SAVONIA

Klusteroitu MySQL- ja sovelluspalvelinarkkitehtuuri

Erno Voutilainen

Opinnäytetyö

___. ___. _____ _____

Ammattikorkeakoulututkinto

SAVONIA-AMMATTIKORKEAKOULU **OPINNÄYTETYÖ**
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Erno Voutilainen	
Työn nimi Klusteroitu MySQL- ja sovelluspalvelinarkkitehtuuri	
Päiväys 7.5.2013	Sivumäärä/Liitteet 50/3
Ohjaaja(t) lehtori Kalevi Kolehmainen	
Toimeksiantaja/Yhteistyökumppani(t) Finnish Net Solutions Oy / Janne Huttunen	

Tiivistelmä

Tämän opinnäytetyön tavoitteena oli suunnitella PHP-ohjelmointikielellä toteutetulle asiakashallinnan verkkosovellukselle sovelluspalvelinarkkitehtuurin muutos ja ottaa se käyttöön testiympäristössä. Arkkitehtuurimuutoksen pääasiallisena tarkoituksena oli parantaa verkkosovelluksen vikasietoisuutta. Lisäksi tuli suunnitella ja toteuttaa verkkosovelluksen PHP-istuntojen hallinta arkkitehtuuriin sopivalla tavalla. Työtä varten oli varattu neljä palvelinkonetta, yksi kannettava sekä kaksi reititintä.

Vikasietoisuuden parantaminen toteutettiin käyttämällä MySQL-klusteria sekä DNS Round Robin -tekniikkaa, joiden ansiosta verkkosovelluksen tarvitsemat palvelut saatiin hajautettua useille palvelimille. Palvelimet on mahdollista sijoittaa tuotantokäyttöä varten fyysisesti eri paikkoihin, jolloin järjestelmän vikasietoisuus kasvaa entisestään.

Arkkitehtuurimuutos saatiin pystytettyä testiympäristöön onnistuneesti ja se tullaan ottamaan muutamien muutosten kanssa tuotantokäyttöön sopivan tilanteen tullen.

Avainsanat
MySQL-klusteri, sovelluspalvelinarkkitehtuuri, PHP-istunnot, SSH, vikasietoisuus

SAVONIA UNIVERSITY OF APPLIED SCIENCES **THESIS**
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Erno Voutilainen			
Title of Thesis Clustered MySQL and Application Server Architecture			
Date	7 May 2013	Pages/Appendices	50/3
Supervisor(s) Mr. Kalevi Kolehmainen, Lecturer			
Client Organisation /Partners Finnish Net Solutions Oy / Janne Huttunen			

Abstract

The aim of this thesis was to plan an application server architecture modification for a PHP web application and to implement it into a test environment. The goal of the modification was to improve the fault-tolerance of the web application.

The improvement of fault-tolerance was carried out by deploying a MySQL Cluster and DNS Round Robin technologies. With the help of them it was possible to diversify the services needed by the application to multiple separate servers. For a production environment it is possible to locate the servers to completely separate places to still increase the fault-tolerance. In addition, the PHP session management needed to be considered in an appropriate way for the new architecture.

The modification was implemented into the test environment successfully and it will be transferred (with few more changes) to the production environment when the time is right.

Keywords
MySQL Cluster, application server architecture, PHP session management, SSH, fault-tolerance

Esipuhe

Tämä työ on tehty Finnish Net Solutions Oy:lle syksyn 2012 ja kevään 2013 aikana.

Haluan kiittää Kalevi Kolehmaista sekä Janne Huttusta työn ohjauksesta ja hyvistä neuvoista. Opin teidän ansiostanne tätä työtä tehdessä enemmän, kuin alun perin odotin.

Tahdon lisäksi kiittää rakasta vaimoani kaikesta tuesta ja rohkaisusta, jota sain tämän työn tekemiseen.

Kuopiossa 7.5.2013

Erno Voutilainen

SISÄLTÖ

1 JOHDANTO

Tässä opinnäytetyössä käsitellään MySQL-klusterointia ja sen ympärille liittyvää sovelluspalvelinarkkitehtuuria. Tavoitteena on suunnitella ja luoda uusi palvelinarkkitehtuuri PHP-ohjelmointikielellä toteutetulle verkkosovellukselle.

Työ tehtiin Finnish Net Solutions Oy:lle syksyn 2012 ja kevään 2013 aikana. Työn tavoitteena on parantaa verkkosovelluksen vikasietoisuutta hajauttamalla sen tarvitsemia palveluita useille, toisistaan erillisille palvelimille. Verkkosovelluksen arkkitehtuurimuutokset toteutetaan niitä varten pystytetyn testiympäristön sisällä. Palveluiden hajautukseen käytetään MySQL-klusteria. Työhön liittyy olennaisena osana myös PHP-istuntojen hallinta.

Tämä raportti etenee siinä järjestyksessä, jossa tehtävien käytännön toteutus pääpiirteittäin tapahtui. Luvuissa käydään aluksi läpi tarvittava teoriatieto kyseisestä asiasta, jota seuraa asian käytännön toteutus.

Työn kohteena on selainkäyttöisen, PHP-ohjelmointikielellä toteutetun asiakashallinnan verkkosovelluksen arkkitehtuurimuutos. Sovelluksen palvelinarkkitehtuuri toimii nykyisellään siten, että kaikki ohjelman tarvitsemat palvelut toimivat samalla palvelinkoneella. Samalla palvelinkoneella pyörii siis WEB- ja tietokantapalvelimet sekä PHP-tulkki.

Nykyisen ratkaisun hyvä puoli on se, että ohjelmisto ei vaadi monta palvelinkonetta. Järjestelmää on siitä syystä helppo hallita. Nykyisen ratkaisun huono puoli on se, että yhdeltä palvelimelta vaaditaan korkeaa suorituskykyä. Palvelimen on kyettävä palvelemaan lukuisia käyttäjiä yhtä aikaa. Lisäksi huono puoli on se, että nykyinen ratkaisu ei ole vikasietoinen. Jos palvelintilassa tapahtuu sähkökatko tai Internet yhteys katkeaa, ohjelmistoa ei voida käyttää.

Tämän työn tavoitteena on saada juuri vikasietoisuuteen liittyvät puutteet korjattua. Korjaus suunnitellaan toteutettavaksi palvelinarkkitehtuurin muutoksella. Uudessa arkkitehtuurissa tietokanta- ja WEB-palvelimet sijoitetaan erillisille palvelinkoneille. Tietokantapalvelimissa tullaan lisäksi käyttämään klusteria. Näiden muutosten avulla yhdessä palvelintilassa tapahtuva sähkö- tai Internet-yhteyden katkeaminen ei haittaa palvelun toimintaa.

Järjestelmän loppukäyttäjälle palvelun tulee toimia yhdestä ja samasta verkko-osoitteesta, vaikka järjestelmän toiminnallisuus onkin jaettu fyysisesti eri sijainteihin. Palvelintilojen sähkö- tai verkkoyhteyden palautuessa vikatilanteen jälkeen palvelinten tulee automaattisesti palata mukaan palvelinverkkoon.

2 TYÖN TAUSTAA

Tässä luvussa esitellään lyhyesti yritys, jolle työ tehtiin. Lisäksi luvussa käydään läpi työhön liittyviä taustatietoja, kuten testiympäristön rakenne, työn haasteet sekä opinnäytetyöprojektin eteneminen.

2.1 Finnish Net Solutions Oy

Finnish Net Solutions Oy (FNS) on vuonna 2001 perustettu ohjelmistoalan yritys. Yrityksessä työskentelee tällä hetkellä 19 henkilöä Kuopion, Lohjan ja Espoon toimipisteissä. FNS tarjoaa erilaisia ohjelmistoja ja palveluita eri yrityksille ja organisaatioille. Yrityksille räätälöidyt ohjelmistot toteutetaan selainkäyttöisinä verkkopalveluina. Toteutustapana käytetään ketterän ohjelmistokehityksen menetelmiä sekä avoimen lähdekoodin työkaluja.

2.2 Testiympäristö

Johdannossa kuvattuja arkkitehtuurimuutoksia varten täytyi pystyttää testiympäristö. Testiympäristön käyttöön varattiin neljä testipalvelinta, kaksi reititintä, kytkin ja kannettava tietokone. Testipalvelimiin oli määrä asentaa CentOS Linux-käyttöjärjestelmät. Kahta niistä oli tarkoitus käyttää WEB-palvelimena ja kahta muuta tietokantapalvelimena. Molemmilla WEB-palvelimilla on oma ulkoinen verkkoyhteys ja kaikkien neljän koneen välillä on koneiden yhteinen sisäverkko.

Kummallekin WEB-palvelimelle oli varattu reitittimiä, joiden kautta kumpikin palvelin saa oman ulkoverkon osoitteen. Molemmille varattiin DynDNS-palvelusta oma dynaaminen verkko-osoitteensa. Koska työn lopputuloksena kyseessä olevaa verkkosovellusta piti pystyä käyttämään yhdestä verkko-osoitteesta, yrityksen DNS-palvelimilta varattiin vielä yksi verkko-osoite (*ernontesti.fns.fi*), joka määrättiin osoittamaan kumpaankin DynDNS-palvelusta varattuun verkko-osoitteeseen (luku 3.3).

WEB-palvelinten tehtävä uudessa arkkitehtuurissa oli HTTP-palvelun tarjoaminen sekä verkkosovelluksen pyörittäminen. Näistä syistä WEB-palvelimien tarvitsemia ohjelmistoja olivat Apachen HTTP-palvelin, PEAR sekä PHP-tulkki tarvittavine liitännäisineen (luku 3.2.5). WEB-palvelimet välittivät lisäksi kaikki verkkosovelluksen tietokannankäsittelypyynnöt tietokantapalvelimille. Tietokantapalvelinten tehtävä uudes-

sa arkkitehtuurissa oli ainoastaan tietokannan käsittely. Arkkitehtuurin vikasietoisuus toteutettaisiin käyttämällä MySQL-klusteria (luku 4).

Testiympäristön hallintaan käytettiin kannettavaa tietokonetta, jossa oli Windows 7 -käyttöjärjestelmä. Palvelinkoneiden asennuksen jälkeen palvelimia hallittiin kannettavan kautta käyttämällä PuTTy-nimistä SSH-terminaaliohjelmaa. Klusterin pystytyksen yhteydessä koneelle asennettiin klusteriin liittyvä hallintasolmu, jolloin sen hallinta onnistuisi koneelta käsin. Konetta käytettiin myös verkkosovelluksen testaamiseen ja ohjelmakoodiin tarvittavien muutosten tekemiseen.

2.3 Haasteet

Työhön liittyviä haasteita olivat MySQL-klusterin pystytys eri käyttöjärjestelmien välille (Windows / Linux), verkkosovelluksen tietokannan muokkaaminen klusterille sopivaksi sekä verkkosovelluksen sessioiden hallinnan toteuttaminen.

MySQL-klusteri ei ollut ennen työn aloittamista tuttu, joten pelkkä klusterin asennus ja sen ominaisuudet tarjosivat uuden haasteen. Työtä tehdessä selvisi, että Windows 7 -kannettavaa joutuisi käyttämään myös klusterin osana, mikä lisäsi haastetta entisestään. Klusteriin tutustuttaessa kävi ilmi, että verkkosovelluksen tietokannan rakennetta jouduttaisiin muuttamaan, jotta kantaa voitaisiin käyttää klusterin kanssa.

Suurin haaste työssä oli kuitenkin verkkosovelluksen sessioiden hallinnan toteuttaminen. Ennen arkkitehtuurin muutosta sessioiden hallinnasta ei koitunut ongelmaa, koska sovellus toimi yhden palvelimen alaisuudessa. Arkkitehtuurin muuttaminen siten, että samasta verkko-osoitteesta vastaa satunnaisesti toinen kahdesta WEB-palvelimesta, aiheutti PHP-sessioihin liittyvän ongelman: Mistä *WEB 1* -palvelin tietää, että käyttäjä A on käyttänyt sovellusta hetkeä aiemmin palvelimella *WEB 2*? Sessioiden hallinnasta, ongelmasta ja sen ratkaisusta on kerrottu tarkemmin luvussa 5.

2.4 Projektin eteneminen

Opinnäytetyöprojekti jaettiin neljään osa-alueeseen neljän kuukauden ajalle. Ensimmäisenä tehtävänä oli testiympäristön pystytys, joka sisälsi palvelimien käyttöjärjestelmien asennuksen, reitittimien asetusten säätämisen, verkko-osoitteiden varaamisen sekä tarvittavien palvelinohjelmistojen asentamisen. Toisena tehtävänä oli

MySQL-klusterin pystyttäminen WEB- ja tietokantapalvelimille sekä kannettavalle. Tehtävään kuului myös verkkosovelluksen tietokannan muokkaaminen klusterille sopivaksi. Kolmantena tehtävänä oli PHP-sessioiden hallinnan toteuttaminen ja mahdollisten eri ratkaisuvaihtoehtojen kartoittaminen. Neljäs tehtävä oli uuden arkkitehtuurin testaus sekä työn raportointi.

2.5 Lisätietoja

Luvussa 2.3 mainitsin, että testiympäristön WEB-palvelimia varten varattiin yrityksen DNS-palvelimilta osoite *ernontesti.fns.fi*, joka laitettiin osoittamaan molempiin WEB-palvelimiin. Tähän käytettiin tekniikkaa nimeltä DNS Round Robin. Tätä osa-aluetta en itse toteuttanut käytännössä, koska sen käytännön toteutus tuli valmiina yrityksen puolesta. Esittelen tekniikan kuitenkin teoriatasolla luvussa 3.5, koska se liittyy työhön olennaisena osana.

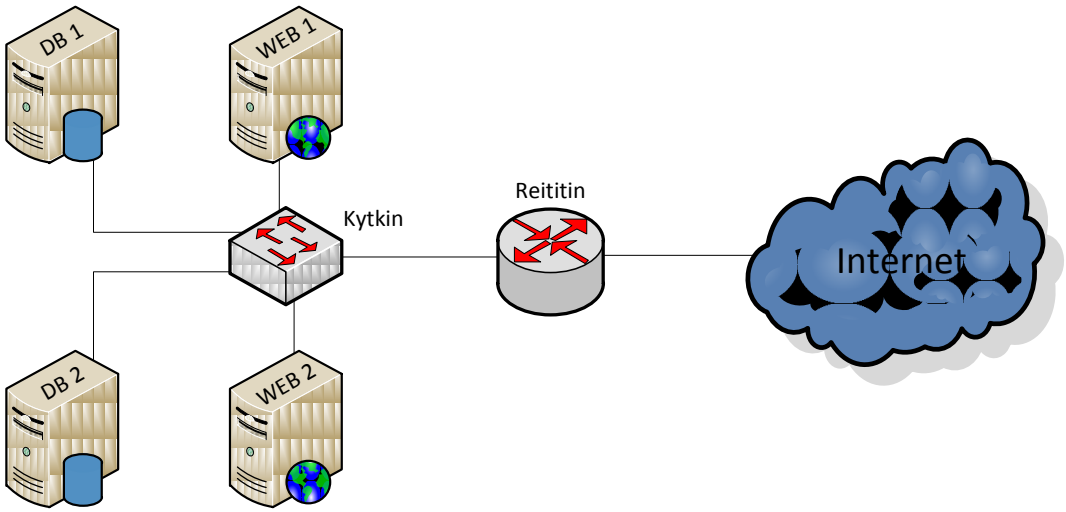
3 TESTIYMPÄRISTÖN PYSTYTYS

Ensimmäinen tehtävä tässä työssä oli testiympäristön pystytys. Työvaiheita olivat käyttöjärjestelmien sekä verkkosovelluksen pyörittämiseen tarvittavien ohjelmistojen asennukset, palomuuriasetusten säätäminen, dynaamisten verkko-osoitteiden varaaminen WEB-palvelimille sekä WEB-palvelinten reitittimien asetusten säätäminen.

3.1 Käyttöjärjestelmän asennus

Kaikkien neljän koneen käyttöjärjestelmäksi tuli asentaa Linuxin CentOS-jakelu. Käyttöjärjestelmän asennus samalla tavalla kaikille koneille, minkä vuoksi esittelen vain yhden koneen asennuksen. Käytin asennuksen käynnistämiseen ns. *Netinstall* CD -levyä, jonka avulla asennuksen voi käynnistää, mutta joka lataa varsinaiset asennustiedostot Internetistä. Jos koko käyttöjärjestelmä olisi asennettu suoraan CD-levyiltä, levyjä olisi täytynyt polttaa 8 kappaletta.

Ennen asennuksen aloittamista tietokoneet kytkettiin verkkoon alla olevan kuvion mukaisesti (kuvio 1), koska kukin kone tarvitsi asennuksen aikana verkkoyhteyden. Kullekin koneelle jaettiin verkkoyhteys konttorin reitittimen yhdestä portista kytkimen kautta. Kukin kone sai tässä tapauksessa reitittimen DHCP:ltä yhden sisäverkon osoitteen.



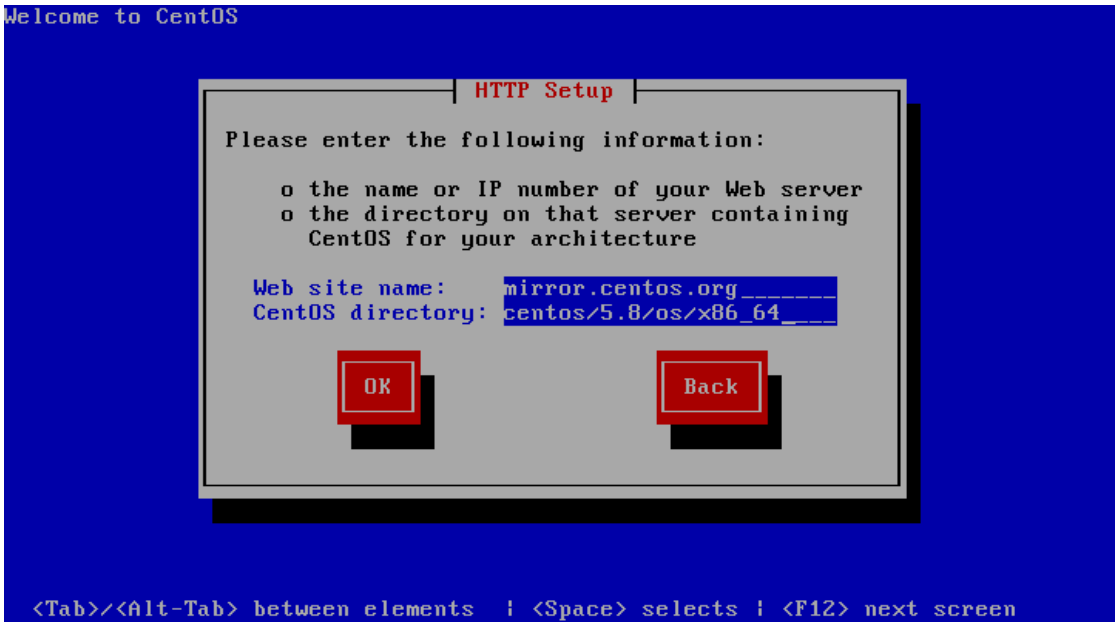
KUVIO 1. Verkon rakenne asennuksen aikana

CentOS-käyttöjärjestelmän asennus käynnistettiin CD-levyltä. Alla on kuva käyttöjärjestelmän asennuksen aloitusvalikosta (kuva 1).

KUVA 1. Aloitusvalikko

Ongelmien välttämiseksi käyttöjärjestelmän asennuksen olisi tässä vaiheessa voinut käynnistää muutamien lisäparametrien kanssa. Joissain laitekokoonpanoissa asennuksen mukana tulevat USB- ja FireWire-ajurit voivat aiheuttaa ongelmia, jolloin käyttöjärjestelmän asennus epäonnistuu. USB- ja FireWire-ajurit voi jättää asennuksesta pois kirjoittamalla yllä olevassa asennusikkunassa komennon *linux nousb nofirewire* ja painamalla Enter. Varsinaisen testiympäristön koneilla asennuksessa ei ilmennyt ongelmia.

Asennuksen käynnistysohjelma kyseli käyttöjärjestelmän asennuksen aikana mm. käytettävää kieltä, näppäimistön kieltä ja asennusmediaa. Asennusmediaksi valittiin HTTP-protokolla, koska asennustiedostot tultaisiin lataamaan Internetistä. Asennusohjelma kysyi verkkosivun nimeä ja kansiota, josta käyttöjärjestelmän asennustiedostot ladataan. CentOS (versio 5.8) -käyttöjärjestelmän 64-bittisen jakelun asennustiedostot löytyvät kuvassa 2 näkyvästä osoitteesta ja kansiosta.



KUVA 2. Asennustiedostojen verkkosijainti

Verkko-osoitteen antamisen jälkeen asennustiedostot alkoivat latautua ja niiden latautumisen jälkeen avautui itse käyttöjärjestelmän asennuksen etusivu (kuva 3).

KUVA 3. Asennuksen aloitus

Koneiden kiintolevyt alustettiin käyttöjärjestelmän asennuksen yhteydessä ja ne osioitiin siten, että käyttöjärjestelmä saa käyttöönsä kaiken tilan. Jokaiselle neljälle koneelle määritettiin eri isäntänimet siten, että tietokantapalvelimet saivat nimen

db1/2.ernontesti.fns.fi ja WEB-palvelimet nimen *web1/2.ernontesti.fns.fi*. Koneiden isäntänimien määrityksellä ei ollut työn kannalta varsinaista merkitystä, koska muissa vaiheissa koneisiin tultaisiin aina viittaamaan pelkällä IP-osoitteella. Isäntänimet määrittämällä oli helpompi pysyä selvillä, mitä palvelinta kulloinkin hallitsee.

Asennusohjelma kyseli lisäksi aikavyöhykettä, pääkäyttäjän (root) salasanaa sekä käyttöjärjestelmän mukana asennettavia lisäosia. Lisäosien listasta oli oletuksena valittu ”Desktop – Gnome”, jolloin käyttöjärjestelmän mukana olisi asennettu Gnome:n graafinen käyttöliittymä. Tuo valinta otettiin pois päältä ja listasta valittiin pelkästään kohta *Server*. Tällöin koneisiin ei asennettu graafista käyttöliittymää ja asennettiin vain palvelimille tyypilliset ohjelmistot. Graafinen käyttöliittymä jätettiin asentamatta, jotta palvelimista saataisi paras suorituskyky irti.

Asennuksen aikana asennusohjelma latasi kaikki tarvittavat ohjelmistot ja muut osat Internetistä ja asensi ne sitä mukaa kun ne saapuivat. Asennusten päätyttyä ohjelma ilmoitti, että käyttöjärjestelmän asennus on valmis ja että asennusmedia (CD-levy) voidaan poistaa asemasta ja kone käynnistää uudelleen. Koneen uudelleenkäynnistyksen jälkeen koneelle päästiin kirjautumaan antamalla käyttäjänimeksi *root* ja kirjoittamalla asennuksen yhteydessä annettu salasana. Myöhemmissä vaiheissa kaikkia palvelimia hallittiin PuTTy-nimisellä SSH-terminaaliohjelmalla. Kulloinkin hallittu kone oli helppo tunnistaa pelkästään komentokehotetta katsomalla (kuva 4).

KUVA 4. Asennus valmis

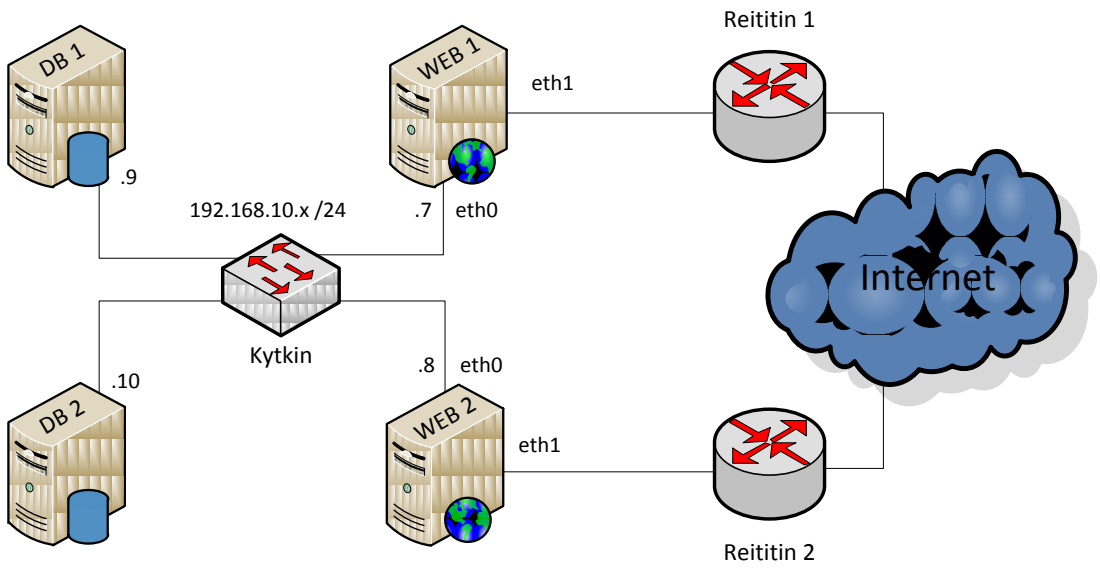
3.2 Asetukset ja ohjelmistot

3.2.1 Päivitykset

Käyttöjärjestelmien asennuksen jälkeen kullekin koneelle ajettiin uusimmat saatavilla olevat päivitykset. Päivitykset ajettiin komennolla *yum update*. Kone latasi komennolla tiedot saatavilla olevista päivityksistä. Ennen päivitysten asennusta kone kysyi, halutaanko varmasti asentaa päivitykset. Ensimmäisellä päivityskerralla kone kysyi myös luotetaanko päivitysten lähteeseen sieltä tulleen sertifikaatin perusteella. Molemmat ilmoitukset hyväksyttiin ja kone latasi ja asensi päivitykset.

3.2.2 Verkon rakenne

Kaikkien koneiden päivityksen jälkeen verkon rakennetta muutettiin. Arkkitehtuurin tietoturvan kannalta oli tärkeää, ettei kumpikaan tietokantapalvelin näy ulkoverkossa millään lailla. Toisin sanoen kummallakaan tietokantapalvelimella ei kuulunut olla Internet-yhteyttä. Kummankin WEB-palvelin tuli puolestaan kytkeytyä verkkoon oman reitittimensä kautta. Näiden tietojen perusteella verkon rakenne muutettiin alla olevan kuvion mukaiseksi (kuvio 2).



KUVIO 2. Lopullinen verkon rakenne

Tämän rakennemuutoksen jälkeen tietokanta- ja WEB-palvelimet keskustelivat keskenään sisäverkossa 192.168.10.x, jossa x on kunkin palvelimen laiteosoitteen numero. Koneiden laiteosoitteet näkyvät kuviossa 2 kunkin koneen kytkimeen liittyvän johdon

päässä. Esim. WEB 1 -koneen sisäverkon osoite on 192.168.10.7, jolloin WEB 1 -koneen laiteosoite on 7.

Kuten aiemmin on käynyt ilmi, kummallakin WEB-palvelimella on kaksi verkkokorttia. Toinen verkkokorteista kytkeytyy samaan sisäverkkoon tietokantapalvelinten kanssa ja toisen kautta palvelin kytkeytyy reitittimen välityksellä ulkoverkkoon.

3.2.3 IP-osoitteiden määritys

Koska palvelinten välisessä sisäverkossa ei ollut reititintä, koneiden sisäverkon osoitteet oli määritettävä käsin. CentOS-Linuxissa kunkin verkkokortin asetukset löytyvät tekstitiedostosta *ifcfg-ethX*, jossa *X* on ko. verkkoliittimen numero. CentOS aloittaa verkkoliittimien numeroinnin nollasta. Jos koneessa on vain yksi verkkokortti, sen asetukset löytyvät tiedostosta *ifcfg-eth0*. WEB-palvelinten tapauksessa näitä asetustiedostoja oli kaksi, kullekin verkkokortille oma. Käytetyssä verkon rakenteessa jokaisen koneen *eth0*-verkkoliitin kuului koneiden väliseen sisäverkkoon.

Liitinten asetustiedostot löytyvät koneiden hakemistosta */etc/sysconfig/network-scripts/*. Kunkin koneen *eth0*-liittimen asetuksia muokattiin *nano*-tekstieditorilla. Tiedostoa päästiin muokkaamaan komennolla *nano /etc/sysconfig/network-scripts/ifcfg-eth0*.

Koska kaikki neljä konetta saivat IP-osoitteen käyttöjärjestelmän asennuksen yhteydessä konttorin reitittimen DHCP:ltä, jokaisen koneen *eth0*-liittimen asetukset näyttivät ennen muokkausta tältä:

```
# Intel Corporation 82540EM Gigabit Ethernet Controller
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=08:00:27:8C:B5:A2
ONBOOT=yes
```

Näissä asetustiedostoissa risuaitamerkillä alkavat rivit ovat kommenttirivejä, joita käyttöjärjestelmä ei ota huomioon. Tiedoston DEVICE-kohta kertoo, minkä verkkoliittimen asetuksista on kysymys. HWADDR-kohta kertoo verkkoliittimen fyysisen MAC-osoitteen. Tähän kohtaan ei koskettu, koska liitinten MAC-osoite ei muutu koskaan. ONBOOT-kohta määrittää, käynnistyykö verkkoliitin koneen käynnistyksen yhteydessä. Tämä kohta jätettiin oletusarvoonsa, jolloin liitin käynnistyy koneen käynnistyessä.

BOOTPROTO-kohta kertoo, millä tavalla liittimen IP-asetukset määritetään verkkoliittimen käynnistyessä. Jokaisen koneen asetuksissa tähän kohtaan muutettiin arvon *none*,

jolloin IP asetusten määritykseen ei käytetä mitään protokollaa. Tällöin kone hakee IP-asetukset tästä samasta tiedostosta suoraan. Kuten yllä näkyy, IP-asetukset eivät ole automaattisesti tässä tiedostossa näkyvissä. Ne jouduttiin lisäämään tiedostoihin käsin. Asetusten lisäämisen jälkeen tiedoston sisältö näytti tältä:

```
# Intel Corporation 82540EM Gigabit Ethernet Controller
DEVICE=eth0
BOOTPROTO=none
IPADDR=192.168.10.7
NETMASK=255.255.255.0
HWADDR=08:00:27:8C:B5:A2
ONBOOT=yes
```

IPADDR-kohdassa määritetään koneen IP-osoite. Kunkin koneen asetuksiin määritettiin konetta vastaava IP-osoite. Yllä olevassa esimerkissä näkyy WEB 1 -koneen IP-osoite. NETMASK-kohdassa määritetään verkon maski. Sisäverkossa käytettiin normaalia C-luokan verkkomaskia. Asetusten muokkauksen jälkeen kunkin koneen verkkokortit käynnistettiin uudestaan, jotta määritetyt muutokset tulisivat voimaan. Tämä tapahtui komennolla *service network restart*.

3.2.4 Palomuurin asetukset

Verkkosovelluksesta ja ulkoverkkoon kytkeytyvistä palvelinkoneista puhuttaessa palomuurilla ja tietoturvalla on suuri merkitys, varsinkin tuotantokäytössä. Tässä työssä käytetyn testiympäristön tapauksessa tilanne ei ollut niin vakava, koska palvelimet olivat käytössä vain silloin, kun olin tekemässä työtä. Siitä huolimatta palomuurin asetuksille on korrektia omistaa oma luku.

Palvelinkoneissa käytettiin käyttöjärjestelmän asennuksen mukana tullutta *Iptables*-palomuurisovellusta. Palomuurin asetukset olisi kuulunut antaa komentoriville yksi kerrallaan. Vaivannäön minimoimiseksi asetusten antamiseen käytettiin BASH-skriptiä, joka ajoi kaikki tarvitut asetusrivit, kun se suoritettiin. Kunkin koneen palomuurin asetusten määrittämiseen käytettiin omaa skriptiä. Skriptit erosivat toisistaan vain niiltä osin, kun viitataan kyseisen koneen IP-osoitteeseen.

Huomioon otettavia verkkoyhteyksiä palomuurien asetuksia mietittäessä olivat

- SSH-yhteys kannettavalta kullekin palvelimelle
- HTTP-yhteys ulkoverkosta WEB-palvelimille
- MySQL-klusterin liikenne sisäverkossa
- Ping sisäverkossa

- muu liikenne sisäverkossa
- oletuskäytännöt sisään ja ulos liikkuvalle sekä välitettävälle liikenteelle.

Palomuurin asetuksiin määritettiin sallittavaksi kannettavan ja palvelinten välinen SSH-yhteys, WEB-palvelimille ulkoverkosta tuleva HTTP-yhteys, sisäverkossa liikkuvan MySQL-klusterin liikenne, sisäverkossa liikkuva ping, muu sisäverkossa kulkeva liikenne ja palvelimilta poispäin lähtevä liikenne. Lisäksi sallittiin liikenne avattua yhteyttä pitkin ja liikenne, joka liittyy johonkin avattuun yhteyteen. Muunlainen sisään tuleva tai välitettävä liikenne määritettiin estettäväksi.

SSH-yhteys määritettiin palvelinten ja kannettavan välille sallituksi, jotta palvelimia voisi hallita PuTTy:llä. WEB-palvelimiin ulkoverkosta tuleva HTTP-yhteys määritettiin sallituksi, jotta niillä pyörivää verkkosovellusta voitaisiin käyttää ulkoverkosta. Sisäverkossa liikkuvan MySQL-klusterin liikenne määritettiin sallituksi, jotta WEB- ja tietokantapalvelimet voisivat keskustella keskenään verkkosovelluksen toiminnan aikana.

Ping-liikenne sallittiin kaikkialla sisäverkossa yhteyksien testaustarkoituksiin. Tuotantoympäristössä ping-säännöt on järkevää sallia vain tietyistä IP-osoitteista tietoturvan maksimoimiseksi. Muun sisäverkossa kulkevan liikenne määritettiin sallituksi siltä varalta, että palvelinten välillä kulkee muunlaista tarpeellista liikennettä. Tätä sääntöä tuskin kannattaa käyttää tuotantokäytössä, mutta testiympäristössä säännöstä ei uskottu koituvan harmia.

BASH-skriptit luotiin asetusten ajoa varten *nano*-tekstieditorilla. Skriptit sijoitettiin kunkin koneen hakemistoon */var/tmp*. Skriptit luotiin *myfirewall*-nimiseen tiedostoon komennolla *nano /var/tmp/myfirewall*. Komentojen kirjoituksen ja tiedoston tallennuksen jälkeen tiedosto muutettiin suoritettavaksi komennolla *chmod +x /var/tmp/myfirewall*. Tämän jälkeen skripti suoritettiin siirtymällä sen kanssa samaan hakemistoon komennolla *cd /var/tmp* ja antamalla komento *./myfirewall*.

Skripti otti käyttöön määritetyt palomuurin asetukset ja tulosti lopuksi näytölle käyttöönotetut säännöt. Esimerkki työssä käytetystä skriptistä löytyy liitteestä 1. (Galuschka 2012.)

3.2.5 Ohjelmistot

Jotta verkkosovellus saatiin toimimaan WEB-palvelimilla, niille tuli asentaa joitain sen vaatimia ohjelmistoja. Vaadittuja ohjelmistoja olivat Apache, PHP-tulkki ja PEAR. Näistä Apache ja PHP-tulkki tulivat valmiina käyttöjärjestelmän asennuksen mukana, mutta

PHP-tulkin versio oli sovellusta varten liian uusi. Siitä syystä palvelimille asennettiin PHP-tulkin vanhempi version sekä PEAR. Apachen asetuksia jouduttiin myös muuttamaan.

Arkkitehtuurimuutoksen kohteena oli verkkosovellus. Tällä tarkoitetaan sitä, että ohjelmaa käytetään nettiselaimella Internetin yli. Sovellus näyttäytyy sen käyttäjälle tavallisen verkkosivun tapaan, ja se sijaitsee WEB-palvelimella samalla lailla kuin tavallinen verkkosivu. Verkkosovellusta ei siis käytännössä tarvinnut asentaa WEB-palvelimille, ainoastaan siirtää kansiossaan WEB-palvelimen hakemistoon. Verkkosovellus sijoitettiin WEB-palvelimien hakemistoon */var/www/html/sovellus*.

Seuraavaksi käydään läpi Apachen asetusten muutokset ja PHP-tulkin ja PEAR:n asennus ja asetusten muutokset.

3.2.5.1 Apache

Apache on vapaan lähdekoodin HTTP-palvelin. Se välittää nettiselaimelle palvelimella sijaitsevan sivuston sisältöä HTTP-protokollan avulla, kun selaimella siirrytään palvelimille määrättyyn verkko-osoitteeseen. Apache asentui käyttöjärjestelmän mukana, joten sitä ei tarvinnut erikseen asentaa. Sen asetuksiin jouduin kuitenkin tekemään joitain muutoksia. (The Apache Software Foundation 2012.)

Apachen asetukset löytyvät tiedostosta */etc/httpd/conf/httpd.conf*. Muutokset, joita Apachen asetuksiin täytyi tehdä, koskivat nettisivujen juurihakemistoa ja sen asetusrivejä. Oletusasetuksena Apache määrittää nettisivujen juurihakemistoksi kansion */var/www/html*. Koska *html*-kansiota ei haluttu täyttää sovelluksen tiedostoilla, sovellus siirrettiin omassa kansiossaan *html*-kansion alle ja nettisivujen juurihakemisto muutettiin sovelluksen kansioksi. Apachen asetuksissa tämä määritetään *DocumentRoot*-rivillä. *DocumentRoot*-asetus muutettiin muotoon

DocumentRoot "/var/www/html/sovellus"

Muutoksen jälkeen sovellus vastasi suoraan osoitteesta *ernontesti.fns.fi*, niin kuin oli tarkoitus. Ilman muutosta sovellus olisi vastannut osoitteesta *ernontesti.fns.fi/sovellus*.
Apachen asetuksissa nettisivujen juurihakemistolle oli valmiiksi luotu kansion asetukset, jotka osoittivat myös oletuksena olleeseen *html*-kansioon. Jokaiselle palvelimella sijaitsevalle verkkosivujen kansiolle voidaan antaa *httpd.conf*-tiedostoon omat asetukset.

Kansioiden asetukset määritetään *Directory*-tagien väliin hieman HTML-syntaksin tapaan. Juurihakemistolle määrätyt kansion asetukset muutettiin vastaamaan edellä tehtyä hakemiston muutosta määrittämällä kansiolle sama polku

<Directory "/var/www/html/sovellus">

Kansion asetusten alla *Options*-rivillä määritetään kansion HTML-asetukset. Oletuksena rivillä olivat arvot *Indexes* ja *FollowSymlinks*. Näiden asetusten lisäksi riville lisättiin PHP-sovelluksen tarvitsemat arvot *Includes* ja *ExecCGI*. Seuraavalla *AllowOverride*-rivillä määritetään, mitkä asetukset voidaan sisällyttää Apachen tekemään *htaccess*-tiedostoon. Oletuksena rivillä on arvo *None*, jolloin asetuksia ei viedä *htaccess*-tiedostoon. Tähän kohtaan muutettiin arvo *All*, jolloin kaikki asetukset menevät em. tiedostoon. Muut asetukset jätettiin oletusarvoonsa. Muutoksen jälkeen kansion asetukset näyttivät kokonaisuudessaan tälle

```
<Directory "/var/www/html/sovellus/">
        Options Indexes Includes FollowSymLinks ExecCGI
        AllowOverride All
        Order allow,deny
        Allow from all
</Directory>
```

3.2.5.2 PHP-tulkki

Verkkosovelluksen käyttäjälle esittämä HTML-sisältö luodaan PHP-koodin avulla. Hakemiston tiedostot ovat tavallisen verkkosivun HTML-tiedostoista poiketen PHP-tiedostoja, joista Apache tavallisena HTTP-palvelimena ei ymmärrä mitään. Tästä syystä palvelimille täytyi asentaa erillinen PHP-tulkki, joka suorittaa PHP-koodin Apachen puolesta.

Kun Apache huomaa, että käyttäjän selain pyytää PHP-tiedostoa, se pyytää PHP-tulkkia suorittamaan PHP-koodin sen puolesta. Tulkki palauttaa Apachelle koodin suorituksen tulokset sen ymmärtämässä HTML-muodossa ja Apache puolestaan palauttaa saamansa tulokset sovelluksen käyttäjälle. (Achour, Betz & muut 2013a.)

Käyttöjärjestelmien asennuksen yhteydessä palvelimille asentui PHP-tulkin versio 5.3. Verkkosovellus on kuitenkin suunniteltu käyttämään PHP:n versiota 5.2, joten PHP-

tulkista täytyi asentaa tuo versio. Vanhaa PHP:n versiota ei löydy CentOS:n virallisesta ohjelmien säilytyspaikasta, josta sen olisi saanut asennettua antamalla komennon *yum install php*. Tästä syystä täytyi ottaa käyttöön kolmannen osapuolen säilytyspaikka, josta vanhempi PHP:n versio löytyisi.

Yksi monista tällaisista säilytyspaikoista on mm. Atomicorp-nimisen yrityksen palvelimilta. Otin Atomic:n säilytyspaikan käyttöön antamalla kummallekin WEB-palvelimelle komennon *wget -q -O - http://www.atomicorp.com/installers/atomic | sh*. Säilytyspaikan vaihtamisen jälkeen pakettienhallintaohjelma *yum*:n asetuksiin täytyi käydä lisäämässä rivi, jolla estetään PHP:n 5.3-version pakettien näkyminen. Asetukset löytyivät tiedostosta */etc/yum.conf*. Tiedoston loppuun lisättiin *nano*-tekstieditorilla rivin *exclude=php-*5.3**, jolloin kaikki PHP:n versio 5.3-paketit jätettiin hakutuloksista pois.

Muutoksen jälkeen vanhempi PHP:n versio pystyttiin asentamaan. PHP-tulkin lisäksi tulkkiin tuli asentaa verkkosovelluksen tarvitsemia lisäosia. Lisäosien ja PHP-tulkin asennus hoitui yhdellä komennolla. Asennettavat paketit ovat ao. komennossa eritelty välilyönnillä. Tulkin ja sen lisäosien asennus tapahtui komennolla

> *yum install php php-xml php-pear php-devel php-soap php-xmlrpc php-common php-pdo php-ldap php-cli php-gd php-mysql php-mbstring*

Asennusten jälkeen verkkosovelluksen tuotantopalvelimilta kopioitiin PHP-tulkin asetustiedosto (*php.ini*) ja se laitettiin testiympäristön WEB-palvelimille */etc*-kansion alle. Tämä helpotti tulkin asetusten määritystä paljon, koska muussa tapauksessa testiympäristön sekä tuotantopalvelimen asetustiedostot olisi jouduttu käymään rivi riviltä läpi.

3.2.5.3 PEAR

Lyhenne PEAR tulee englanninkielen sanoista **P**HP **E**xtension and **A**pplication **R**epository. PEAR on PHP:tä varten tehty kehys, jonka kautta PHP:n toiminnallisuutta on mahdollista jatkaa erilaisilla liitännäisillä. Verkkosovelluksessa mm. kaikki tietokannan käsittelyt on tehty käyttämällä PEAR:n DB-liitännäistä. Tästä syystä PEAR oli pakko asentaa WEB-palvelimille. (The PHP Group 2013.)

PEAR asennettiin WEB-palvelimille edellisessä kappaleessa esitetyllä komennolla, jolla asennettiin PHP-tulkin osia. Komennon mukana ei kuitenkaan asentunut verkkosovelluksen tarvitsemia PEAR-liitännäisiä. Verkkosovelluksen tarvitsemia PEAR-liitännäisiä oli PHP-tulkin liitännäisten tapaan melkoinen lista, joten ne on listattu vain ao. komen-

toon. Liitännäiset on eritelty komentoon samalla tapaa välilyönnillä, kuin PHP:n liitännäisten tapauksessa. Tarvittavat PEAR:n liitännäiset asennettiin komennolla

pear install Archive_Tar Auth Cache Cahce_Lite Console_Getopt DB DB_Pager HTML_Common HTML_Table PEAR Structures_Graph XML_RPC XML_Util

Näiden liitännäisten lisäksi asennettiin kaksi muuta liitännäistä, joista oli saatavana vain kokeiluversiot. Nämä liitännäiset oli asennettava erikseen komennoilla

pear install channel://pear.php.net/Spreadsheet_Excel_Writer-0.9.3

ja

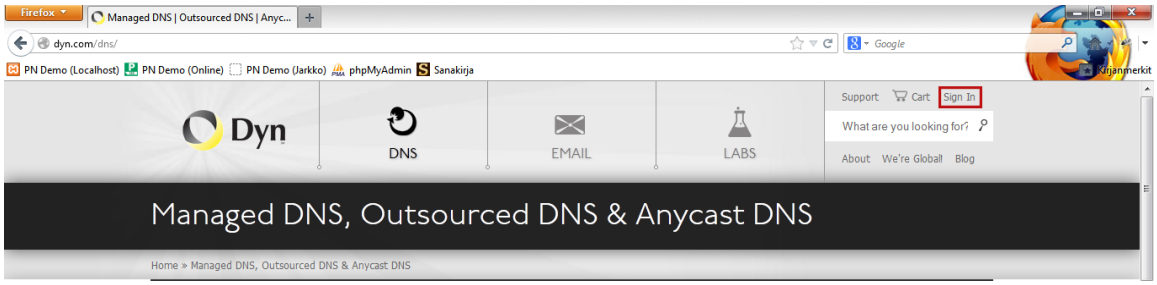
pear install channel://pear.php.net/OLE-1.0.0RC2

Näiden asennusten jälkeen PEAR oli kaikkine liitännäisineen käyttövalmis.

3.3 Verkko-osoitteen varaaminen

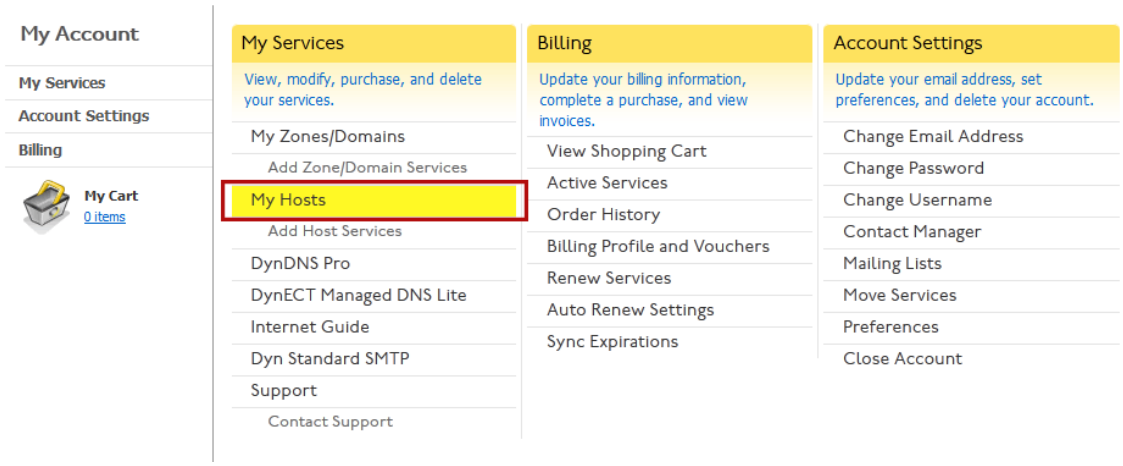
Jotta testiympäristön WEB-palvelimiin pääsisi selaimella helposti käsiksi, niitä varten täytyi varata omat dynaamiset verkko-osoitteet. Verkko-osoitteen varaamisella tarkoitetaan sitä, että palvelimen ulkoisen IP-osoitteeseen ”pariksi” varataan käyttäjäystävällisempi verkko-osoite, esim. *abc.testi.fi*. Tämä osoite on huomattavasti helpompi muistaa, kuin palvelimen IP-osoite, joka voisi olla esim. 37.88.211.10.

Koska molemmat WEB-palvelimet saivat oman reitittimensä kautta verkko-operaattorilta oman ulkoisen IP-osoitteen, ei ollut takuu varmaa, että IP pysyisi jatkuvasti samana. Dynaamisella verkko-osoitteella tarkoitetaan sitä, että esim. osoitteeseen *abc.testi.fi* päästään aina käsiksi, vaikka palvelimen IP-osoite muuttuisikin. IP- ja verkko-osoiteparin ylläpidosta vastaa palvelu, josta verkko-osoite on varattu. Testiympäristön WEB-palvelimia varten dynaamiset verkko-osoitteet varattiin DynDNS-palvelusta. Osoitteiden varaaminen aloitettiin menemällä palvelun verkkosivulle osoitteeseen *http://dyn.com/dns* ja kirjautumalla sisään (kuva 5).



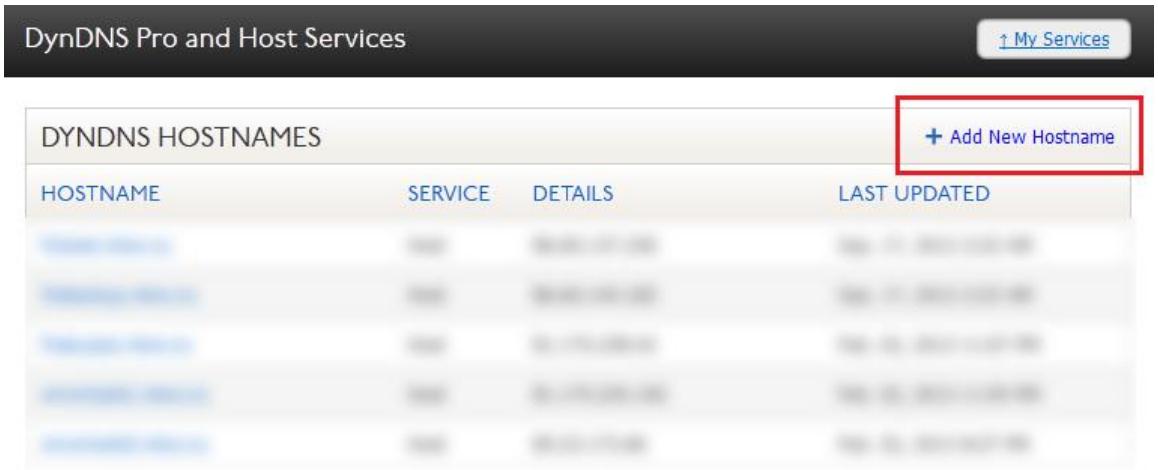
KUVA 5. DynDNS kirjautuminen

Palveluun kirjauduttiin käyttämällä yrityksen tunnuksia. Seuraavalta sivulta valittiin kohta *My Hosts*, jonka kautta päästiin hallitsemaan dynaamisia osoitteita (kuva 6).



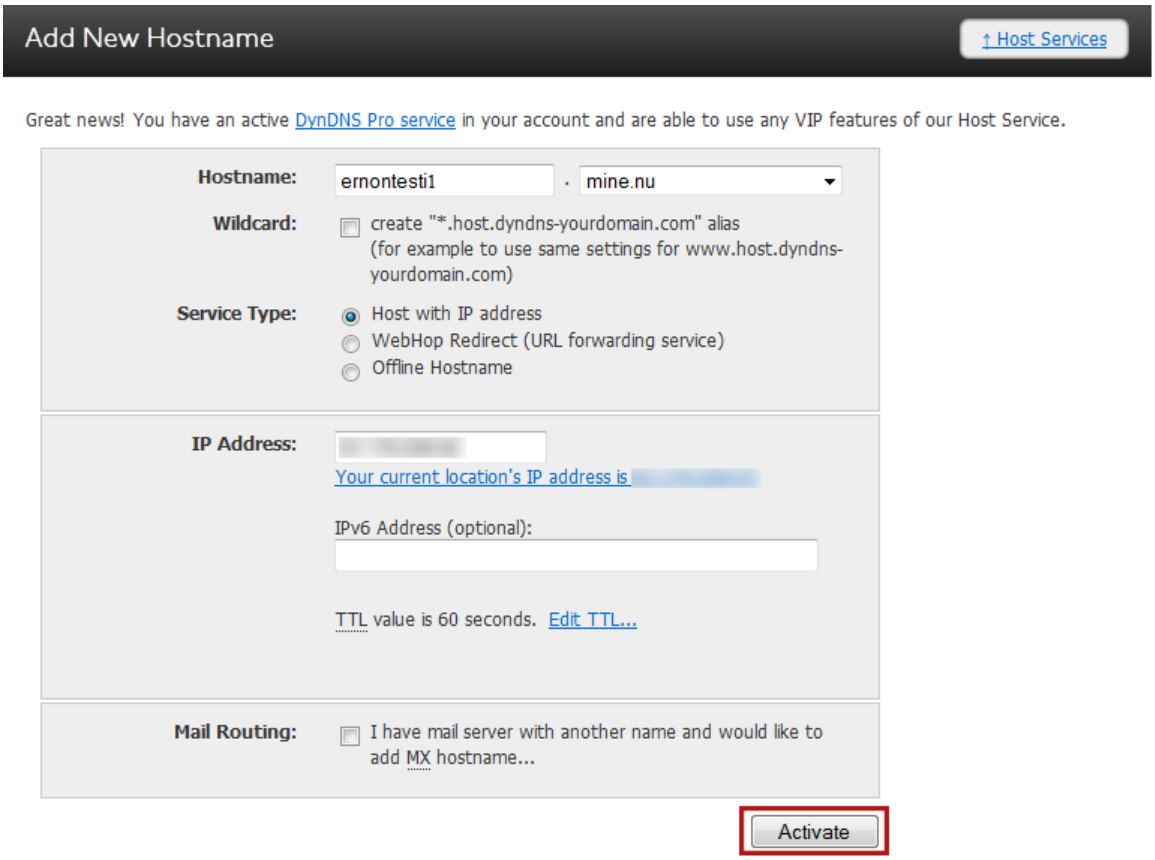
KUVA 6. Isäntäosoitteiden hallinta

Seuraavalla sivulla näkyivät kaikki tämänhetkiset dynaamiset osoitteet. Uusi verkko-osoite varattiin painamalla *Add New Hostname* (kuva 7).



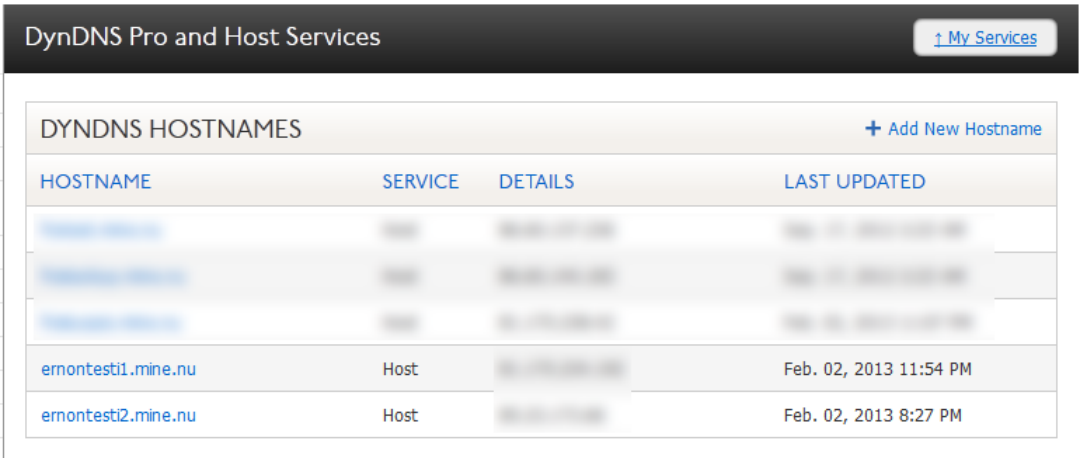
KUVA 7. Isäntäosoitteen lisäys

Seuraavalla sivulla WEB 1 -palvelimelle määritettiin dynaaminen osoite. Palvelimelle varattiin osoite *ernontesti1.mine.nu*. Palvelun tyypiksi valittiin IP-osoitteen omaava isäntä ja IP-osoitteeksi määritettiin nykyinen ulkoverkon osoite. WEB 2 -palvelinta varten tarkistettiin sen reitittimen nykyinen osoite ja WEB 2 -osoitteen varaamiseen käytettiin sen reitittimen ulkoverkon osoitetta. Osoite varattiin painamalla *Activate*. WEB 2 -palvelimelle lisättiin dynaaminen osoite *ernontesti2.mine.nu* (kuva 8).



KUVA 8. Isäntäosoitteen määritys

Osoitteiden lisäyksen jälkeen ne näkyivät dynaamisten osoitteiden listassa (kuva 9).



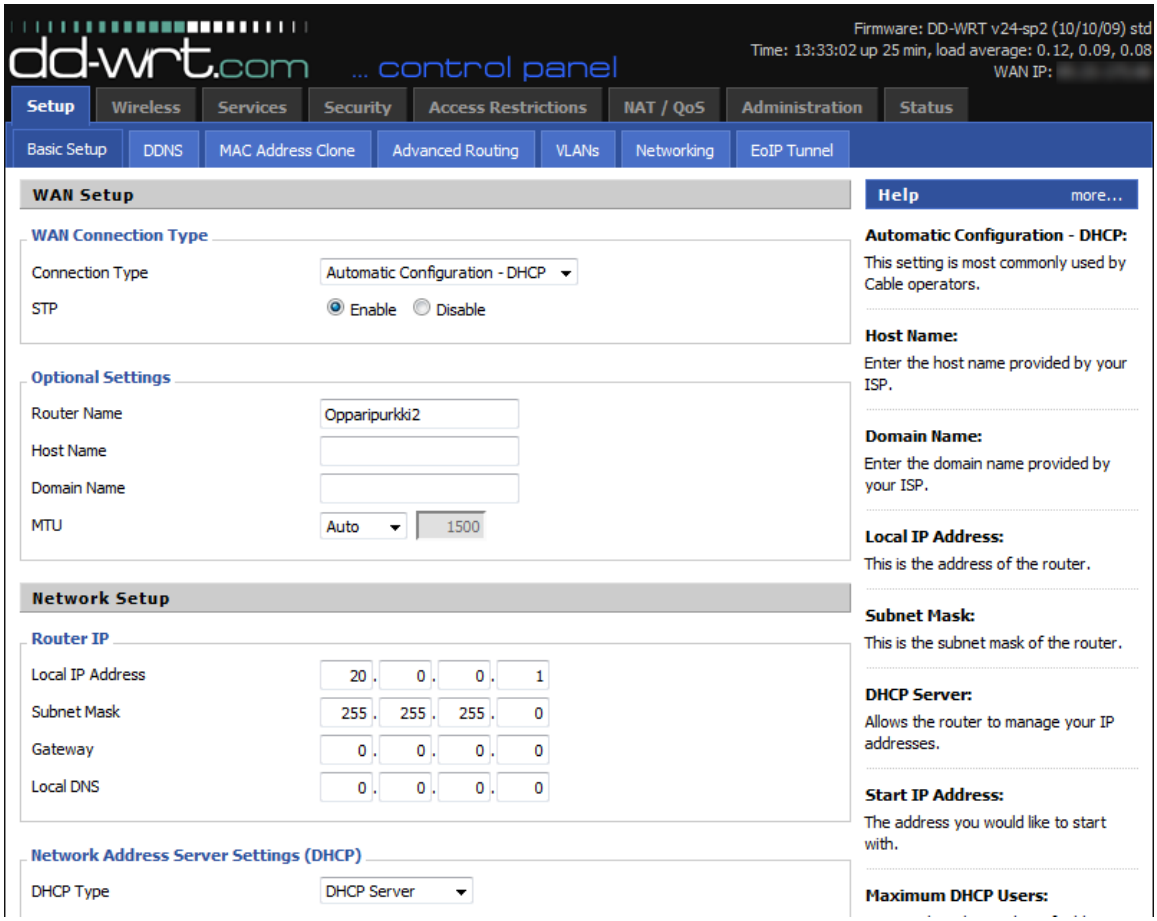
KUVA 9. Lisätyt isäntäosoitteet

Nyt kun WEB-palvelimille oli varattu dynaamiset osoitteet, testiympäristön pystytyksessä oli jäljellä enää reitittimien asetusten muokkaaminen.

3.4 Reitittimien asetukset

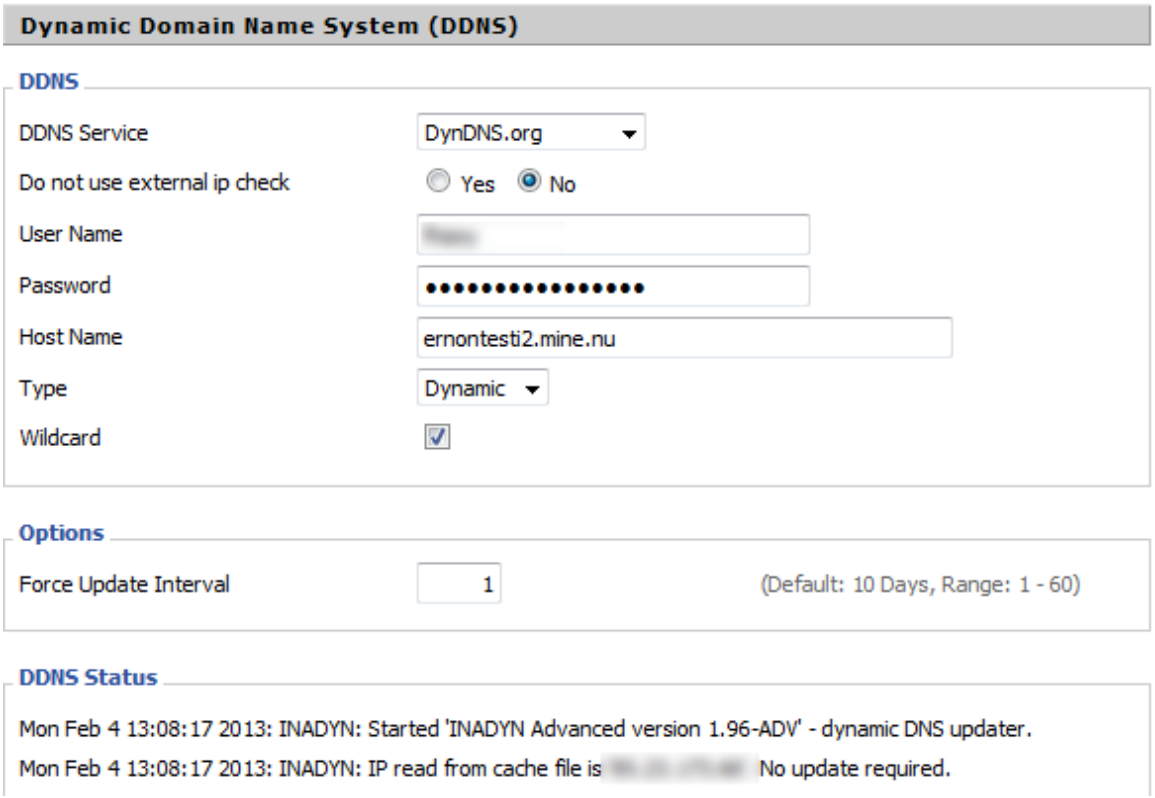
WEB-palvelinten reitittiminä oli käytössä kaksi Linksys WRT45GL-reititintä. Kumpikin WEB-palvelin kytkeytyi omaan reitittimeensä *eth1*-portin kautta ja kumpikin reititin kytkeytyi toimiston kaapelimodeemiin. Kaapelinmodeemi laitettiin siltaamaan operaattorilta tulevat yhteydet edelleen reitittimille. Näin kumpikin reititin sai operaattorilta oman ulkoverkon IP-osoitteen sen sijaan, että modeemi olisi antanut kummallekin reitittimelle sen määräämän sisäverkon osoitteen.

Reititinten hallinta tapahtui nettiselaimen kautta kirjoittamalla osoiteriville kulloinkin hallittavan reitittimen IP-osoitteen. Alla olevassa kuvassa näkyvät reitittimen perusasetukset (kuva 10).



KUVA 10. Reitittimen perusasetukset

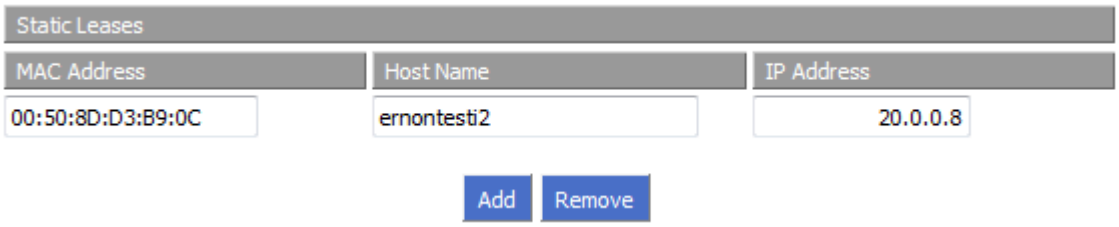
Kumpikin reititin laitettiin vastaanottamaan ulkoverkon (WAN) osoite automaattisesti kaapelimodeemilta. Palvelinten ja reitittimien välillä käytettiin sisäverkkoja 10.0.0.0 (WEB 1) ja 20.0.0.0 (WEB 2). Jotta kumpikin palvelin saatiin käyttämään edellä varattu- ja dynaamisia osoitteita, DynDNS-asetukset tuli määrittää kullekin reitittimelle ja palvelinten ja reititinten välille tuli määrittää 1:1 verkko-osoitteen käännös (ns. One-to-One NAT). DynDNS-asetukset määritettiin hallintasivun *Setup/DDNS* välilehdeltä (kuva 11).



KUVA 11. DDNS asetukset

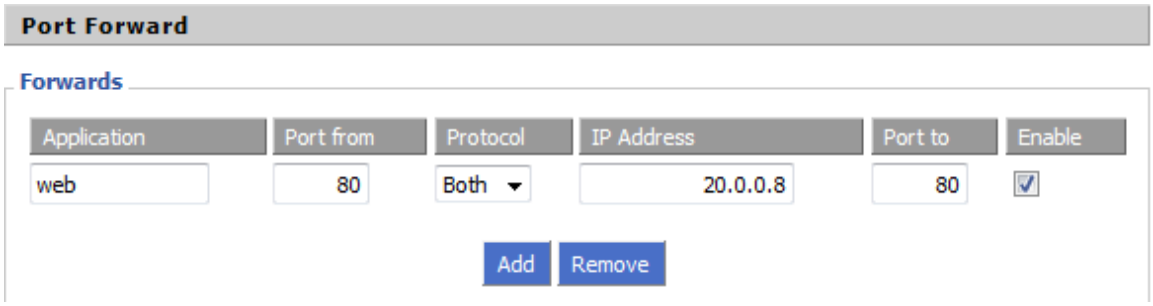
DynDNS-asetuksiin tuli määrittää käytettävä palvelu (DynDNS), palvelun käyttäjänimi ja salasana ja dynaamisen verkko-osoitteen isäntänimi sekä osoitteen tyyppi. Näiden asetusten jälkeen osoite *ernontesti2.mine.nu* osoitti pelkästään käytettyyn reitittimeen. Jotta se saatiin osoittamaan reitittimen sijaan WEB-palvelimeen, reitittimen ja palvelimen välille tuli määrittää verkko-osoitteen käännös.

Jotta käännös saatiin tehtyä oikein, kumpikin reititin laitettiin jakamaan 2kpl sisäverkon osoitteita DHCP:llä (toinen osoite palvelinta ja toinen kannettavaa varten). Tämän lisäksi kummankin reitittimen asetuksiin määritettiin staattinen varaus reitittimeen kytkeytyvää palvelinta varten, jolloin palvelimen sisäverkon osoite ei koskaan muutu (kuva 12).



KUVA 12. Staattinen IP-osoitteen varaus

Verkko-osoitteen käännöstä varten piti lisäksi määrittää HTTP-porttiin 80 tulevan liikenteen välitys reitittimeltä palvelimille. Määritys tehtiin *NAT/QOS / Port Forward*-välilehdeltä. Portin välitystä varten tuli määrittää porttia käyttävän ohjelman nimi (web), sisääntuloportti reitittimelle, välitettävät protokollat, palvelimen IP-osoite ja sisääntuloportti palvelimelle (kuva 13).



KUVA 13. NAT

Asetukset määrityksen jälkeen muutokset otettiin voimaan painamalla *Administration / Management* välilehdeltä *Apply Settings* -painiketta. Asetusten käyttöönoton jälkeen reitittimet käynnistettiin uudestaan samalta välilehdeltä painamalla *Reboot Router* -painiketta.

3.5 DNS Round Robin

Jotta kahdelle WEB-palvelimelle asennettua verkkosovellusta pystyttiin käyttämään yhdestä verkko-osoitteesta, yrityksen DNS-palvelimilta varattiin sovellusta varten osoite ernontesti.fns.fi. Molempien WEB-palvelinten dynaamiset osoitteet määritettiin DNS-palvelimilta osoittamaan tuohon samaan osoitteeseen. Koska molemmat palvelimet eivät pysty vastaamaan osoitteeseen tuleviin pyyntöihin yhtä aikaa, DNS-palvelimille tehtiin määritys, jonka avulla ernontesti.fns.fi-osoitteeseen tulevat pyynnöt ohjataan tasaisesti kahden WEB-palvelimen kesken. Tähän käytettiin DNS Round Robin -tekniikkaa.

DNS Round Robin on tekniikka, jonka avulla tiettyyn verkkoon tai verkkolaitteisiin kohdistuvaa liikennettä voidaan tasata DNS-palvelimen toimesta. Round Robin muodostaa siihen lisättyjen palvelimien luettelosta jonon, ja välittää määritettyyn verkko-osoitteeseen tulevan pyynnön jonon ensimmäiselle palvelimelle. Tämän jälkeen jonon ensimmäinen laite siirtyy jonon perälle ja jonossa seuraava laite siirtyy ensimmäiseksi. Näin jokainen pyyntö ohjautuu vuorollaan eri palvelimille. (Wikipedia 2013.)

Round Robinin toiminnassa huomattiin kuitenkin työn alkuvaiheessa puute. Kun Round Robin oli arponut listalta vastaavan palvelimen, kaikki tämän kyseisen asiakkaan pyynnöt ohjautuivat jatkossa ensin vastanneelle palvelimelle. Jos palvelin sattui kaatumaan, pyyntöjä ei ohjattu toiselle, vielä pystyssä olevalle palvelimelle. Tästä syystä DNS Round Robin laitettiin toimimaan siten, että verkkosovelluksen asiakaskoneiden pyyntöihin vastaavaa WEB-palvelinta vaihdetaan 30 sekunnin välein.

Järjestely poisti yllä mainitun ongelman, mutta lopputulos ei silti ollut paras mahdollinen arkkitehtuurin toimintavarmuuden kannalta. Se riitti hyvin simuloimaan oikeantapaista tilannetta, mutta tuotantoympäristössä tätä tekniikkaa ei kannata käyttää. DNS Round Robinista on kerrottu lisää arkkitehtuurin testauksen yhteydessä (luku 6). Tuotantokäytössä DNS Round Robinin tilalle sopivampi kuorman tasaus tekniikka olisi Ultra Monkey. (Horman 2005.)

Kaikkien tässä luvussa esitettyjen asetusten ja asennusten jälkeen testiympäristö oli pystytetty. Molemmat palvelimet vastasivat osoitteesta *ernontesti.fns.fi*, jolloin päästiin verkkosovelluksen etusivulle.

4 MYSQL-KLUSTERI

Seuraava tehtävä testiympäristön pystyttämisen jälkeen oli MySQL-klusterin pystyttäminen testiympäristön palvelimille. Työvaiheita olivat klusterin käyttöönoton suunnittelu, solmujen asennukset, solmujen asetusten määrittäminen sekä verkkosovelluksen tietokannan muokkaus klusterille sopivaksi. MySQL-klusteri on erittäin laaja, joten tässä työssä käsitellään klusteria vain siltä osin, kuin se liittyi työhön käytännössä.

4.1 MySQL-klusterin esittely

Suurin ongelma, joka verkkosovelluksen arkkitehtuurimuutoksella pyrittiin korjaamaan, liittyi järjestelmän vikasietoisuuteen. Kuten mainitsin kappaleessa 2.2, sovelluksen arkkitehtuurissa on monta ns. heikkoa lenkkiä. Jos sovelluksen palvelimien säilytystilassa tapahtuu sähkökatko tai verkkoyhteys katkeaa, sovellusta ei pystytä käyttämään. Ongelma on vakava, koska sovelluksella on useita 24h käyttäjiä.

Koska sovelluksen käyttämä tietokanta on toteutettu MySQL:llä, sovelluksen vikasietoisuutta pyrittiin parantamaan ottamalla käyttöön MySQL-klusteri. MySQL-klusteri on helposti skaalautuva tietokannan hajautusjärjestelmä, jolla pystytään saavuttamaan jopa 99,999 % tietokannan saavutettavuus. Sitä käyttämällä uuteen verkkosovelluksen arkkitehtuuriin ei jäänyt yksittäistä heikkoa lenkkiä. (Davies & Fisk 2006, 1.)

4.1.1 MySQL-klusterin edut

Perinteisessä MySQL-tietokannassa tieto tallennetaan riveihin, jotka järjestetään sarakkeittain taulukoksi. Taulukot tallennetaan tiedostoon, joka sijoitetaan tietokantapalvelimelle ja siihen tehdään kyselyitä. Jos palvelin kaatuu, myös tietokanta kaatuu. Jos palvelimelle tehdään paljon kyselyitä, ainoa ratkaisu kyselyiden nopeuttamiseen on kasvattaa palvelimen suorituskykyä. Klusteria käyttämällä (ts. klusteroinnilla) saavutetaan tilanne, jossa tietokannan kyselyjen käsittely jaetaan usean palvelimen kesken. Tästä syntyy hyvin saavutettava ja hyvän suorituskyvyn omaava tietokanta-arkkitehtuuri, joka ei ole riippuvainen jostain yksittäisestä tietokoneesta. (Davies & Fisk 2006, 1 - 2.)

MySQL-klusteroinnin etuja ovat, että yksittäisen palvelimen kaatuminen ei hajota klusteria eikä uuden palvelimen lisäys klusteriin tuota sen käyttäjälle käyttökatkoa. Nämä edut ovat mahdollisia siksi, koska MySQL-klusteri luo siihen liittyvien palvelimien välille ns. jaetun tyhjyyden. Tällä tarkoitetaan sitä, että kukin klusteriin liittyvä laite omistaa omat

laiteresurssinsa. Palvelimet eivät jaa keskenään esim. kiintolevytilaa, jolloin levyn lukituksista tulisi uusi arkkitehtuurin heikko lenkki, koska vain yksi palvelin saa kerrallaan tehdä muutoksia levylle.

MySQL-klusteri voittaa vikasietoisuudessa perinteisen tietokannan replikoinnin, koska replikoinnissa toissijaisilla palvelimilla on vain lukuoikeudet tietokantaan, joka aiheuttaa jälleen heikon lenkin. Replikointia käytettäessä ensisijaiselta palvelimelta vaaditaan lisäksi korkeaa suorituskykyä. Yhteenvetona MySQL-klusterin etuja ovat siis hyvä suorituskyky ja vikasietoisuus. (Davies & Fisk 2006, 3.)

4.1.2 MySQL-klusterin solmut

MySQL-klusteriin kuuluvia palvelimia nimitetään solmuiksi. Klusteri muodostuu kolmenlaisista solmuista: SQL-solmuista, datasolmuista ja hallintasolmuista. Yhteen klusteriin kuuluu useita SQL- ja datasolmuja sekä yksi tai useampi hallintasolmu. Yksi klusteri voi sisältää enintään 63 solmua. Yhdelle palvelinkoneelle on mahdollista laittaa useita solmuja, mutta suorituskyvyn ja vikasietoisuuden turvaamiseksi on suositeltavaa käyttää yhtä solmua yhdellä palvelimella.

Klusterin *datasolmut* säilyttävät siihen kuuluvan tietokannan taulujen osia ja tekevät tietokantakyselyjen esikäsittelyjä. Klusteroidun tietokannan tieto varastoidaan käytön ajaksi datasolmujen RAM-muistiin. Datasolmut vaativat paljon RAM-muistia varsinkin, jos klusteriin aiotaan laittaa suuri tietokanta. Datasolmuja hallitaan hallintasolmun kautta käyttämällä asetustiedostoja ja hallintaohjelmaa. Yhdessä klusterissa tulisi olla parillinen määrä datasolmuja. Klusterin ominaisuuksista aletaan saada todellista hyötyä, kun klusteri sisältää kaksi tai useampia datasolmuja.

Klusterin *SQL-solmut* toimivat tavallisen MySQL-palvelimen päällä. SQL-solmut toimivat rajapintana sovellusten ja klusterin välillä. Testiympäristössä pyörivä PHP-sovellus on siis yhteydessä klusteriin juuri SQL-solmujen kautta. SQL-solmut ovat yhteydessä niiden takana oleviin datasolmuihin, jotka esikäsittelevät SQL-solmuille tulleita tietokantakyselyitä. SQL-solmut suorittavat kyselyiden lopullisen käsittelyn. Käsittelyn määrä SQL- ja datasolmujen välillä vaihtelee kyselyjen mukaan.

Klusterin *hallintasolmut* toimivat klusterin ”aivoina” ja niillä on siitä syystä monta tärkeää tehtävää. Jos klusterin solmujen välillä on verkko-ongelmia, hallintasolmut päättävät, mikä osa klusterista sammutetaan. Hallintasolmun on oltava toiminnassa ennen kuin klusterin muita osia käynnistetään, koska se pitää hallussaan muiden solmujen asetuk-

sia. Hallintasolmut huolehtivat myös verkon yli tapahtuvista varmuuskopioinneista (jos sellaisia on määritetty). Hallintasolmua käytetään lisäksi klusterin solmujen sammuttamiseen tai uudelleen käynnistämiseen, klusterin tilan seuraamiseen sekä klusterin lokitiedostojen luomiseen ja säilyttämiseen. Yhdessä klusterissa käytetään yleensä vain yhtä hallintasolmua, mutta klusteriin on mahdollista lisätä useampikin hallintasolmu.

Jos hallintasolmu sammuu kesken klusterin toiminnan, siitä ei seuraa välitöntä ongelmaa. Koska hallintasolmu toimii klusterin aivoina, ongelmia alkaa seurata, kun klusterissa tapahtuu hallintasolmun sammumisen jälkeen seuraava muutos. Kun hallintasolmu on sammunut ja jos esim. toinen kahdesta datasolmusta kaatuu, toinen datasolmu jatkaa toimintaansa ikään kuin mitään ei olisi tapahtunut, koska se ei saa hallintasolmulta tietoa toisen solmun kaatumisesta.

Tieto liikkuu klusterin solmujen välillä lähiverkkoa pitkin. Klusteri edellyttääkin käyttöönsä vähintään 100Mbit/s-yhteyttä. Lisäksi kaikkien klusteriin liittyvien palvelimien ja hallintasolmun välillä liikkuu verkkoa pitkin ns. sydämenlyöntiviestejä, jotka ilmaisevat, että ko. klusterin solmu on hengissä. Jos sydämenlyönnit jäävät verkosta johtuen saapumatta tai niiden saapumisessa on liikaa viivettä, hallintasolmu tiputtaa ko. solmun pois klusterista, koska se olettaa solmun vikaantuneen. (Davies & Fisk 2006, 7 - 15.)

4.1.3 MySQL-klusterin tietokantamoottori

Perinteisissä MySQL-tietokannoissa yleisimmin käytettyjä tietokantamoottoreita ovat MyISAM ja InnoDB. MyISAM soveltuu hyvin ympäristöihin, joissa tapahtuu paljon lukukyselyitä ja InnoDB tukee puolestaan muita kehittyneitä ominaisuuksia. MySQL-klusteri ei kuitenkaan käytä kumpaakaan näistä moottoreista. Se käyttää niiden sijaan sitä varten erikseen suunniteltua NDB-tietokantamoottoria. NDB-moottorin ansiosta kyselyjen suoritusta on mahdollista jakaa datasolmujen kesken. Tässä työssä käytettiin NDB-moottorin versiota 7.2.10 (uusin saatavilla oleva versio tätä työtä tehdessä).

Tässä työssä käytetyssä NDB-moottorin versiossa oli kuitenkin joitain rajoituksia, jotka aiheuttivat muutoksia PHP-sovelluksen tietokantaan. NDB-moottorin rajoituksia ovat, että

- se ei tue ollenkaan vierasavaimia
- se ei tue FULL-TEXT-indeksejä
- se sallii vain yhden automaattisesti kasvatettavan (auto-increment) kentän per taulu
- sen jokaisella tietokannan taululla on oltava pääavain.

Moottorilla oli useita muitakin rajoituksia. Yllä on listattu vain tämän työn kannalta olennaiset rajoitteet. (Davies & Fisk 2006, 8 - 10.)

4.2 Klusterin suunnittelu

Klusterin käyttöönottoa varten testiympäristössä oli käytettävissä yhteensä viisi tietokonetta: neljä Linux-palvelinta sekä Windows-kannettava. Klusterin data- ja SQL-solmujen valinta oli testiympäristön tapauksessa helppo, koska arkkitehtuuri oli jo aiemmin suunniteltu siten, että kaksi palvelinta omistetaan tietokannan käsittelyä ja kaksi PHP-sovelluksen pyörittämistä varten. Tästä syystä tietokantapalvelimista tuli klusterin data-solmuja ja WEB-palvelimista SQL-solmuja.

Klusterin hallintasolmun valinta oli hieman mutkikkaampi. Hallintasolmun olisi voinut käytännössä laittaa mille tahansa neljästä Linux-palvelimesta, mutta silloin klusteriin olisi syntynyt uusi heikko lenkki. Jos se palvelin olisi kaatunut, jolla hallintasolmu ja jokin toinen solmu olisivat olleet, silloin klusteriin myöhemmin tulleet ongelmat olisivat hajottaneet klusterin. Tästä syystä ainut järkevä ratkaisu oli laittaa hallintasolmu erilliselle koneelle. Ja ainut enää käytettävissä oleva kone oli Windows-kannettava.

MySQL-klusteri voidaan asentaa palvelimille, jotka käyttävät Windows-, Linux- tai Solaris-käyttöjärjestemää. Yhdessä klusterissa voidaan käyttää eri käyttöjärjestelmää käyttäviä palvelimia, mutta sitä suositellaan yhteensopivuuden varmistamiseksi käyttämään vain yhden käyttöjärjestelmän alaisuudessa. Tätä suositusta oli laiteresurssien rajallisuuden vuoksi pakko rikkoa. Rike ei kuitenkaan ollut merkittävä, koska kyseessä oli testiympäristö. Työn aikana eri käyttöjärjestelmien käyttämisestä klusterissa ei ilmennyt ongelmia. (Davies & Fisk 2006, 17.)

4.3 Klusterin asennus

4.3.1 Solmut

MySQL-klusterin solmuja ei asennettu tavanomaisen Windows- tai Linux-ohjelmiston tapaan. Sen vaatimat tiedostot tulivat suoraan ZIP-paketteina, jotka piti vain ladata ja purkaa sopivaan paikkaan. Pakettien kopioimisen lisäksi solmujen asennukseen liittyi solmujen asetusten määritys (luku 4.4). Kaikessa yksinkertaisuudessaan uusi solmu lisätään klusteriin

- luomalla solmulle klusterin tiedostoja varten sopivat hakemistopolut ja kopioimalla solmun tiedostot paikoilleen
- määrittämällä solmun *my.cnf*-tiedostoon hallintasolmun osoite sekä
- määrittämällä hallintasolmun *config.ini*-tiedostoon lisättävän solmun tiedot.

Uusi solmu voidaan käynnistää em. tehtävien jälkeen (luku 4.6). Koska klusterin solmujen asennus erosi hieman Windows- ja Linux-koneiden välillä, tässä luvussa käydään läpi klusterin asennus molemmille käyttöjärjestelmille.

Klusterin 64-bittiset paketit ladattiin molemmille käyttöjärjestelmille *dev.mysql.com*-sivustolta. Klusteria varten Windows-koneelle luotiin hakemisto *C:\usr\local* ja sen alle kansiot *cluster* ja *mysqlc*. Ladatut paketin purettiin *mysqlc*-kansioon. Windows-koneella käytettiin samaa hakemistopolkua kuin Linux-koneilla, jotta eri polusta ei aiheutuisi myöhemmin ongelmia.

Tämän jälkeen Winodws-koneen *cluster*-kansion alle luotiin kansiot *conf*, *mysql_data* ja *ndb_data*. Klusterin asetustiedostot sijoitettaisiin *conf*-kansioon ja MySQL:n tarvitsemat datatiedostot *mysql_data*-kansioon. Klusterin hallintasolmun loki- ja muut tiedostot tulisivat *ndb_data*-kansioon. Datasolmut käyttävät *ndb_data*-kansiota muista solmuista poiketen klusterin tietokannan väliaikaiseen säilytykseen. Lopuksi *mysqlc/data*-kansion sisältö kopioitiin kansioon *cluster/mysql_data*.

Linux-palvelimille luotiin samanlainen *cluster*-kansio alikansioineen */usr/local/*-hakemiston alle ja paketin purettiin kansioon */usr/local/mysqlc*. Linux-koneilla *mysqlc/data*-kansion sisältöä ei tarvinnut kopioida erikseen, koska ne tulisivat siirtymään paikoilleen tietokannan asennuksen yhteydessä.

Näin kaikki tarvittavat hakemistopolut oli luotu ja tarvittavat tiedostot kopioitu paikoilleen. Seuraava vaihe oli klusterin tietokannan asennus.

4.3.2 Tietokanta

Klusterissa SQL-solmut toimivat periaatteessa tavallisen MySQL-palvelimen tapaan. Ne toimivat samalla lailla myös siinä suhteessa, että ne tarvitsevat käyttöönsä ”mysql”-tietokannan, jonne tallennetaan SQL-solmujen prosessien tarvitsemia järjestelmän tietoja. Tämän vuoksi klusterin tietokantojen asennukset tapahtuvat SQL-solmuilta käsin.

Mysql-tietokannan asennus tuli tehdä kummallekin SQL-solmulle (ja usean SQL-solmun tapauksessa kaikille solmuille). Ladattujen ZIP-pakettien mukana tuli valmis skripti tieto-

kannan luomista varten. Tietokannan asennus molemmille SQL-solmuille onnistui ajamalla skripti kummallakin palvelimella komennolla *scripts/mysql_install_db --no-defaults –datadir=/usr/local/cluster/mysqld_data/*. Komento suoritettiin kansiossa */usr/local/mysqlc*. (Oracle Corporation 2012a, 1.)

4.4 Klusterin asetukset

Klusterin asetusten määrittämiseen käytetään kahta tiedostoa: *config.ini* ja *my.cnf*. Kumpikin asetustiedosto voi sijaita käytännössä missä hakemistossa tahansa. Jokaiselle solmulle luotiin asetustiedostoja varten kuitenkin oma kansio, jotta tiedostot eivät pääse hukkumaan. Klusteriin kuuluu pitkä lista erilaisia asetuksia. Suurta osaa niistä ei tarvittu tässä työssä ollenkaan, jonka vuoksi raportissa käydään läpi vain työn kannalta olennaiset asetukset.

4.4.1 Config.ini

Klusterin asetukset määritetään suurelta osin c*onfig.ini*-tiedoston avulla. Sen avulla määritetään kaikki klusteriin kuuluvat solmut ja erityyppisten solmujen asetukset. Tiedostoa säilytetään klusterin hallintasolmun kansiossa *cluster/conf*. Kaikki muut klusterin solmut ottavat käynnistyessään yhteyden hallintasolmuun ja saavat samassa yhteydessä asetukset *config.ini*-tiedostosta.

Jokaisella klusteriin kuuluvalla solmulla tulee olla tiedostossa oma asetuslohko. Asetuslohko alkaa riviltä, johon on hakasulkein merkitty ko. solmun prosessin nimi. Asetuslohko päättyy joko tiedoston loppuun tai ennen seuraavan lohkon alkua. Hallintasolmun prosessi on *ndb_mgmd*, datasolmun *ndbd* ja SQL-solmun *mysqld*. Esimerkiksi hallintasolmun asetuslohko alkaisi siis riviltä *[ndb_mgmd]*.

Useissa tapauksissa kaikille samantyyppisille solmuille (erityisesti datasolmuille) halutaan joitain samoja asetuksia. Solmuille yhteiset asetukset voidaan määrittää erillisen *default*-lohkon sisään sen sijaan, että jokaiselle yksittäiselle solmulle kopioitaisiin samat asetukset. Jokaiselle solmutyypille on mahdollista kirjoittaa oma *default*-lohko. Esimerkiksi datasolmujen yhteinen asetuslohko alkaisi riviltä *[ndbd default]*.

Koska testiympäristöön pystytettävässä klusterissa tulisi olemaan kaksi datasolmua, kaksi SQL-solmua ja yksi hallintasolmu, määritin *config.ini*-tiedostoon yhden hallintasolmun-, kaksi datasolmun- ja kaksi SQL-solmun asetuslohkoa. Lisäksi käytin datasolmujen *default*-lohkoa. Tässä työssä käytetty *config.ini*-tiedosto löytyy liitteestä 2.

4.4.1.1 Attribuutit

Jokaisella asetuksiin määritetyllä solmulla täytyy olla isäntänimi. Isäntänimi määritetään *hostname*-attribuutilla. Isäntänimet on mahdollista määrittää IP-osoitteilla tai DNS-nimillä. DNS-nimien käyttämistä ei kuitenkaan suositella, koska jos nimien selvityksessä ilmenee ongelmia, klusterin solmuihin ei saada yhteyttä. Tämän vuoksi nimien määrittämiseen käytettiin IP-osoitteita.

NodeId-attribuutilla määritetään solmun ID, jolla solmu yksilöidään klusterissa. ID-numerointia on käytettävä, jos samalla palvelimella aiotaan käyttää useita samantyyppisiä solmuja. Muussa tapauksessa se ei ole välttämätöntä, mutta sen käyttäminen helpottaa klusterin hallinnointia.

DataDir-attribuutilla määritetään, mitä kansiota solmu käyttää sen luomien tiedostojen säilytykseen. Datasolmut käyttävät tätä kansiota klusterin tietokannan väliaikaiseen säilytykseen ja hallintasolmut varastoivat sinne klusterin lokitiedostoja yms.

NoOfReplicas-attribuutilla määritetään yhdessä solmuryhmässä käytettävien kopioiden määrä. Solmuryhmiin ei tässä työssä tarvinnut sen suuremmin perehtyä. Työssä käytettiin yhtä solmuryhmää, joka on klusterissa vakioasetus. Koska työssä käytettiin kahta tietokantapalvelinta, kopioiden määrä oli kaksi.

MaxNoOfAttributes-attribuutilla määritetään, montako kenttää, indeksiä ja pääavainta klusterissa voi yhteensä olla. Oletusarvoisesti määrä on 1000, joka rikkoutuu suurempien tietokantojen kohdalla helposti. Attribuutilla ei ole maksimirajaa. Koska 1000 attribuutin raja meni rikki PHP-sovelluksen tietokannan kohdalla, attribuutin arvoksi määritettiin 50000.

DataMemory-attribuutilla määritetään, kuinka paljon RAM-muistia klusteri varaa käyttöönsä kaikesta käytettävissä olevasta muistista. Klusterin käyttöön ei ole viisasta varata kaikkea saatavilla olevaa muistia, jotta muille palvelimella pyöriville prosesseille riittää sitä. Klusterin kaikille datasolmuille tulisi määrittää samat *DataMemory*- ja *IndexMemory*-arvot, koska klusterin data jaetaan aina tasan kaikkien datasolmujen kesken. Jos Datasolmuilla käytetään eri määrää muistia, datasolmut käyttävät joka tapauksessa sen verran muistia, kuin vähimmillään on määritetty. Datasolmujen datamuistin määräksi asetettiin 512 Mt.

IndexMemory-attribuutilla määritetään, paljonko datasolmujen muistista varataan tilaa pääavaimille. Attribuutin määrittämiseen pätevät samat säännöt kuin *DataMemory*-attribuutillekin. Attribuutin arvoksi määritettiin 128 Mt.

MaxNoOfTables-attribuutilla määritetään klusterin tietokannan taulujen maksimimäärä. Tauluihin lasketaan mukaan myös klusterin itse luomat ylimääräiset taulut. Taulujen maksimimäärä nostettiin reilusti yli tarpeen, koska liian suuren arvon määrittämisestä ei olisi haittaa. Arvoksi määritettiin 1000. (Davies & Fisk 2006, 68 - 82.)

4.4.2 My.cnf

My.cnf on kunkin solmun omia asetuksia sisältävä tiedosto. Asetukset, jotka määritetään *config.ini*-tiedostossa, voitaisiin määrittää myös my.cnf:ssa, mutta se ei ole suositeltavaa. Kaikkien solmujen asetukset on hyvä säilyttää hallintasolmun hallussa, jotta asetukset eivät häviä, jos esim. jokin klusterin solmu hajoaa. *My.cnf* voi sisältää yksinkertaisimmillaan pelkän yhdistysrivin, jossa solmulle kerrotaan, missä IP-osoitteessa hallintasolmu sijaitsee. (Davies & Fisk 2006, 39 - 40.)

Tässä työssä *My.cnf*-tiedostot sisälsivät tiedon siitä, että

- klusterin solmut käyttävät klusteria käyttäjänimellä *mysql*
- klusterin hallintasolmu sijaitsi osoitteessa 192.168.10.2
- klusterin datatiedostojen hakemistopolku oli */usr/local/cluster/mysql_data*
- klusteriin yhdistämiseen tarvittava socket-tiedosto löytyi hakemistosta */usr/local/cluster/mysql_data/mysql.sock.*

4.5 PHP-sovelluksen tietokannan muokkaus

Klusterin asetusten määrittämisen jälkeen alkuperäinen suunnitelma oli laittaa PHP-sovelluksen tietokanta klusteriin ja PHP-sovellus olisi tämän jälkeen ollut käyttövalmis. Tietokannan tuonnissa klusteriin ilmeni kuitenkin ongelmia ensimmäisellä kerralla. Syy ongelmiin liittyi kappaleessa 4.1.3 kerrottuihin NDB-tietokantamoottorin rajoituksiin. Joissain PHP-sovelluksen tietokannan tauluissa käytettiin FULL-TEXT -indeksejä ja joistain puuttui pääavain.

Sovelluksen tietokantaa muutettiin siten, että

- kunkin taulun tietokantamoottori on NDB
- kullakin taululla on pääavain

- missään taulussa ei käytetä FULL-TEXT- tai VARCHAR-indeksejä
- kussakin taulussa käytetään korkeintaan yhtä juoksevaa numerointia.

Kaikki em. muutokset tehtiin tietokannanluontitiedostoon (SQL-tiedosto). Koska tietokantamoottori tuli vaihtaa kaikkiin tietokannan tauluihin, muutoksen tekemiseen käytettiin PHP-skriptiä, joka teki muutoksen jokaiseen tauluun automaattisesti. Muut muutokset tehtiin tiedostoon käsin, koska skriptin tekeminen olisi vienyt liikaa aikaa.

Hakemistojen luonnin, pakettien purkamisen ja asetusten muokkaamisen jälkeen klusterin solmut oli asennettu.

4.6 Klusterin käynnistäminen

Asetustiedostojen luonnin ja PHP-sovelluksen tietokantamuutosten jälkeen klusterin solmut käynnistettiin ja tietokannan tuonti klusteriin aloitettiin. Klusterin solmujen käynnistäminen tapahtui kutsumalla solmun prosessia komentoriviltä. Jotta komennon antamiseksi ei joka kerta tarvinnut kirjoittaa prosessin koko hakemistopolkua, prosessien kansio (*mysqlc/bin*) lisättiin Windows-koneella järjestelmämuuttujiin ja Linux-koneilla root-käyttäjän BASH-profiiliin.

Klusterin solmujen käynnistämisen aloitettiin hallintasolmusta. Solmu käynnistettiin Windows-koneen komentoriviltä seuraavalla komennolla: *start /B ndb_mgmd -f c:\usr\local\cluster\conf\config.ini --initial --configdir=c:\usr\local\cluster\conf*.

Initial-arvoa käytetään vain ensimmäisen käynnistyksen yhteydessä tai kun halutaan jostain muusta syystä tuoda tietokanta uudestaan klusteriin. Hallintasolmun käynnistymisen jälkeen itse hallintaohjelma käynnistettiin komentoriviltä komennolla *ndb_mgm*.

Hallintasolmun käynnistymisen jälkeen DB-koneiden datasolmut käynnistettiin komennolla *ndbd –initial*. Datasolmut tuli käynnistää yhtä aikaa, jotta hallintasolmu sai klusterin käynnistettyä. Hallintaohjelmaan tuli ilmoitus, kun molemmat datasolmut käynnistyivät. Datasolmujen jälkeen oli jäljellä enää SQL-solmujen käynnistys. Aluksi WEB-koneiden erillisiin terminaali-ikkunoihin käynnistettiin *mysqld*-palvelu, eli MySQL-palvelin komennolla *mysqld --defaults-file=/usr/local/cluster/conf/my.cnf –u root*. Palvelimien käynnistymisen jälkeen klusterin datasolmuihin päästiin käsiksi WEB-palvelimelta normaalin MySQL-palvelimen tapaan komennolla *mysql –u root*. Klusterin käsittelyn pystyi tekemään kummalla WEB-palvelimella tahansa.

Edellisessä *mysql*-komennossa huomion arvoinen seikka on se, että normaali MySQL-palvelimen asiakasohjelma käynnistetään tismalleen samalla komennolla. Jos klusterin *mysqlc/bin*-kansiota ei olisi lisätty BASH-profiiliin, tuo sama komento olisi avannut väärän MySQL-asiakasohjelman.

Nyt kun kaikki klusterin osat olivat toiminnassa ja klusteriin oli päästy käsiksi, verkkosovelluksen tietokannan tuonti klusteriin aloitettiin. Tietokannanluontitiedosto laitettiin WEB 1 -palvelimen hakemistoon */var/tmp* ja ennen *mysql*-komennon antamista siirryttiin tuohon kansioon. Kansioon siirtymisen ja *mysql*-komennon antamisen jälkeen verkkosovellusta varten luotiin tietokanta komennolla *create database kanta* ja se otettiin käyttöön komennolla *use kanta*. Tämän jälkeen annettin komento *source luokanta.sql*, jossa ”*luokanta.sql*” oli tietokannanluontitiedoston nimi. Komento otti juuri luodun tietokannan käyttöön.

Komennon antamisen jälkeen tietokannan tuonti klusteriin alkoi. Tietokannan tuonti onnistui kantaan tehtyjen muutosten ansiosta ja näin ollen verkkosovelluksen tietokanta saatiin muutettua käyttämään MySQL-klusteria. (Oracle Corporation 2012b, 2.)

5 PHP-SESSIOIDEN HALLINTA

Tässä luvussa käsitellään PHP-verkkosovelluksen istuntojen hallintaa ja työssä siihen liittyneitä ongelmia. Aiheeseen liittyviä työvaiheita olivat istuntojenhallintamenetelmän valinta, toteutus ja käyttöönotto. Tässä kappaleessa käytetyillä käsitteillä *luokka* ja *olio* viitataan olio-ohjelmoinnin luokkiin ja ilmentymiin.

5.1 Sessioiden hallinnan ongelma

Klusterin pystyttämisen jälkeen uusi verkkosovelluksen arkkitehtuuri vaati lopuksi PHP-sessioiden hallinnan toteutuksen. Kuten aiemmin on käynyt ilmi, arkkitehtuurimuutoksen kohteena ollut verkkosovellus on toteutettu PHP-kielellä. Sovellus käyttää sessioita (ts. istuntoja) käyttäjäkohtaisten tietojen tallentamiseen istunnon ajaksi. Istunto alkaa, kun käyttäjä navigoi selaimella sovelluksen verkkosivulle. Istunto päättyy, kun käyttäjä sulkee selaimen tai kirjautuu sovelluksesta ulos.

WEB-palvelin yksilöi kunkin käyttäjän istunnon satunnaisella, n. 30 merkkiä pitkällä ID:llä. Istunnon tiedot tallentuvat oletuksena paikallisesti WEB-palvelimelle tiedostoon *sess_ID*, missä *ID* on palvelimen arpoma ID. Palvelimen luoma ID välitetään istunnon aloituksen yhteydessä käyttäjän selaimelle keksinä. (Williams & Lane 2004, 339.)

Verkkosovelluksen uusi kahden sovelluspalvelimen arkkitehtuuri aiheutti sovelluksen käytettävyyteen ongelman: Mistä WEB 2 -palvelin tietää, että käyttäjä on käyttänyt sovellusta hetki sitten palvelimella WEB 1, kun DNS Round Robin vaihtaa palvelinta, joka käyttäjän selaimelle vastaa? Ilman sessioiden hallinnan muokkausta yhdelle käyttäjälle syntyy kaksi sessiota, yksi kummallekin palvelimelle. Tämä tilanne ei tietenkään ollut toivottu, joten sessioiden hallinta täytyi toteuttaa eri tavalla.

Sessioiden hallinnan tuli toimia samalla tavoin vikasietoisesti kuin MySQL-klusterin solmut. Sessiotietoja ei siis saanut hävitä, jos toinen WEB-palvelimista kaatuisi. Ainut järkevä tapa toivotun tilanteen saavuttamiseksi oli tallentaa sessiotiedot sovelluksen tietokantaan. WEB-palvelimet tuli määrätä tallentamaan ja lukemaan sessiotiedot tietokannasta paikallisen sessiotiedoston sijaan. Tällä tavoin kumpikin WEB-palvelin tietäisi, jos ko. käyttäjä on käyttänyt sovellusta äskettäin toisella palvelimella. (Smith 2007.)

5.2 Sessiotaulu

Sessiotietojen säilyttämistä varten verkkosovelluksen tietokantaan täytyi luoda uusi taulu. Tauluun talletettavia tietoja olivat istunnon ID (session_id), joka yksilöi kunkin käyttäjän istunnon, istunnon tiedot (data) sekä tieto siitä, milloin istunnon tietoja on viimeksi päivitetty (usetime). Päivitysaika päätettiin lisätä tauluun, jotta taulusta olisi helppo putsata ns. ”vanhentuneet” istunnot pois. Istunto määritettiin vanhentuneeksi, jos sitä ei ole päivitetty 12 tunnin sisällä. Istuntojen vanhenemisajan määritys tehtiin WEB-palvelinten php.ini-tiedostoon seuraavalla rivillä: *session.gc_maxlifetime = 43200*. Istuntojen säilyttämistä varten luotiin taulu käyttämällä *phpMyAdmin*-ohjelmaa.

5.3 Istuntojen hallinnan toteutus

Tietokantataulun luonnin jälkeen verkkosovellusta täytyi muokata siten, että se käyttäisi juuri luotua taulua istuntojen säilyttämiseen paikallisten tiedostojen sijaan. Aluksi tuli selvittää, kuinka moinen muutos olisi mahdollista toteuttaa. Hyväksi onneksi kävi ilmi, että PHP-tulkista löytyi vakiona valmiita funktioita, joita käyttämällä se voidaan määrätä käyttämään tietokantaa istuntojen tallennukseen. Valmiista funktioista huolimatta sovellusta varten täytyi kirjoittaa uusi istuntojenhallintaluokka ja ottaa se käyttöön paikoissa, joissa istuntoja käytetään. Luokalle annettiin nimi *sessionmanager*.

5.3.1 Sessionmanager-luokka

PHP-tulkin käsikirjassa oli valmiiksi määritelty, mitä funktioita luokan tuli sisältää ja mitä kustakin funktiosta tulisi palauttaa, kun sitä kutsutaan. Tarvittavia funktioita olivat istunnon *avaus*, *sulkeminen*, *lukeminen*, *kirjoittaminen*, *tuhoaminen* ja *putsaaminen*. Käytännössä istuntojen lukemisfunktio on ainut funktio, josta palautetaan tietoa. Muista funktioista palautetaan vain tieto funktion suorituksen onnistumisesta.

5.3.1.1 Avaus

Kun verkkosovelluksen istunto avataan *session_start*-funktiolla, PHP-tulkki kutsuu istunnon avausfunktiota, joka toimii luokan *rakentajana* eli konstruktorina. Avausfunktiossa ei olisi käytännössä tarvinnut tehdä itse mitään, koska PHP-tulkki hoitaa automaattisesti istunnon avauksen funktion kutsun jälkeen. Funktiossa käydään kuitenkin hakemassa php.ini-tiedostosta globaaliin muuttujaan tieto siitä, kuinka kauan käyttämättömiä istuntoja tietokannassa säilytetään.

5.3.1.2 Luku

Kun istunto on avattu, PHP-tulkki kutsuu automaattisesti istunnon lukufunktiota. Lukufunktion tuli palauttaa tulkille samanlainen merkkijono, joka olisi palautunut tiedostosta, jonne tulkki tavallisesti olisi tallentanut istunnon tiedot. Tulkki pyytää tietoa aina istunnon ID:n perusteella, jonka se välittää lukufunktiolle. Koska uudessa arkkitehtuurimallissa istunnon tiedot säilytetään tietokannassa, tietojen luvun täytyi tietysti tapahtua tietokannan sessiotaulusta.

Lukufunktioon luotiin SQL-lause, jonka suorittamalla tietokannasta haetaan tulkin välittämän istunnon ID:n perusteella sen tiedot, jos istunto ei ole vanhentunut. Istunnon voimassaolo tarkistetaan vertaamalla istunnon muokkausajan ja nykyisen ajan välistä erotusta tulkin asetuksissa määritettyyn voimassaoloaikaan. Jos erotus on pienempi kuin asetusarvo, istunto on voimassa. Kaikki ajat on esitetty sekunteina. SQL-lauseen palauttama tieto otetaan kiinni muuttujaan ja se palautetaan PHP-tulkille. Jos tietokannasta ei löydy ko. istunnon ID:llä mitään, PHP-tulkille välitetään tyhjä tieto.

5.3.1.3 Kirjoitus

Istunnon kirjoitusfunktiota kutsutaan, kun istunnon tietoja ollaan tallentamassa ja istuntoa sulkemassa. PHP-tulkki välittää kirjoitusfunktiolle istunnon ID:n ja kirjoitettavan istuntotiedon. Kirjoitusfunktiosta palautetaan ainoastaan tieto kirjoituksen onnistumisesta.

Istuntojen kirjoitus tapahtuu samalla periaatteella, kuin niiden lukeminen. Koska tiedot tallennetaan tietokantaan, kirjoittaminen tapahtuu SQL-lauseella. Kirjoituksessa täytyy ottaa huomioon, ollaanko tallentamassa uutta istuntoa vai päivittämässä jo olemassa olevaa. Koska päivittämisessä ja uuden tiedon lisäyksessä käytettävä SQL-lause on erilainen, tilanne täytyy ensin tarkistaa lukemalla tietokannasta tulkin välittämää ID:tä vastaava tietokannan rivi. Jos kysely palauttaa tyhjän tiedon, ollaan lisäämässä uutta riviä. Muussa tapauksessa ollaan päivittämässä vanhaa tietoa.

Tietokantaan tehtävä kysely muodostetaan ja suoritetaan käsillä olevan tapauksen mukaan. Kirjoituksen onnistumisesta välitetään tieto PHP-tulkille.

5.3.1.4 Sulkeminen

Istunnon sulkemisfunktio toimii luokan *hajottajana* eli destruktorina. Sulkemisfunktiota

kutsutaan automaattisesti istunnon kirjoituksen jälkeen. PHP-tulkki hoitaa itse kaiken istunnon sulkemiseen liittyen, joten funktiossa ei tarvinnut tehdä muuta, kuin palauttaa ”tosi”.

5.3.1.5 Tuhoaminen

PHP-tulkki kutsuu istunnon tuhoamisfunktiota, kun koodissa käytetään funktiota *session_destroy*. Tulkki välittää tuhoamisfunktiolle istunnon ID:n, jonka perusteella istunto tuhotaan. Normaalisti *session_destroy*-funktio poistaa kaiken tiedon, joka kyseiselle istunnolle on tallennettu. Tästä syystä tuhoamisfunktioon täytyi luoda samanlainen toiminnallisuus. Tiedot tuhottiin SQL-lauseella, joka päivittää ko. istunnon datakentän arvon tyhjäksi. Funktiosta palautetaan tieto kyselyn onnistumisesta.

5.3.1.6 Putsaaminen

PHP-tulkki kutsuu itse sattumanvaraisesti putsausfunktiota aika ajoin. Putsauksen satunnaisuus määritetään php.ini-tiedostossa asetuksilla *session.gc_probability* ja *session.gc_divisor*. Putsausfunktion tehtävä on poistaa vanhentuneet istunnot. Poisto tietokannasta tapahtuu SQL-lauseella, joka suoritetaan, kun funktiota kutsutaan. Funktio palauttaa tiedon kyselyn onnistumisesta. (Achour, Betz & muut 2013b.)

5.3.2 Sessionmanager-olio

Istuntojenhallintaluokan käyttäminen verkkosovelluksessa tapahtui luomalla luokasta olio niihin paikkoihin, joissa istuntoja avataan. Luotuun olioon liitettiin *sessionmanager*-luokka ja se määritettiin istuntojen tallentajaksi PHP-tulkin sisäisellä funktiolla *session_set_save_handler*, joka sai parametrinaan juuri luodun olion sekä istuntojen hallinnassa käytettävien funktioiden nimet.

Olion luonnin jälkeen istuntoja käytettiin koodissa tismalleen samalla lailla kuin ennenkin, paitsi että nyt istunnot tallennettiin tietoineen MySQL-klusteriin. Kun luokkatiedosto ja muut muutokset lisättiin kummallekin WEB-palvelimelle, ne alkoivat käyttää MySQL-klusteria istuntojen tietojen tallentamiseen ja lukemiseen.

Luokkatiedosto ja *sessionmanager*-olion luontiin tarvittavat koodit löytyvät tämän työn liitteestä 3.

6 ARKKITEHTUURIN TESTAUS

Viimeisenä vaiheena tässä työssä oli arkkitehtuurimuutosten testaus. Testit suoritettiin kahteen otteeseen sekä WEB- että tietokantapalvelimille.

6.1 Testaus

WEB- ja tietokantapalvelimien testaus suoritettiin kaksi kertaa eri päivinä. Arkkitehtuurimuutoksen ansiosta verkkosovelluksen tuli toimia silloin, kun vähintään yksi WEB- ja tietokantapalvelin olivat ylhäällä. Testeissä palvelimilta irrotettiin verkko- ja sähköjohtoja tietyssä järjestyksessä.

Aluksi testattiin WEB-palvelimien ja DNS Round Robinin toiminta ottamalla WEB-palvelimilta sähkö- tai verkkojohto irti yksi kerrallaan. WEB-palvelinten jälkeen tietokantapalvelimet testattiin samalla periaatteella. Lopuksi testattiin vielä arkkitehtuurin toiminta yhden WEB- ja tietokantapalvelimen ollessa ylhäällä. Verkkosovellusta käytettiin jatkuvasti sillä välin, kun testejä tehtiin.

6.2 Tulokset

Ensimmäisellä testauskerralla arkkitehtuuri toimi tismalleen odotetulla tavalla. Kun yksi WEB-palvelin oli alhaalla, toinen palvelin vastaili käyttäjän pyyntöihin normaalisti. WEB-palvelimen vaihtaminen aiheutti enimmillään n. 30 sekunnin käyttökatkon.

Tietokantapalvelimien testaus onnistui lähes yhtä odotetuin tuloksin. Testeissä huomattiin, että jos pääasiallinen tietokantapalvelin kaatuu, toissijaisella palvelimella menee hetki, ennen kuin se siirtyy pääasialliseksi palvelimeksi. Tänä aikana järjestelmä ei vastannut käyttäjän pyyntöihin. Tietokantapalvelinten viiveiltä olisi todennäköisesti säästytty, jos klusteriin olisi kuulunut enemmän datasolmuja. Täysin sammuneen datasolmun uudelleenkäynnistyminen kesti sovelluksen tietokannan kanssa n. 5 minuuttia.

Toisella testauskerralla WEB-palvelinten testauksessa ilmeni huomattavampia ongelmia. Jostain syystä toisen WEB-palvelimen tiputtaminen aiheutti sen, ettei käyttäjä pystynyt käyttämään sovellusta huomattavan pitkään aikaan. Ongelman todellista syytä ei alettu selvittää.

Todennäköisin syy ongelmaan on se, että testiympäristön reitittimet oli sammutettu ennen testausta joksikin aikaa. Reitittimien sammumisen vuoksi toinen tai molemmat ulkoiset IP-osoitteet olisivat saattaneet vaihtua ja määritetyt verkko-osoitteiden käännökset (NAT) eivät olisi pitäneet paikkaansa, koska ne oli määritetty käsin.

Toisella testauskerralla tietokantapalvelimet toimivat samalla lailla kuin ensimmäisessä testissä. Kokonaisuudessaan testitulokset olivat toisen testauksen WEB-palvelinongelmia lukuun ottamatta halutunlaisia.

7 YHTEENVETO

Työn tavoitteena ollut verkkosovelluksen arkkitehtuurimuutos onnistui suunnitelmien mukaan. Arkkitehtuurimuutoksen ansiosta verkkosovellusta pystyttiin käyttämään yhdestä verkko-osoitteesta, jonka takaa vastasi vuorotellen kaksi sovelluspalvelinta. Yhden WEB- ja/tai tietokantapalvelimen sammuminen ei haitannut sovelluksen toimintaa merkittävästi. Parempiin tuloksiin olisi varmasti päästy, jos käytettävissä olisi ollut enemmän palvelimia.

DNS Round Robin oli työn tilaajan puolelta tarjottu tekniikka, jota käytettiin WEB-palvelinten kuorman tasaukseen. Työn edetessä kävi ilmi, että se ei ota kantaa siihen, onko siihen liitetyt palvelimet ylhäällä vai eivät. Se ohjasi pyynnöt vuorotellen eri palvelimille 30 sekunnin välein. Tekniikka toimi testiympäristössä riittävän hyvin, mutta tuotantokäytössä sen toiminta saattaa aiheuttaa arkkitehtuuriin vikatilanteessa heikon lenkin.

LÄHTEET

Achour, M., Betz, F. & muut. 2013a. What is PHP? [verkkojulkaisu]. The PHP Group [viitattu 26.4.2013]. Saatavissa:
http://www.php.net/manual/en/intro-whatis.php

Achour, M., Betz, F. & muut. 2013b. Session Set Save Handler [verkkojulkaisu]. The PHP Group [viitattu 26.4.2013]. Saatavissa:
http://www.php.net/manual/en/function.session-set-save-handler.php

The Apache Software Foundation 2012. Apache HTTP Server Project [verkkojulkaisu]. The Apache Software Foundation [viitattu 26.4.2013]. Saatavissa:
http://httpd.apache.org/

Davies, A. & Fisk, H. 2006. MySQL Clustering. Indiana: MySQL Press.

Galuschka, C. 2012. IPTables [verkkojulkaisu]. The CentOS Team [viitattu 26.4.2013]. Saatavissa:
http://wiki.centos.org/HowTos/Network/IPTables

Horman, S. 2005. Ultra Monkey [verkkojulkaisu]. Horms Solutions [viitattu 26.4.2013]. Saatavissa:
http://www.ultramonkey.org/

Oracle Corporation 2012a. MySQL Cluster Quick Start Guide – LINUX [verkkodokumentti]. Oracle Corporation [viitattu 26.4.2013]. Saatavissa:
http://downloads.mysql.com/tutorials/cluster/mysql_wp_cluster_quickstart_linux.pdf

Oracle Corporation 2012b. MySQL Cluster Quick Start Guide – Windows [verkkodokumentti]. Oracle Corporation [viitattu 26.4.2013]. Saatavissa:
http://downloads.mysql.com/tutorials/cluster/mysql_wp_cluster_quickstart_windows.pdf

The PHP Group 2013. PEAR – PHP Extension and Application Repository [verkkojulkaisu]. The PHP Group [viitattu 26.4.2013]. Saatavissa:
http://pear.php.net/

Smith, R. 2007. Storing PHP Sessions in a Database [verkkojulkaisu]. Dev Shed [viitattu 26.4.2013]. Saatavissa:
http://www.devshed.com/c/a/PHP/Storing-PHP-Sessions-in-a-Database/

Wikipedia 2013. Round-robin DNS [verkkojulkaisu]. Wikipedia Foundation [viitattu 26.4.2013]. Saatavissa:
http://en.wikipedia.org/wiki/Round-robin_DNS

Williams, H. E. & Lane, D. 2004. Web Database Applications with PHP and MySQL. California: O'Reilly Media Inc.

Linux-palvelinten palomuurien asetusskripti

Alla oleva skripti on kopio WEB 1 -palvelimella käytetystä palomuurin asetusskriptistä. Muilla Linux-palvelimilla skriptit olivat IP-osoitteita lukuun ottamatta samanlaisia.

```
#!/bin/bash
# Iptables asetus skripti
#
# Laitetaan oletuskäytäntö sisääntulevalle liikenteelle
# väliaikaisesti sallivaksi, jotta nykyisten asetusten
# tyhjentäminen ei katkaise etähallintayhteyttä (SSH)
#
 iptables -P INPUT ACCEPT
#
# Tyhjennetään kaikki tämänhetkiset iptables säännöt
#
 iptables -F
#
# Sallitaan SSH liikenne TCP porttiin 22
# Tämä tarvitaan etähallintaa varten
# Sallitaan lisäksi verkkoliikenne TCP porttiin 80
#
 iptables -A INPUT -p tcp --dport 22 -j ACCEPT
 iptables -A INPUT -p tcp --dport 80 -j ACCEPT
#
# Sallitaan MySQL klusterin liikenne molempiin suuntiin
# TCP ja UDP porteista 1186 ja 5000
#
 iptables -A INPUT -p tcp --dport 1186 -j ACCEPT
 iptables -A INPUT -p udp --dport 1186 -j ACCEPT
 iptables -A INPUT -p tcp --dport 5000 -j ACCEPT
 iptables -A INPUT -p udp --dport 5000 -j ACCEPT
#
 iptables -A OUTPUT -p udp --dport 1186 -j ACCEPT
 iptables -A OUTPUT -p tcp --dport 1186 -j ACCEPT
 iptables -A OUTPUT -p udp --dport 5000 -j ACCEPT
 iptables -A OUTPUT -p tcp --dport 5000 -j ACCEPT
#
# Määritetään oletuskäytännöt sisääntulevalle (INPUT),
# välitettävälle (FORWARD) ja ulos lähtevälle (OUTPUT)
# liikenteelle. Sallitaan oletuksena vain ulos lähtevä
# liikenne.
#
 iptables -P INPUT DROP
 iptables -P FORWARD DROP
 iptables -P OUTPUT ACCEPT
#
# Sallitaan liikenne paikalliselta käyttäjältä
#
 iptables -A INPUT -i lo -j ACCEPT
#
# Sallitaan paketit, jotka tulevat sisään avattua yhteyttä
# pitkin tai jotka liittyvät johonkin avattuun yhteyteen
#
 iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
#
#
```

```
# Sallitaan liikenne verkko-osoitteista 10.0.0.0/24 ja
# 192.168.10.0/24
#
 iptables -A INPUT -s 192.168.10.0/24 -j ACCEPT
 iptables -A INPUT -s 10.0.0.0/24 -j ACCEPT
#
# Sallitaan eth0 liittimeen sisään tuleva ping.
# HUOM! IP-osoiteeksi ko. palvelimen liittimen IP
#
 iptables -A INPUT -p icmp --icmp-type 8 -s 0/0 -d 192.168.10.7 -
m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
 iptables -A OUTPUT -p icmp --icmp-type 0 -s 192.168.10.7 -d 0/0
-m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Sallitaan eth0 liittimestä ulos lähtevä ping
# HUOM! IP-osoiteeksi ko. palvelimen liittimen IP
#
 iptables -A OUTPUT -p icmp --icmp-type 8 -s 192.168.10.7 -d 0/0
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
 iptables -A INPUT -p icmp --icmp-type 0 -s 0/0 -d 192.168.10.7 -
m state --state ESTABLISHED,RELATED -j ACCEPT
#
# Tallennetaan asetukset
#
 /sbin/service iptables save
#
# Listataan säännöt lopuksi ruudulle
#
 iptables -L –v
```

MySQL-klusterin config.ini

Klusterin config.ini tiedosto sijaitsi tässä työssä Windows-koneen (eli hallintasolmun) hakemistossa *C:\usr\local\cluster\conf*

```
# Hallintasolmu (windows)
[ndb_mgmd]
hostname=192.168.10.2
datadir=/usr/local/cluster/ndb_data
NodeId=1

# Yleiset asetukset kaikille datasolmuille
[ndbd default]
noofreplicas=2
datadir=/usr/local/cluster/ndb_data
maxnoofattributes=50000
datamemory=850MB
indexmemory=128MB
MaxNoOfOrderedIndexes=8000
maxnooftables=800

# DB 1
[ndbd]
hostname=192.168.10.9
NodeId=9

# DB 2
[ndbd]
hostname=192.168.10.10
NodeId=10

# WEB 1
[mysqld]
hostname=192.168.10.7
NodeId=7

# WEB 2
[mysqld]
hostname=192.168.10.8
NodeId=8
```

Sessionmanager-luokka ja olion luonti

Sessionmanager-luokka

```
<?php

class sessionmanager
{

    public $life_time;

    function open($save_path, $session_name)
    {

            // Luetaan max session elinaika php.inistä
            $this->life_time = get_cfg_var("session.gc_maxlifetime");

            register_shutdown_function('session_write_close');
            return true;
    }

    function close()
    {
            return true;
    }

    function read($session_id)
    {
            // Avataan yhteys ja asetetaan nykyinen aika
            $conn = db_conn();
            $time = date('Y-m-d H:i:s');

            // Sessiodatan haku sql
            $read_sql = "SELECT data, usetime FROM sessio WHERE session_id =
'{$session_id}' AND time_to_sec(timediff(sysdate(), usetime)) < '{$this-
>life_time}'";

            // Haetaan sessiodata taulusta
            $rs = $conn->query($read_sql);
            $a = $rs->numRows();

            if($a > 0)
            {
                        while($row = $rs->fetchRow (DB_FETCHMODE_ASSOC))
                        {
                            $data = $row['data'];
                        }
            }

            return $data;

    }

    function write($session_id, $data)
    {
        // Asetetaan nykyinen aika ja avataan yhteys
        $usetime = date('Y-m-d H:i:s');

        $sql = "SELECT * from sessio where session_id = '{$session_id}'";

        $conn = db_conn();

        if($result = $conn->query($sql))
        {
            $data = mysql_real_escape_string($data);
            if($result->numRows() > 0 && $data != '')
            {
               $update_sql = "UPDATE sessio SET session_id = '{$session_id}',
data = '{$data}', usetime = '{$usetime}'";
```

```
               $rs = $conn->query($update_sql);
             }
             else
            {
              $new_sql = "INSERT INTO sessio (session_id,data,usetime) VA-
LUES('{$session_id}','{$data}','{$usetime}')";
    $rs = $conn->query($new_sql);
             }

       }

         return true;
   }

   function destroy($session_id)
   {
         // Session tuhoamis sql
         $sql = "DELETE FROM sessio WHERE session_id = '{$session_id}'";
         $conn = db_conn();

         $rs = $conn->query($sql);

         return true;

   }

   function gc()
   {
         // Garbage Collection (Roskien poisto)
         // sql lause jolla poistetaan kaikki vanhentuneet sessiot

         $sql = "DELETE FROM sessio WHERE time_to_sec(timediff(sysdate(), useti-
me)) > {$this->life_time}";
         $conn = db_conn();

         $rs = $conn->query($sql);

         // Palautetaan aina tosi
         return true;

   }

}

?>
```

Sessionmanager-olion luonti

```
<?php
include_once ('sessionmanager.php');

global $sess;
$sess = new sessionmanager();
session_set_save_handler(
                                    array( &$sess, "open" ),
                                    array( &$sess, "close" ),
                                    array( &$sess, "read" ),
                                    array( &$sess, "write"),
                                    array( &$sess, "destroy"),
                                    array( &$sess, "gc" )
                          );
session_start();
?>
```


SAVONIA