHE_PID_FILE}


#
```

```
# Timeout: The number of seconds before receives and sends time out.

#

Timeout 300



#

# KeepAlive: Whether or not to allow persistent connections (more than

# one request per connection). Set to "Off" to deactivate.

#

KeepAlive On



#

# MaxKeepAliveRequests: The maximum number of requests to allow

# during a persistent connection. Set to 0 to allow an unlimited amount.

# We recommend you leave this number high, for maximum performance.

#

MaxKeepAliveRequests 100



#

# KeepAliveTimeout: Number of seconds to wait for the next request from the

# same client on the same connection.

#

KeepAliveTimeout 15
```

```
##

## Server-Pool Size Regulation (MPM specific)

##


# prefork MPM

# StartServers: number of server processes to start

# MinSpareServers: minimum number of server processes which are kept spare

# MaxSpareServers: maximum number of server processes which are kept spare

# MaxClients: maximum number of server processes allowed to start

# MaxRequestsPerChild: maximum number of requests a server process serves

<IfModule mpm_prefork_module>

   StartServers         5

   MinSpareServers      5

   MaxSpareServers     10

   MaxClients          150

   MaxRequestsPerChild   0

</IfModule>


# worker MPM

# StartServers: initial number of server processes to start

# MaxClients: maximum number of simultaneous client connections
```

```
# MinSpareThreads: minimum number of worker threads which are kept spare
# MaxSpareThreads: maximum number of worker threads which are kept spare
# ThreadLimit: ThreadsPerChild can be changed to this maximum value during a
#           graceful restart. ThreadLimit can only be changed by stopping
#           and starting Apache.
# ThreadsPerChild: constant number of worker threads in each server process
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule mpm_worker_module>
   StartServers         2
   MinSpareThreads      25
   MaxSpareThreads      75
   ThreadLimit          64
   ThreadsPerChild      25
   MaxClients           150
   MaxRequestsPerChild   0
</IfModule>


# event MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads: minimum number of worker threads which are kept spare
# MaxSpareThreads: maximum number of worker threads which are kept spare
```

```
# ThreadsPerChild: constant number of worker threads in each server process
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule mpm_event_module>
    StartServers         2
    MaxClients         150
    MinSpareThreads      25
    MaxSpareThreads      75
    ThreadLimit          64
    ThreadsPerChild      25
    MaxRequestsPerChild   0
</IfModule>

# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

#
# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.  See also the AllowOverride
# directive.
#
```

```
AccessFileName .htaccess


#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy all
</Files>


#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value.  If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain
```

```
#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off


# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog ${APACHE_LOG_DIR}/error.log


#
# LogLevel: Control the number of messages logged to the error_log.
```

```
# Possible values include: debug, info, notice, warn, error, crit,

# alert, emerg.

#

LogLevel warn



# Include module configuration:

Include mods-enabled/*.load

Include mods-enabled/*.conf



# Include all the user configurations:

Include httpd.conf



# Include ports listing

Include ports.conf



#

# The following directives define some format nicknames for use with

# a CustomLog directive (see below).

# If you are behind a reverse proxy, you might want to change %h into %{X-
Forwarded-For}i

#
```

```
LogFormat "%v:%p %{X-Forwarded-For}i %l %u %t \"%r\" %>s %O \"%{Referer}i\"
\"%{User-Agent}i\"" vhost_combined

LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-
Agent}i\"" combined

LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %O" common

LogFormat "%{Referer}i -> %U" referer

LogFormat "%{User-agent}i" agent


# Include of directories ignores editors' and dpkg's backup files,

# see README.Debian for details.


# Include generic snippets of statements

Include conf.d/


# Include the virtual host configurations:

Include sites-enabled/
```

9.9 Liite 9: Apachen oletus-VirtualHost SSL-asetuksin tiedostosta /etc/apache2/sites-enabled/000-default

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost



    DocumentRoot /var/www

    <Directory />

        Options FollowSymLinks

        AllowOverride None

    </Directory>

    <Directory /var/www/>

        Options Indexes FollowSymLinks MultiViews

        AllowOverride None

        Order allow,deny

        allow from all

    </Directory>



    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

    <Directory "/usr/lib/cgi-bin">

        AllowOverride None

        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch

        Order allow,deny
```

```
        Allow from all
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog ${APACHE_LOG_DIR}/access.log combined

  Alias /doc/ "/usr/share/doc/"
  <Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
  </Directory>

</VirtualHost>
```

```
<VirtualHost *:443>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>
```

```
    ErrorLog ${APACHE_LOG_DIR}/error.log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined

  Alias /doc/ "/usr/share/doc/"
  <Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
  </Directory>

SSLEngine on
SSLCertificateFile /etc/apache2/cert.crt
SSLCertificateKeyFile /etc/apache2/cert.key
</VirtualHost>
```

9.10 Liite 10: DB1-palvelimen palvelinkohtaiset MySQL:n asetukset asetustiedostoon /etc/mysql/my.cnf

Nämä asetukset menevät ”**[mysqld]**”-kohtaan:

server-id = 1

replicate-same-server-id = 0

auto-increment-increment = 2

auto-increment-offset = 1

master-host = 192.168.56.111

master-user = replication_user

master-password = Np9VvhKdNFtzZqub

master-connect-retry = 60

log-bin = mysql-bin

log-error = mysql-bin.err

report-host = db1.unksi.info

9.11 Liite 11: DB2-palvelimen palvelinkohtaiset MySQL-asetukset asetustiedostoon /etc/mysql/my.cnf

Nämä asetukset menevät ”**[mysqld]**”-kohtaan:

server-id = 2

replicate-same-server-id = 0

auto-increment-increment = 2

auto-increment-offset = 2

master-host = 192.168.56.110

master-user = replication_user

master-password = Np9VvhKdNFtzZqub

master-connect-retry = 60

log-bin = mysql-bin

log-error = mysql-bin.err

report-host = db2.unksi.info

9.12 Liite 12: DB1-palvelimen MySQL-asetustiedosto /etc/mysql/my.cnf

```
#

# The MySQL database server configuration file.

#

# You can copy this to one of:

# - "/etc/mysql/my.cnf" to set global options,

# - "~/.my.cnf" to set user-specific options.

#

# One can use all long options that the program supports.

# Run program with --help to get a list of available options and with

# --print-defaults to see which it would actually understand and use.

#

# For explanations see

# http://dev.mysql.com/doc/mysql/en/server-system-variables.html



# This will be passed to all mysql clients

# It has been reported that passwords should be enclosed with ticks/quotes

# escpecially if they contain "#" chars...

# Remember to edit /etc/mysql/debian.cnf when changing the socket location.

[client]

port        = 3306
```

```
socket        = /var/run/mysqld/mysqld.sock


# Here is entries for some specific programs

# The following values assume you have at least 32M ram


# This was formally known as [safe_mysqld]. Both versions are currently parsed.

[mysqld_safe]

socket        = /var/run/mysqld/mysqld.sock

nice          = 0


[mysqld]

#

# * Basic Settings

#

user          = mysql

pid-file      = /var/run/mysqld/mysqld.pid

socket        = /var/run/mysqld/mysqld.sock

port          = 3306

basedir       = /usr

datadir       = /var/lib/mysql

tmpdir        = /tmp

language      = /usr/share/mysql/english
```

```
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 192.168.56.110
#
# * Fine Tuning
#
key_buffer              = 16M
max_allowed_packet      = 16M
thread_stack            = 192K
thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
myisam-recover          = BACKUP
#max_connections        = 100
#table_cache            = 64
#thread_concurrency     = 10
#
# * Query Cache Configuration
#
query_cache_limit       = 1M
```

```
query_cache_size       = 16M
#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
#general_log_file       = /var/log/mysql/mysql.log
#general_log            = 1
#
# Error logging goes to syslog due to /etc/mysql/conf.d/mysqld_safe_syslog.cnf.
#
# Here you can see queries with especially long duration
#log_slow_queries       = /var/log/mysql/mysql-slow.log
#long_query_time = 2
#log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
server-id              = 1
replicate-same-server-id       = 0
```

```
auto-increment-increment       = 2

auto-increment-offset   = 1

master-host           = 192.168.56.111

master-user           = replication_user

master-password       = Np9VvhKdNFtzZqub

master-connect-retry    = 60

log-bin               = mysql-bin

log-error             = mysql-bin.err

report-host           = db1.unksi.info

#log_bin                    = /var/log/mysql/mysql-bin.log

expire_logs_days       = 10

max_binlog_size        = 100M

#binlog_do_db          = include_database_name

#binlog_ignore_db      = include_database_name

#

# * InnoDB

#

# InnoDB is enabled by default with a 10MB datafile in /var/lib/mysql/.

# Read the manual for more InnoDB related options. There are many!

#

# * Security Features

#
```

```
# Read the manual, too, if you want chroot!

# chroot = /var/lib/mysql/

#

# For generating SSL certificates I recommend the OpenSSL GUI "tinyca".

#

# ssl-ca=/etc/mysql/cacert.pem

# ssl-cert=/etc/mysql/server-cert.pem

# ssl-key=/etc/mysql/server-key.pem




[mysqldump]

quick

quote-names

max_allowed_packet      = 16M


[mysql]

#no-auto-rehash # faster start of mysql but no tab completition


[isamchk]

key_buffer              = 16M
```

```
#
# * IMPORTANT: Additional settings that can override those from this file!
#   The files must end with '.cnf', otherwise they'll be ignored.
#
!includedir /etc/mysql/conf.d/
```

9.13 Liite 13: Kuormantasauspalvelu Pound:n asetustiedosto /etc/pound/pound.cfg

```
## Minimal sample pound.cfg

##

## see pound(8) for details




######################################################################
#

## global options:


User            "www-data"

Group           "www-data"

#RootJail       "/chroot/pound"


## Logging: (goes to syslog by default)

##      0       no logging

##      1       normal

##      2       extended

##      3       Apache-style (common log format)

LogLevel        3
```

```
## check backend every X secs:

Alive        2


## use hardware-accelleration card supported by openssl(1):

#SSLEngine     "<hw>"


# poundctl control socket

Control "/var/run/pound/poundctl.socket"




######################################################################
#

## listen, redirect and ... to:


## redirect all requests on port 8080 ("ListenHTTP") to the local webserver (see "Ser-
vice" below):

ListenHTTP

    Address 192.168.56.131

    Port    80


    ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:

    xHTTP        0
```

```
        Service
            BackEnd
                Address 192.168.56.100
                Port    80
            End
            BackEnd
                Address 192.168.56.101
                Port    80
            End
        End
    End
    ListenHTTPS
        Address 192.168.56.131
        Port    443
        Cert    "/etc/pound/cert.crt"

        ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
        xHTTP         0

        Service
            BackEnd
```

```
            Address 192.168.56.100
            Port    80
        End
        BackEnd
            Address 192.168.56.101
            Port    80
        End
    End
End
```