Sami Vartio

Avoimen lähdekoodin HA-tietokannat

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

1.1.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Sami Vartio Avoimen lähdekoodin HA-tietokannat 46 sivua + 4 liitettä 1.1.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja	Lehtori Pasi Ranne

Insinöörityössä perehdyttiin avoimen lähdekoodin periaatteisiin ja lisenssimalleihin sekä huomioitiin liiketaloudelliset lähtökohdat.

Tavoitteena oli tutustua järjestelmän saatavuuteen sekä klusteroinnin periaatteisiin. Järjestelmän saatavuus (HA) on tietojärjestelmien suunnittelussa käytettävä käytäntö, joka pyrkii siihen että järjestelmä on aina käyttäjän käytettävissä. Aluksi käsiteltiin perustietoja sekä teoriaa. Tutkimuksen edetessä paneuduttiin laajemmin yleisimpiin avoimen lähdekoodin tietokantaratkaisuihin, tulevaisuuden näkymiin ja uusiin tuotteisiin.

Yleisimmät relaatiotietokannat, kuten PostgreSQL, MySQL sekä MariaDB soveltuivat hyvin kriittisiin tuotantojärjestelmiin, ja näistä kaikista voitiin toteuttaa vikasietoinen klusteri eli suorituskykyinen järjestelmä. Työssä tutustuttiin Red Hatin sekä muiden organisaatioiden tuotteisiin ja panostukseen. Relaatiotietokantojen lisäksi perehdyttiin myös NoSQL-tietokantatuotteiden toteutuksiin.

Insinöörityössä tutkittiin myös muistinvaraista analytiikkaa sekä Big datan keskeisiä tiedon tarpeita. Samalla tutustuttiin Hadoop-projektin ratkaisuihin pintapuolisesti. Tämän lisäksi perehdyttiin pilvipalveluihin sekä tiedon tallentamista pilveen teoriatasolla ja virtualisointiin yleisellä tasolla.

Lopuksi pohdittiin periaatteita ja sopimuksia, jotka olivat tärkeässä roolissa korkeakäyttöistä järjestelmän saatavuutta valittaessa. Tuloksena huomattiin, kuinka sopimukset vaikuttivat suoraan hintatasoon.

Avainsanat	avoin lähdekoodi, korkea saatavuus, tietokannat, NoSQL, Big data, pilvipalvelut

Author Title Number of Pages Date	Sami Vartio Open source HA databases 46 pages + 4 appendices 1 January 2014
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Specialisation option	Data Networks
Instructor	Pasi Ranne, Senior Lecturer

This thesis examines in open-source principles and licensing models and takes into account the economic conditions of business.

The purpose was to explore the high availability of the system as well as clustering in general. High availability (HA) is a system design approach and associated service implementation that ensures a prearranged level of operational performance will be met during a contractual measurement period. The thesis reviewed basic information and theory. The study focused more broadly on the most common open source database solutions, future prospects and new products.

The most common relational databases such as PostgreSQL, MySQL and MariaDB were well suited for critical production systems, and all of these allowed implementing a fault-tolerant cluster which means to the performance of the system. Products of Red Hat and other organisations were included in this study. In addition to relational databases, the NoSQL database product implementations were examined.

The thesis also examined memory analytics and the key information needs regarding Big data. At the same time, Hadoop project solutions were acquainted with superficially. In addition, cloud services and data storage to the cloud at the level of theory and virtualization in general were analysed.

Finally, the principles and agreements of the high availability systems were discussed, as they played an important role in the high-to-use system for the selection of access. The results show that the contracts were directly affected by the price level.

Keywords	open source, high availability, databases, NoSQL, Big data, clouds

Sisällys

Lyhenteet

1 Johdanto 1

2 Avoin lähdekoodi 2

2.1 Avoimen lähdekoodin lisenssit 3
2.2 OSI-malli 4
2.3 Tilastotietoa avoimen lähdekoodin ohjelmista 6

3 Järjestelmän saatavuus 8

4 Klusterointi 9

5 Järjestelmän saatavuuden takaaminen 12

5.1 Palvelutasosopimukset 12
5.2 Verkkopalvelutyypit 13

6 Avoimen lähdekoodin relaatiotietokannat 15

7 Järjestelmän saatavuuden vaihtoehtoja relaatiotietokannoille 19

7.1 Varatietokannat 19
7.2 Laitteiston klusterointi 21
7.3 Replikointi tietokantatasolla 24
7.4 Kuormantasaus ja yliheitto 28

8 NoSQL 32

9 Big data 36

9.1 Hadoop 37
9.2 Muistinvarainen analytiikka 38
9.3 Keskeiset tiedontarpeet ja kehityskohteet 38

10 Pilvipalvelut 40

11 Pohdinta 45

Lähteet 47

Liitteet

Liite 1. MySQL MasterSlave-konfigurointi

Liite 2. GaleraCluster-konfiguraatio

Liite 3. PostgreSQL Streaming-replikointi

Liite 4. MongoDB:n replicat-asennus

Lyhenteet

ACID	*Atomicity, Consistency, Isolation, Durability.* Tietokantajärjestelmien periaate, jonka avulla turvataan järjestelmän tietojen eheys kaikissa tilanteissa.
API	*Application programming interface.* Ohjelmointirajapinta, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja keskenään.
BSD	*Berkeley Software/System Distribution.* Avoimen lähdekoodin lisenssi.
DDL	*Data Definition Language.* Kieli, jonka avulla kuvataan tietokannan määritteitä, erityisesti taulukoita, kenttiä, indeksejä ja tallennustapoja.
DOCSIS	*Data over Cable Service Interface Specification.* Spesifikaation mukaan rakennetut kaapelimodeemiverkot.
DR	*Disaster Recovery.* Vikasietoisuus.
DRBD	*Distributed Replicated Block Device.* Ohjelmisto, joka mahdollistaa kovalevyjen peilaamisen verkon yli.
GFS	*Global File System.* Maailmanlaajuinen tiedostojärjestelmä.
GPL	*General Public License.* Avoin lisenssi.
HA	*High Availability.* Järjestelmän saatavuus.
HADR	*High Availability Disaster Recovery.* Tietokannan vikasietoisuus.
iSCSI	*Internet Small Computer System Interface.* Kiintolevy tai levyvaranto, jota käytetään IP-verkossa SCSI -komentojen avulla.
LGPL	*Lesser General Public Licence.* Avoimen lähdekoodin lisenssi.
LVM	*Logical Volume Manager.* Linuxin ytimen loogisten taltioiden hallintakomponentti.

LVS	*Linux Virtual Server.* Virtuaalipalvelin.
MIT	*Massachusetts Institute of Technology.* Vapaa ohjelmistolisenssi.
MVCC	*Multiversion Concurrency Control.* Samanaikaisuuden hallintamenetelmä.
NAT	*Network Address Translation.* Osoitteenmuunnos.
NDB	*Network Database.* MySQL-klusteri.
Noodi	*Node.* Palvelinkomponentti
NoSQL	*Not only SQL.* Käsite, jolla kuvataan perinteisestä relaatiomallista poikkeavia tietokantoja.
ORM	*Object-relational mapping.* Oliomallin mukaisen esityksen kuvaus relaatiomallin mukaiseksi esitykseksi.
OSI	*Open Source Iniative.* Järjestö, joka on määritellyt avoimen lähdekoodin tunnuspiirteet.
PITR	*Point In Time Recovery.* Ajankohtaan elpyminen.
QoS	*Quality of Service.* Tietoliikenteen luokittelu ja priorisointi.
RAID	*Redundant Array of Independent Disks.* Vikasietoinen laitteistoalijärjestelmä.
RHCS	*Red Hat Cluster Suite.* Ohjelmistokokonaisuus korkeaan käytettävyyteen ja kuormantasaukseen.
RHEL	*Red Hat Enterprise Linux.* Red Hatin Linux -jakeluiden tuote.
SAN	*Storage Area Network.* Kuitulevyjärjestelmä.
SLA	*Service Level Agreement.* Palvelutasosopimus.
SLO	*Service Level Objective.* Palvelutason tavoitteet.

SQL	*Structured Query Language.* IBM:n kehittämä standardoitu kyselykieli.
SR	*Streaming Replication.* Streaming-replikointi.
SPOF	*Single Point Of Failure.* Yksittäinen vikaantumispiste.
TIFF	*Tagged Image File Format.* Kuvien tallennukseen käytetty häviötön tiedostomuoto.
TKHJ	*Tietokannan hallintajärjestelmä.* Ohjelmisto, jonka avulla hallinnoidaan tietokantoja.
VIP	*Virtual IP address.* Virtuaalinen IP-osoite.
WAN	*Wide Area Network.* Laajaverkko.
WSLA	*Web Service Level Agreement.* IBM:n kehittämä XML-pohjainen kehys.
xDSL	*Digital Subscriber Line (DSL) technology.* Modeemireititin.
XML	*Extensible Markup Language.* Merkintäkieli.

1 Johdanto

Insinöörityössä perehdytään avoimen lähdekoodin HA-tietokantoihin. HA on lyhenne sanoista High Availability, joka suomeksi tarkoittaa järjestelmän saatavuutta.

Työn alussa tutustutaan teoriaan ja käsitteisiin, termeihin ja termistöihin, kuten avoin lähdekoodi, korkea käytettävyys, klusterointi sekä palvelutasosopimukset. Lopussa on vuorossa käytännön asiaa.

Avoin lähdekoodi (eng. Open Source) on yhteisnimitys ohjelmille, joiden lähdekoodia jaetaan vapaasti netin kautta ja joita kuka tahansa voi kehittää eteenpäin. Järjestelmän saatavuus (engl. High Availability, HA) on tietojärjestelmien suunnittelussa käytäntö, joka pyrkii siihen, että järjestelmä on aina käyttäjän käytettävissä. Tietotekniikassa termi ryväs (englanniksi cluster) on käytössä monien teknisten ratkaisujen yhteydessä. HA-klusteri tarjoaa mahdollisimman suuren toimivuuden järjestelmälle. Järjestelmien saatavuuden määrä on usein määritelty palvelutasosopimuksissa. HA-järjestelmät ovat yleensä saatavilla yli 99 % ajasta.

Työssä paneudutaan yleisimpiin avoimen lähdekoodin relaatiotietokantoihin ja käydään läpi korkean käytettävyyden vaihtoehdot relaatiotietokannoille (peruskäsitteet, jaettu levyvaranto, laitteiston klusterointi, replikointitavat, kuormantasaus). Lisäksi tutustutaan uusiin NoSQL-tietokantoihin. NoSQL-tuotteita kehittäviä ns. startup-yrityksiä perustetaan nopealla tahdilla. Tämän vuoksi NoSQL-tuotteiksi kutsuttavia ohjelmistoja on erittäin paljon, ja joka käyttöön löytyy kattava kirjaus NoSQL -erikoistuneita tietokantatuotteita.

Työn lopussa pohditaan Big datan haasteita ja tutkitaan pilvipalveluita. Big datan osaajapula tällä hetkellä ja tulevaisuudessa on Suomessa merkittävä haaste. Big datalla tarkoitetaan valtavia ja muodoltaan vaihtelevia datamassoja sekä niiden hyödyntämistä. Hadoop on uusi ja tunnetuin yksittäinen teknologia. Yritykset, jotka tarjoavat pilvipalveluita ovat tarttuneet big datan avaamiin mahdollisuuksiin. Rajaton tallennustila ja laskentakapasiteetti, jotka joustavat täysin tarpeen mukaan, sopivat erinomaisesti ratkaisuiksi moniin big datan esittämiin haasteisiin. Lähteinä olen käyttänyt Internetin lisäksi alan kirjallisuutta, lehtiä sekä aiempia tutkimuksia. Suuri osa lähteistä on alun perin englanninkielisiä.

2 Avoin lähdekoodi

Avoimen lähdekoodin ohjelmalla ei ole yhtä standardoitua määritelmää. Yhdysvaltalainen Free Software Foundation (FSF) loi 1980-luvulla termin free software (vapaa ohjelma). Vuonna 1998 yhdysvaltalainen Open Source Initiative (OSI) loi termin open source (avoin lähdekoodi). Yksityisen tai julkisen sektorin näkökulmasta käsitteillä ei ole merkittävää sisällöllistä eroa ja ne molemmat sisältävät samat käyttöön, kopiointiin, muokkaamiseen ja levittämiseen liittyvät vapaudet. Open Source Initiativen määritelmä on yleisimmin käytetty erityisesti yritysmaailmassa ja esimerkiksi JIT 2007 -ehdoissa (JHS 166) viitataan siihen määriteltäessä avointa lähdekoodia. [1.]

Avoin lähdekoodi (eng. Open Source) on yhteisnimitys ohjelmille, joiden lähdekoodia jaetaan vapaasti verkon kautta ja joita kuka tahansa voi kehittää eteenpäin. Usein vapaan lähdekoodin ohjelmien mukana tulee lisenssi, joka velvoittaa luovuttamaan koodiin tehdyt muutokset vastaavalla tavalla muiden vapaaseen käyttöön. Vaikka itse ohjelmakoodi on yleensä ilmaista ja kaikkien saatavilla, yritykset voivat tarjota sen päällä omia maksullisia räätälöinti-, tuki- ym. palveluita. [2, s. 56.]

Hajautettu ohjelmankehitys on tehokasta ja havaitut ohjelmointivirheet korjataan nopeasti, mikä parantaa tietoturvaa. Myös monet kaupalliset laite- tai ohjelmavalmistajat ovat luovuttaneet koodiaan vapaaseen käyttöön. Ne pyrkivät joko hyötymään vapaaehtoisten koodaajien työstä tai lisäämään laitemyyntiä muuttamalla siihen liittyvän ohjelmiston ilmaiseksi. [2, s. 56.]

Avoimen lähdekoodin määritelmä

Open Source Initiativen määritelmän mukaan avoimen lähdekoodin ohjelman tulee täyttää seuraavat vaatimukset:

1. Ohjelman täytyy olla vapaasti levitettävissä ja välitettävissä.
2. Lähdekoodin täytyy tulla ohjelman mukana tai olla vapaasti saatavissa.
3. Myös johdettujen teosten luominen ja levitys pitää sallia.
4. Lisenssi voi rajoittaa muokatun lähdekoodin levittämistä vain siinä tapauksessa, että lisenssi sallii korjaustiedostojen ja niiden lähdekoodin levittämisen. Voidaan myös vaatia, ettei johdettua teosta levitetä samalla nimellä tai versionumerolla kuin lähtöteosta.

5. Yksilöitä tai ihmisryhmiä ei saa asettaa eriarvoiseen asemaan.
6. Käyttötarkoituksia ei saa rajoittaa.
7. Kaikilla ohjelman käsiinsä saaneilla on samat oikeudet.
8. Lisenssi ei saa olla riippuvainen laajemmasta ohjelmistokokonaisuudesta, jonka osana ohjelmaa levitetään, vaan ohjelmaan liittyvät oikeudet säilyvät, vaikka se irrotettaisiin kokonaisuudesta.
9. Lisenssi ei voi asettaa ehtoja muille ohjelmille. Ohjelmaa saa levittää myös yhdessä sellaisten ohjelmien kanssa, joiden lähdekoodi ei ole avointa.
10. Lisenssin sisällön pitää olla riippumaton teknisestä toteutuksesta. Oikeuksiin ei saa liittää varaumia jakelutavan tai käyttöliittymän varjolla. [1.]

2.1 Avoimen lähdekoodin lisenssit

Open Source -ohjelmia jaetaan useilla eri lisensseillä, joten tarkkoja yhtenäisiä pelisääntöjä ei ole olemassa. Open Source -lisensseihin luetaan mm. Apache-, BSD-, GPL-, LGPL-, MIT-, Eclipse Public ja Mozilla Public lisenssit. [3.]

Sallivat lisenssit

MIT-, BSD- ja Apache-lisenssit ovat sallivia. Lisensoitua ohjelmaa saa muokata ja levittää vapaasti ja lähdekoodin voi, mutta ei välttämättä tarvitse, laittaa levitettävän ohjelman mukaan. Nämä lisenssit ovat hyvin lyhyitä ja selviä. Esimerkiksi MIT-lisenssi mahtuu kolmeen kappaleeseen. Tunnetuimmat tähän kategoriaan kuuluvia lisenssejä käyttävät ohjelmistot ovat BSD-käyttöjärjestelmä, PostgreSQL-tietokanta ja Apache www-palvelin. [4.]

Vastavuoroisuutta edellyttävät lisenssit

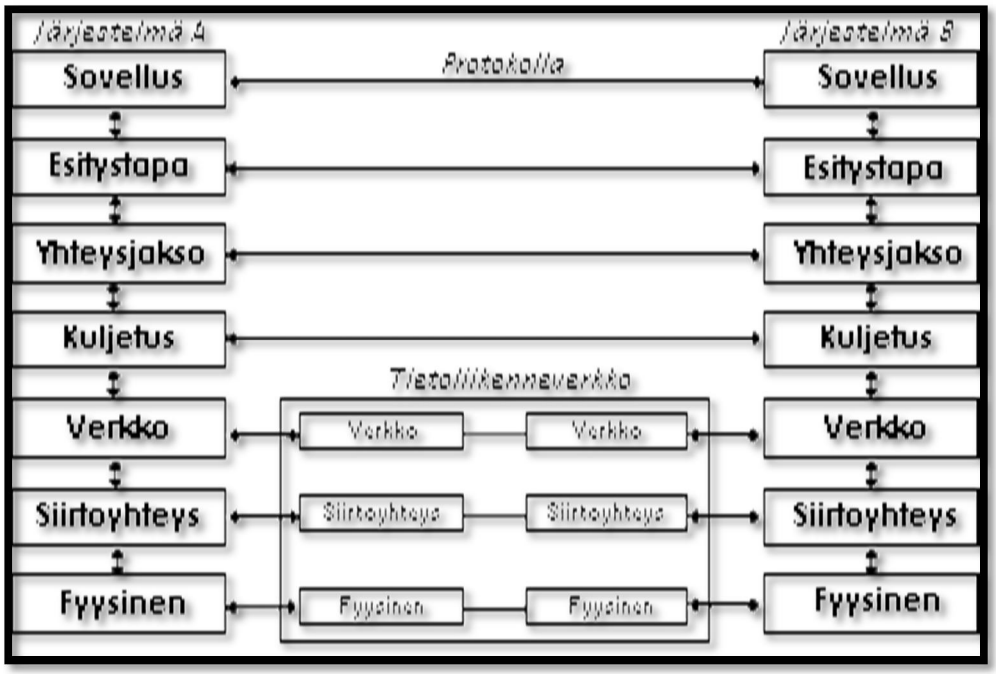
Eclipse Public License, Mozilla Public License ja Lesser General Public License (LGPL.) kuuluvat vastavuoroisuutta edellyttäviin lisensseihin. Näissä lisensseissä edellytetään, että ohjelmatiedostoihin tehdyt muutokset tulee julkaista “alavirtaan” tietyin edellytyksin, joista yleisin on ohjelmiston julkinen levittäminen muutoksen jälkeen. Näillä lisensseillä julkaistuja ohjelmistoja saa kuitenkin yhdistää vapaasti muihin, eri lisensseillä tehtyihin ohjelmistoihin esimerkiksi linkittämällä kirjastoihin. [4.]

Vahvaa vastavuoroisuutta edellyttävät lisenssit

Kyseessä on vastavuoroisuutta edellyttävien lisenssien erityiskategoria, sillä nämä lisenssit edellyttävät lähdekoodin julkaisemista alkuperäisellä lisenssillä kaikissa niissä tilanteissa, joissa alkuperäistä ohjelmistoa muokataan tai siihen yhdistetään uusia elementtejä esimerkiksi linkittämällä. Tämän hetken kaikista suosituin avoin lisenssi, GNU General Public License (GPL) kuuluu tähän kategoriaan. Näiden lisenssien kohdalla ongelmana on yhteensopivuus muiden avoimen lähdekoodin lisenssien kanssa, mikä estää komponenttien jakamista eri projektien välillä. [4.]

2.2 OSI-malli

OSI-malli muodostuu seitsemästä kerroksesta, joiden varaan tiedon välitys muodostuu. Kerrostuksen idea on, että malli toimii pyramidin tavoin: ylempi kerros käyttää hyväkseen alempia kerroksia, jotka taas ovat toiminnaltaan "alkeellisempia". Kuvassa 1 esitellään OSI-mallin eri kerrokset. [5.]



Kuva 1. OSI-malli [5.]

- Alin kerros on fyysinen kerros. Siihen kuuluvat kaikki tiedonsiirtoon liittyvät loogiset, sähköiset sekä mekaaniset asiat. Tietoa voidaan siirtää sekä sarja- että rinnakkaismuotoisesti: Sarjamuotoisessa tiedonsiirrossa siirretään bitit yksi kerrallaan peräkkäin. Etuna on se, että siirtojohtimia ei tarvita välttämättä kuin kaksi. Koska bitit siirretään peräkkäin, lähetettävien bittien ja tavujen alku sekä loppu tulee merkitä jollain tavalla, jotta peräkkäiset bitit erottuvat toisistaan. Rinnakkaismuotoisessa tiedonsiirrossa siirretään yhden merkin kaikki bitit samaan aikaan, omaa johdintaan pitkin. Tiedonsiirto on nopeampaa kuin sarjamuotoisessa siirrossa. Peräkkäisten merkkien erottamiseen käytetään usein liipaisujohdinta, jonka signaali ilmoittaa uuden merkin alkamisen. Rinnakkainen tiedonsiirto ei ole kannattavaa pitkillä matkoilla eikä langattomissa yhteyksissä. Sitä käytetään etupäässä tietokoneen sisällä ja lähellä olevien oheislaitteiden välillä.

- Toinen kerros, eli siirtoyhteyskerros luo yhteyden, korjaa virheet sekä purkaa yhteyden. Yhteyden luominen ja purku toteutuu fyysisestä kerroksesta riippuen, esimerkiksi puhelinyhteyden luominen. Pakettivälitteisessä yhteydessä tätä ei ole. Lisäksi siirtoyhteyskerros pitää huolen vuonohjauksesta, joka tarkoittaa ettei tietoa lähetetä nopeammin kuin vastaanottaja pystyy sen käsittelemään. Siirtoyhteyskerros huolehtii myös siitä, että läpi kulkeva tieto on virheetöntä. Virheettömyyden varmistaminen tapahtuu yleensä käyttämällä koodeja, jotka havaitsevat virheen ja lähettävät virheellisen tiedon uudelleen.

- Verkkokerros tarjoaa verkon rakenteesta riippumattoman tiedonsiirron. Verkkokerroksen tarkoitus on piilottaa tiedonsiirron fyysiseen toteutukseen liittyvät piirteet. Samanlainen tietokoneverkko voidaan rakentaa eri tavalla. Verkkokerroksen tehtävänä on valita, mitä kautta sanomat monihaaraisessa tietokoneverkossa lähetetään.

- Kuljetuskerros takaa muun muassa luotettavan päästä päähän -yhteyden tietokoneverkossa. Joskus saattaa syntyä katkos vaikkapa tietokonevian vuoksi. Siinä tapauksessa kuljetuskerros huolehtii siitä, että tietoliikenne ei katkea vaan käytetään vaihtoehtoista reittiä.

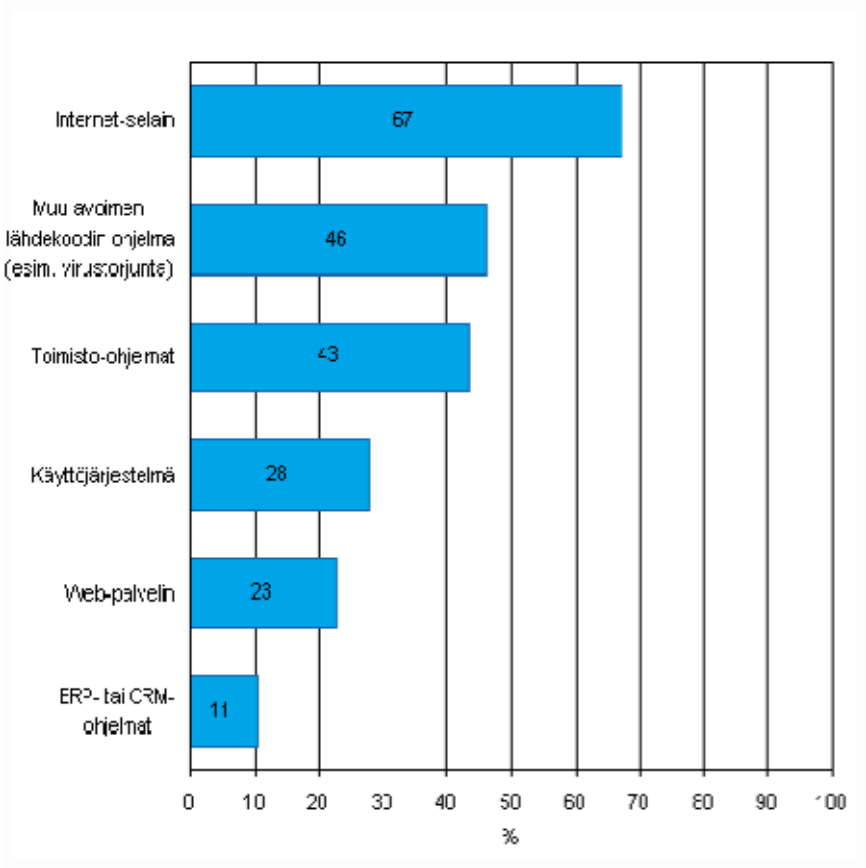
- Yhteysjaksokerros pitää huolen, ettei tiedonsiirto sekoa esimerkiksi fyysisten yhteyksien katketessa. Myös tiedon salaaminen kuuluu yhteysjaksokerroksen tehtäviin.

- Esitystapakerros hoitaa tiedon esitysmuodon oikeaksi tiedonsiirron yhteydessä. Esitystapakerros päättää, missä muodossa kokonaisluvut, teksti, kuva tai ääni esitetään tiedonsiirron yhteydessä. Tarkoituksena on, että lähetetty tieto on muodossa, jonka myös vastaanottaja ymmärtää.

- Ylimpänä OSI-mallissa on sovelluskerros. Sen tehtävänä on toimia linkkinä ohjelmaan, joka tiedonsiirtoa tarvitsee. [5.]

2.3 Tilastotietoa avoimen lähdekoodin ohjelmista

Tilastokeskuksen mukaan avoimen lähdekoodin ohjelmia on käytössä hyvin yleisesti vähintään kymmenen henkilöä työllistävissä yrityksissä. Ainakin jokin avoimen lähdekoodin ohjelmisto on käytössä 79 prosentilla yrityksistä. Tiedot on kerätty Tilastokeskuksen kyselytutkimuksella keväällä 2011, ja ne koskevat vähintään kymmenen henkilöä työllistäviä yrityksiä. Kuvassa 2 on esitelty avoimen lähdekoodin ohjelman käyttötietoa yrityksissä. [6.]



Kuva 2. Avoimen lähdekoodin ohjelmien käyttö keväällä 2011.[6.]

Tavallisin avoimen lähdekoodin ohjelma yrityksissä on Internet-selain (67 %). Avoimen lähdekoodin selain on jonkin verran harvemmin käytössä isoissa, vähintään 100 henkilöä työllistävissä yrityksissä (56 %) kuin pienemmissä kokoluokissa (66-69 %). Toimialoittain yleisyys on selvästi yleisintä informaation ja viestinnän toimialalla (85 %). [6.]

Avoimen lähdekoodin toimisto-ohjelma on käytössä 43 prosentissa yrityksistä. Pienimmässä kokoluokassa se on käytössä 49 prosentilla yrityksistä ja suurimmassa kokoluokassa 23 prosentilla. [6.]

Myös avoimen lähdekoodin käyttöjärjestelmä on varsin yleinen ja on käytössä 28 prosentilla yrityksistä. Yleisemmin sellainen on käytössä isoissa vähintään 100 henkilöä työllistävissä yrityksissä (41 %) kuin pienemmissä kokoluokissa joissa sen yleisyys vaihtelee 26:sta 30 prosenttiin. Toimialoittain yleisyys on selvästi yleisintä informaation ja viestinnän toimialalla (58 %). [6.]

3 Järjestelmän saatavuus

Järjestelmän saatavuus (engl. High availability, HA) on tietojärjestelmien suunnittelussa käytettävä käytäntö, joka pyrkii siihen että järjestelmä on aina käyttäjän käytettävissä.

Järjestelmän saatavuus tarkoittaa, että järjestelmän käyttäjät voivat käyttää järjestelmää normaaleihin tehtäviinsä, esimerkiksi pääsevät lukemaan ja päivittämään siihen tallennettuja tietoja. Järjestelmä voi olla käyttämättömissä joko suunnitellun tai suunnittelemattoman katkon vuoksi. HA-suunnittelun avulla pyritään minimoimaan suunnittelemattomien katkojen olemassaolo ja jos niitä ilmaantuu, pyritään pitämään niiden kesto mahdollisimman lyhyenä. [7.]

Vikasietoisuudella tarkoitetaan palvelujen jatkumista taukoamatta vikatilanteesta huolimatta, esimerkiksi kahdentamalla kriittiset komponentit ja osoittamalla laitteiden ja käyttöjärjestelmän palvelut niin, että sovellusten toiminta jatkuisi häiriintymättä yhden komponentin viasta huolimatta. Tyypillisesti vikasietoinen laitteistoalijärjestelmä on raid-levy-yksikkö (Redundant Array of Independent Disks). RAID on tekniikka, jolla tietokoneiden vikasietoisuutta ja/tai nopeutta kasvatetaan käyttämällä useita erillisiä kiintolevyjä, jotka yhdistetään yhdeksi loogiseksi levyksi. RAID-tekniikkaa käytetään etenkin siellä, missä levyjen vasteajat tai virheettömyys ovat tärkeitä, kuten levy- ja tietokantapalvelimissa. Koko palvelinjärjestelmä on vikasietoinen vasta silloin, kun sen kaikki komponentit verkkosovittimia myöten ovat kahdennettuja ja varusohjelmisto takaa, että yksittäisen komponentin vikaantuminen ei keskeyttäisi käyttäjien sovellusistuntoja. Järjestelmän saatavuudella tarkoitetaan, että järjestelmän suunniteltujen ja suunnittelemattomien katkoksien kokonaiskesto tietyllä aikavälillä kyetään pitämään annetun raja-arvon ulkopuolella. Esimerkiksi 99,9 prosentin käytettävyys merkitsee vuodessa korkeintaan 8 tunnin 45 minuutin yhteenlaskettua käyttökatkosta. Järjestelmän saatavuus merkitsee sovellustasolla matalampaa palvelutasoa kuin vikasietoisuus. Vikasietoisena sovellusistunnot säilyvät laitteiston häiriöstä huolimatta. Korkeaan käytettävyyteen riittää, että laitevioista aiheutuvat katkokset saadaan pidettyä riittävän lyhyinä ja harvinaisina, vaikka palvelun katkeaminen näkyisi myös käyttäjille asti. Käyttäjä saattaa joutua esimerkiksi käynnistämään katkenneen tiedonsiirron uudestaan. [7.]

Vikasietoisellakin laitteistolla järjestelmän käytettävyys saattaa olla matala, jos esimerkiksi varmistusten ottaminen vaatii sovellusten ajamisen alas yöllä tai viikonloppuisin. Suunnitellutkin käyttökatkot ovat käyttökatkoja. [7.]

HA-järjestelmä jatkaa toimintaansa vaikka mikä tahansa yksittäinen laite järjestelmästä hajoaisi. Järjestelmä voi kestää myös useamman komponentin tai laitteen hajoamisen. Laitteistot ja niiden tärkeimmät komponentit pyritään vähintään kahdentamaan, jotta yhden komponentin hajoaminen ei vaikuta toimintaan tai aiheuttaa mahdollisimman lyhyen ja automaattisesti korjautuvan katkoksen eikä hävitä dataa tai minimoi vahingoittuneen datan määrän. Kahdentaminen tai monentaminen vaatii tuen laitteilta, käyttöjärjestelmiltä ja sovelluksilta toimiakseen. Laitteita ei siis tavallisesti voi vain monistaa ilman erikoisjärjestelyitä vaan se vaatii tuen järjestelmiltä. [7.]

4 Klusterointi

Tietotekniikassa termi ryväs (englanniksi cluster) on käytössä monien teknisten ratkaisujen yhteydessä. Perusmerkityksessään ryväs tarkoittaa joukkoa toisiinsa kytkettyjä tietokoneita, jotka näkyvät käyttäjälle yhtenä järjestelmänä. Yleensä klusterilla pyritään parantamaan joko palvelimen käytettävyyttä tai tehokkuutta. [8.]

Suureen käsittelytehoon taas päästään niin, että palvelinprosessit jaetaan useamman koneen kesken. Kaksi konetta ehtii enemmän kuin yksi, eli kyseessä on kuorman jako. ”Load-balancing” klusteri jakaa sovelluksien aiheuttamaa kuormaa usealle palvelimelle. Samalla saavutetaan vikasietoisuutta. [8.]

Teknisesti korkea käytettävyys ja suuri käsittelyteho ovat erisuuntaisia tavoitteita, vaikka ne useimmiten tukevatkin toisiaan. Ryvästekniikoilla, eli klustereilla keskitytään usein tavoittelemaan ensisijaisesti jompaakumpaa. Varsinkin yrityskäytössä korkea käytettävyys on yleinen syy hankkia rypäitä, kun taas korkeakouluissa ja tutkimuslaitoksissa suuri laskentateho on useimmiten hankintoja ohjaava tekijä. Teknisissä ratkaisuissa riittää valinnan varaa tavoitteen määrittelyn jälkeenkin. Lähtökohtana on palvelimissa ajettavat sovellukset kuten tietokantaohjelmistot, sillä paras hyöty saadaan kun sovellukset käyttävät klusterin ominaisuuksia hyväkseen. Kaikkia sovelluksia tukevia yleisklustereita ei ole olemassa. [8.]

High Availability -klusteri (HA) tarjoaa mahdollisimman suuren toimivuuden järjestelmälle. Korkeaan käytettävyyteen päästään siten, että rypään tietokoneet voivat toimia toistensa varakoneina. HA-klusteri on suunniteltu kestämään yhdessä palvelimessa tapahtunut häiriö. Häiriön sattuessa sovelluspalvelut siirretään toiselle, samassa klusterissa toimivalle palvelimelle. Tavallisesti siirron aikana tapahtuu lyhyt katkos (muutamista sekunneista muutamiin minuutteihin). Kriittiset tietokannat ovat yleisesti HA-klusteroituja. Järjestelmä suunnitellaan siten, että siinä ei ole Single Point Of Failure kohtia lainkaan. [9.]

Huolellinen suunnittelu ja testaus on tarpeen, jotta HA-ratkaisu toimii oikein ja heikoilta lenkeiltä vältytään (SPOF, Single Point Of Failure). Yksittäinen vika (SPOF) saattaa lopettaa koko järjestelmän toiminnan. Verkossa jossa tavoitteena on korkea käytettävyys, luotettavuus sekä saatavuus, SPOF ei ole toivottavaa. [9.]

Arvioimalla mahdolliset yksittäiset viat pyritään löytämään kriittiset komponentit monimutkaisesta järjestelmästä, joka saattaisi aiheuttaa vian toimintahäiriön sattuessa. Yksittäinen osa/vika ei saisi olla mahdollista luotettavissa järjestelmissä. [9.]

Kuormantasausklusterissa liikenne tasataan useammalle palvelimelle yhden sijaan. Tilanteissa, joissa palveluun kohdistuu suuri määrä liikennettä samanaikaisesti, saavutetaan parempi toimivuus. [10.]

Laskennallisessa klusterissa yhdistetään useita laitteita suorittamaan suurempaa laskutoimitusta. Hajauttamalla laskenta voidaan saada riittävä määrä laskentatehoa käyttöön ilman kalliita yksittäisinvestointeja. Usein on edullisempaa kasvattaa tehoa lisäämällä klusteriin peruskoneita kuin ostaa yksi supernopea- ja kallis laitteisto. [10.]

Grid Computing on ns. distributed computing -malli, joka tarkoittaa että laitteisto voi sijaita useissa eri paikoissa ja jopa eri valtioiden ja organisaatioiden alueilla. Sitä voidaan hyödyntää sekä palvelinpuolella että työasemapuolella. Mallin mukaan suunniteltu järjestelmä voidaan suunnitella myös toteuttamaan ratkaisu useaan eri tarpeeseen, jolloin se koostuu useammista komponenteista, jotka hoitavat oman osaamisalueensa tehtävät. [10.]

Klusterointi on monentamisen erikoisratkaisu, joka tähtää taulukossa 1 esiteltyihin ominaisuuksiin:

Taulukko 1. Klusterityypit [10.]

Klusteri	Tarkoitus	Ympäristö
HA-klusteri	Korkea saatavuus, katkokset minimoitu	Palvelin
Kuormantasausklusteri	Suuri suorituskyky, resurssien tasainen käyttö	Palvelin
Laskennallinen klusteri	Suuri laskentateho, matalat kustannukset	Palvelin, työasema
Grid Computing	Suuri suorituskyky, resurssien hajauttaminen	Palvelin, työasema

HA-klusterin huono puoli on se, että käytössä olevat varapalvelimet ovat yleensä käyttämättöminä normaalitilanteissa, joten ne eivät tuota mitään liiketoiminnan kannalta. HA-klusterit ovat kuitenkin välttämättömiä, sillä ne tuottavat palvelun, mikäli ensisijaiset palvelimet rikkoutuvat. Mikäli käytössä on myös kuormantasausklusteri, saadaan varapalvelimet hyötykäyttöön myös normaalitilanteissa. Näin ollen nekin tuottavat sekä vähentävät yksittäisen palvelimen kuormaa antaen klusterille paremman suorituskyvyn. Kuormantasauksen ansiosta palvelut näkyvät yhtenä suurena järjestelmänä käyttäjälle, joten järjestelmä täyttää ilmenemisensä ja toimintansa vuoksi Grid Computingin tunnusmerkit. Mikäli tarvitaan suurta laskentatehoa, siihen sopii laskennallinen klusteri. Hajauttamalla laskenta suuremmalle määrälle edullisempia

laitteita, saadaan riittävä määrä laskentatehoa käyttöön ilman kalliita yksittäisinvestointeja. [10.]

5 Järjestelmän saatavuuden takaaminen

Järjestelmien saatavuuden määrä on usein määritelty palvelutasosopimuksissa. Yleensä saatavuus ilmoitetaan prosentteina ajasta; esimerkiksi 90 %:n saatavuus tarkoittaa korkeintaan 16 tunnin 48 minuutin yhteenlaskettua katkosta viikossa. Suunniteltuja katkoksia ei aina oteta huomioon prosentteja arvioitaessa. HA -järjestelmät ovat yleensä saatavilla yli 99 % ajasta, ja usein niitä mainostetaan "yhdeksikköjen määrällä"; monet HA -järjestelmät ovat esimerkiksi ns. viiden yhdeksikön-järjestelmiä eli saatavilla 99,999 % ajasta (korkeintaan 5 minuuttia 16 sekuntia katkosaikaa vuoden aikana). [11.]

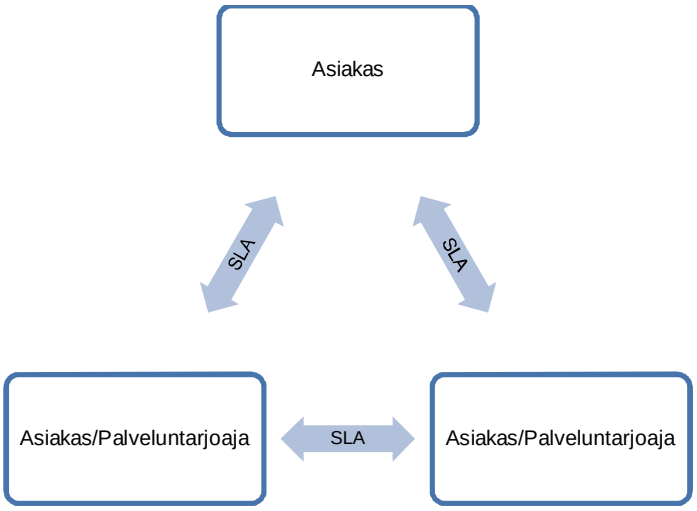
Nykyiset yritysten liiketoimintamallit vaativat datakeskuspalveluiden korkeaa käytettävyyttä. Tämä asettaa vaatimuksia sekä datakeskusten sisäisen arkkitehtuurin että datakeskusten välisten yhteyksien suunnitteluun. Datakeskuspalveluiden tulee olla aina saatavilla. Käyttökatkoilla voi olla merkittäviä taloudellisia vaikutuksia yrityksen toimintaan tai ne voivat pahimmassa tapauksessa aiheuttaa koko liiketoiminnan lopettamiseen. [12.]

5.1 Palvelutasosopimukset

Palvelutasosopimus (engl. Service Level Agreement, SLA) on asiakkaan ja palveluntarjoajan välinen sopimus, jossa määritellään asiakkaalle tarjotun palvelun palvelutaso. Palvelutasosopimuksella pyritään takamaan tarjotun palvelun laatu. Sopimus tarjoaa asiakkaalle selkeyttä ja realistiset odotukset palvelutasonsuhteen. Palveluntarjoajan on helpompi varata resursseja palvelulle, kun asiakkaan odottama palvelun taso on selkeästi määritelty. Palvelutasosopimuksia määritellään organisaation ulkopuolelta hankittujen palveluiden lisäksi myös organisaation sisäisten yksikköjen toisilleen tarjoamille palveluille. [13.]

SLA sopimuksena on pitkän aikavälin kiinteä sopimus asiakkaan ja palveluntarjoajan kanssa (kuva 3.) Se laaditaan asiakassuhteen muodostumisen yhteydessä ja

sopimuksella on monitahoinen merkitys: markkinoinnillinen, tekninen sekä juridinen. [13.]



Kuva 3. Palvelutasosopimus [13.]

Palvelutasosopimuksessa määritellään tarjottava palvelu, prioriteetit, vastuut ja takuut. Sopimuksessa kuvataan myös luvatun palvelutason alittamisesta seuraavat sanktiot, sekä mittarit, joilla palvelutasoa seurataan. Palvelutasosopimus määrittelee palvelulle esitetyt vaatimukset korkealla abstraktiotasolla, eivätkä ne siten sovellu sinällään palvelutason mittareiksi. Palvelutason tavoitteet eli SLO:t (Service Level Objective) ovat palvelutasosopimuksen pohjalta tehtyjä vaatimusten mittareita, kuten esimerkiksi saatavuus, palvelun nopeus ja vasteaika. Yksittäinen palvelutason tavoite voi riippua useammasta palvelun komponentista, joilla voi olla useita QoS-mittareita (laatutavoite). Vertaamalla palvelutasosopimuksen SLO:ita toteutuneeseen palvelutasoon voidaan varmistua SLA:n toteutumisesta. [13.]

5.2 Verkkopalvelutyypit

Yksityinen

Yksityinen liitäntäpalvelu perustuu asiakkaan tilaajajohdon hallintaan ja sen yhteydessä mahdollisesti tarjottavaan suljettuun palveluverkkoon. Tilaajajohdot on teknisesti toteutettu kiinteinä yhteyksinä (kotikäytössä xDSL-, DOCSIS- tai Ethernet-tekniikalla, yrityskäytössä millä tahansa symmetrisellä tekniikalla) tai langattomina yhteyksinä

(3G/4G, @450, satelliitit). Palveluntarjoajilla on mahdollisuus tarjota differentioituja liitäntäpalveluja, esimerkiksi kapasiteetti, siirtoviive ja pakettihukka. Näiden laatu- ja suorituskykyparametrien takaamista ajasta ja kuormitustasosta riippumatta kutsutaan palvelunlaaduksi: QoS. [14.]

Julkinen

Julkinen liitäntäpalvelu, eli Internet-yhteys perustuu palveluntarjoajan tekemiin asiakas- ja yhteenliitossopimuksiin muiden palveluntarjoajien kanssa. Jos asiakas hyödyntää tätä palvelua, asiakkaalle tarvitaan globaalisti yksilöidyt IP-osoitteet (ei NAT eikä privaattiosoitteita) sekä liitospiste, josta suoritetaan tietovirtojen reititys muihin osoitteisiin. Palvelun maksimaaliset suorituskykyarvot ovat palveluntarjoajan hallittavissa. [14.]

Yhdysliikenne

Pienet palveluntarjoajat ovat asiakkaina suuremmille palveluntarjoajille, eli suuremmat palveluntarjoajat välittävät pienempien tarjoajien liikenteen omana asiakasliikenteenä. Keskenään samansuuruiset palveluntarjoajat toimivat yhdenveroisesti, kumpikin on toiselle asiakas. Sopimus sisältää rajoitteita, joita keskinäisen liikenteen välittämisessä käytetään. [14.]

SLA tarjoaa myös lisäarvopalveluja, joita ovat kaikki verkkoliikenteen tukipalvelut. Ne perustuvat joko normaalin verkkopalvelun lisäksi tarjottuihin virtuaaliverkkoihin, erillisen sovelluspalvelun operointiin, tietoturvapalveluihin, verkkopalvelun integriteetin hallintaan sekä asiakkaan konfiguraation hallintaan. [14.]

Palvelunlaatuparametrien taso määrittää sen, kuinka kallista palvelu on. Mitä tiukemmat kaista- tai viivevaateet ovat, sitä kalliimpi palvelu on. Sama pätee myös käytettävyyteen. Esimerkiksi 99,99 %:n käytettävyys viikon mittausajan yli tarkoittaa palvelua, joka sisältää vian noin minuutin verran virhetilanteita viikossa. Se ei ole mahdollista tilaajajohdolla olevien laitteiden huoltoa ilman erillisjärjestelyjä. 99 %:n käytettävyys mahdollistaa normaalit huoltotoimenpiteet ilman sopimusrikkomuksia (noin puolitoista tuntia/viikko.) [14.]

Palvelutasosopimuksen määrittelykielet ja monitorointi

SLA:n määrittelyyn ja monitorointiin on tarjolla useita valmiita kieliä, joista suurin osa on XML-pohjaisia. Näiden kielien kypsyys ja määrittelytaso (ja täten soveltuvuus eri tilanteisiin) vaihtelee suuresti. Palvelutasosopimuksen määrittely XML-pohjaisella kielellä tarjoaa mahdollisuuden määrittelyn uudelleenkäyttöön uusissa samantyyppisten palveluiden SLA-sopimuksissa. [13.]

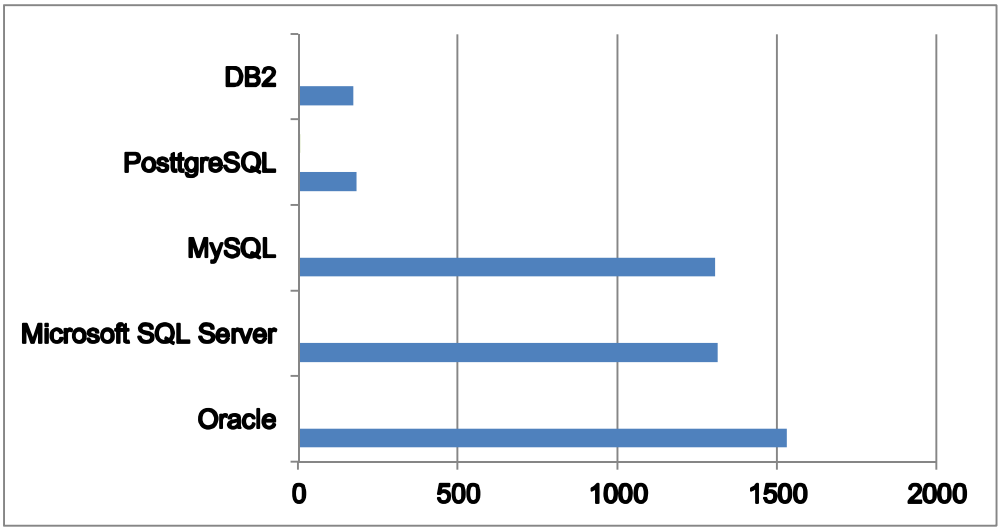
WSLA

WSLA on IBM:n kehittämä XML-pohjainen kehys Web Service -rajapinnoilla toteutettujen palveluiden palvelutasosopimusten määrittelyyn ja monitorointiin. WSLA-kehys sisältää monitorointiympäristön, jolla palvelua voidaan monitoroida dynaamisesti. Monitorointiympäristö osaa muuntaa SLA:n automaattisesti konfiguraatioparametreiksi. WSLA:lla voidaan myös määritellä toimenpiteitä, jotka tulee suorittaa kun luvattu palvelutaso alittuu. WSLA on suunniteltu mahdollistamaan myös kolmannen osapuolen suorittamaa monitorointia. [13.]

6 Avoimen lähdekoodin relaatiotietokannat

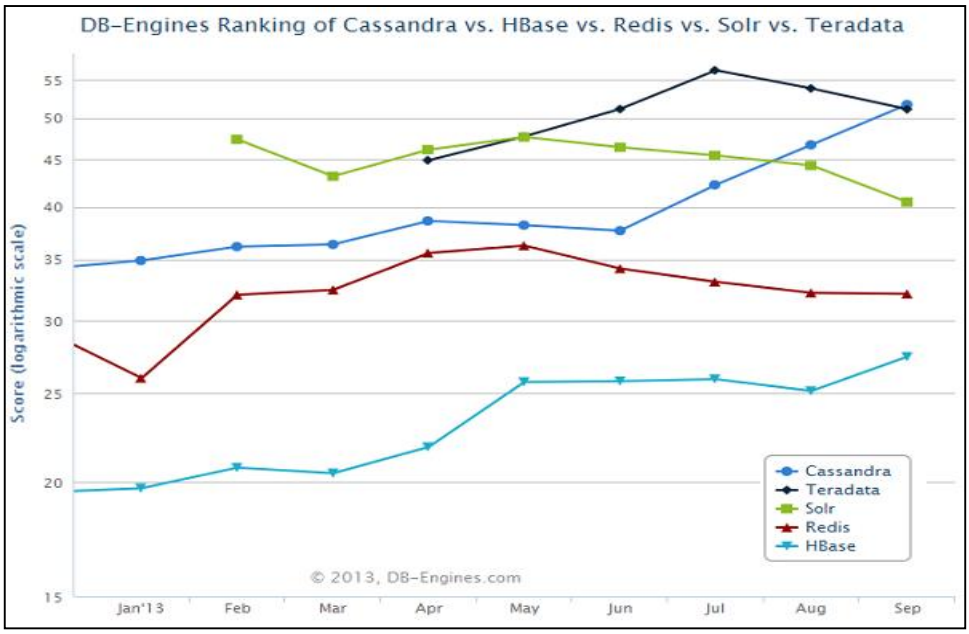
Insinöörityössä keskitytään yleisimpiin avoimen lähdekoodin relaatiotietokantoihin ja uusiin NoSQL-kantoihin sekä tutustutaan Big dataan ilmiönä ja tekniikkana.

Kuvassa 4 on esitelty avoimen lähdekoodin tietokantojen käyttöaste verrattuna Oraclen, Microsoftin ja IBM:n kaupallisiin tietokantoihin. Suosituimmat tietokannat syyskuussa 2013 olivat Oracle (1530), Microsoft SQL Server (1314), MySQL (1306), PostgreSQL (182), DB2 (172), ja lisäksi myös MongoDB (152), joka ei näy kuvassa. [15.]



Kuva 4. Suosituimmat tietokantojen hallintajärjestelmät syyskuussa 2013 [15.]

Avoimen lähdekoodin hallintajärjestelmiä ovat muun muassa Cassandra, HBase, Redis, Solr ja Teradata. Kuvassa 5 on verrattu näiden järjestelmien ominaisuuksia ja seurataan Bigdatan ja NoSQL-tuotteiden nousevaa kehityskaarta.



Kuva 5. Avoimen lähdekoodin hallintajärjestelmät, syyskuu 2013 [15.]

Cassandra on tehnyt huiman nousun heinäkuusta syyskuuhun, HBase jatkaa matalaa nousuaan ja muut etenevät tasaisesti.

PostgreSQL

PostgreSQL on tehokas, avoimen lähdekoodin relaatiotietokantajärjestelmä. Se on yli 15 vuoden aktiivisen kehittämisen tietokanta-arkkitehtuurin tulos. Se on ansainnut vahvan maineen ja luotettavuuden tietojen eheyden ja oikeellisuuden suhteen.

PostgreSQL toimii lähes kaikissa tärkeimmissä ja yleisimmissä käyttöjärjestelmissä, kuten Linux-, UNIX (AIX, BSD-, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), ja Windows. [16.]

PostgreSQL on täysin ACID-yhteensopiva, se tukee täysin esimerkiksi viiteavaimia, liitoksia, näkymiä, triggereitä, ja tallennettuja menetelmiä (useilla kielillä). Siihen sisältyy useita SQL2008:n tietotyyppejä, kuten INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, ja TIMESTAMP. Se tukee myös binäärisiä suurten objektien varastointia, kuten kuvia, ääniä tai videoita. PostgreSQL-tietokannassa on sisäänrakennetut ohjelmointirajapinnat monille ohjelmointikielille, kuten C / C + +, Java,. Net, Perl, Python, Ruby, TCL, ODBC yms. PostgreSQL-tietokannan dokumentointi on kattava ja laaja. [16.]

PostgreSQL on suuryritysluokan (Enterprise) tietokanta. Se tarjoaa kehittyneitä ominaisuuksia, kuten Multi-version, samanaikaisuuden hallinnan (MVCC), PITR (point in time recovery eli ajankohtaan elpyminen), tablespacet, asynkroninen replikointi, sisäkkäiset transaktiot (savepoints), online-varmuuskopiot, kyselyn suunnittelut ja vikasietoisuuden korjaamisen. PostgreSQL tukee useita kansainvälisiä merkistöjä, multi-byte merkkikoodistoja, Unicode, ja sekä myös locale-aware (sijainti) lajittelua, case sensitiivisen laskentakaavan ja formatoinnin. [16.]

Enterprisedb ja Red Hat (US) ovat merkittävimmät kaupalliset PostgreSQL-tietokantatukijat, ja ne tarjoavat tukipalveluita. Muita sponsoreita ovat mm. 2ndQuadrant (Iso-Britannia), Dalibo (Ranska), Command prompt, Inc. (USA), Redpill Linpro (Ruotsi), NTT Group (Japani) sekä kymmenet muut yhtiöt ympäri maailman. [16.]

MySQL

MySQL on laajasti käytetty avoimeen lähdekoodin perustuva relaatiotietokantaohjelmisto. Alun perin MySQL:ää kehitti ruotsalainen yritys MySQL AB. Sun Microsystems osti yrityksen 16. tammikuuta 2008. Ohjelmistoyritys Oracle Corporation osti Sun Microsystemsin huhtikuussa 2009 noin 7,4 miljardilla dollarilla. Kaupan yhteydessä MySQL:n omistus siirtyi Oraclelle. MySQL on saatavissa vapaalla GNU GPL -lisenssillä tai kaupallisella lisenssillä, mikäli asiakas ei halua käyttää GPL-lisensoitua ohjelmistoa. [17.]

MySQL-tietokannan loi vuonna 1995 suomalainen Michael "Monty" Widenius yhdessä ruotsalaisen David Axmarkin kanssa. MySQL:n ensimmäinen versio julkaistiin 1996. Uusin MySQL-versio on 5.0. MySQL-tietokanta on hyvin suosittu ja laajalti käytetty web-palveluiden tietokantana. MySQL-tietokannan päälle rakennettava ohjelmalogiikka tehdään usein PHP-, Python- tai Perl-ohjelmointikielellä. Sivut julkaistaan Apache-webpalvelimella, joka edelleen toimii Linux-käyttöjärjestelmän päällä. Tätä kutsutaan joskus LAMP-alustaksi. Myös muilla ohjelmointikielillä on mahdollista käyttää MySQL-tietokantaa. [17.]

MySQL sisältää rajapinnan mm. C:lle, C++:lle, C#:lle, Smalltalkille, Javalle, Rubylle ja TCL:lle. MySQL:lle on olemassa MyODBC-niminen ODBC-rajapinta. [17.)

MariaDB

MariaDB:n pääkehittäjä on Michael "Monty" Widenius, joka myös perusti MySQL:n ja Monty Program AB-yhtiön. Hän oli aiemmin myynyt MySQL:n Sun Microsystems:lle yhdellä miljardilla dollarilla, jonka jälkeen hän perusti MariaDB:n. MariaDB on nimetty Wideniuksen nuoremman tyttären Maryn mukaan. Widenius perusti MariaDB:n, sillä hän oli huolestunut Oraclen MySQL:n kehitystahdista. MySQL oli päätynyt Oraclen omistukseen. MariaDB on lähes identtinen MySQL-tietokantaan verrattuna. Erot ovat hyvin pieniä, ja MariaDB:N tuotteet mielletään usein samoiksi tuotteiksi MySQL:n kanssa. MariaDB on viime aikoina kasvattanut suosiotaan merkittävästi. [18, s. 48.]

7 Järjestelmän saatavuuden vaihtoehtoja relaatiotietokannoille

Tietokannan järjestelmän saatavuuden ja vikasietoisuuden pyrkimyksenä on taata, että tietokanta olisi aina käytettävissä eikä yksittäisen laitteistokomponentin vikaantuminen tai konesalitason ongelma aiheuttaisi mittavaa käyttökatkoa.

Tietokannoissa järjestelmän saatavuuteen liittyy useita teknologioita, joista jokainen vastaa tiettyyn tarpeeseen. Teknologiat yhdessä muodostavat vikasietoisen kokonaisuuden.

Suunniteltaessa HA-tietokantajärjestelmiä kannattaa tiedostaa, mihin tarpeeseen eri teknologiat vastaavat. Klusteriteknologia takaa vikasietoisen palvelinkerroksen, mutta levyjärjestelmien tai kokonaisten konesaliympäristöjen vikaantumisen varalta pitää suojautua eri tavoilla. Jos halutaan varmistua tietokantapalveluiden jatkuvasta toiminnasta esimerkiksi tilanteessa, jossa konesalista katoavat sähköt tai lattioille tulee yhtäkkiä kuutioittain vettä, käytännössä tietokanta on kopioitava jollain menetelmällä toisaalle.

Tietokannan jatkuva kopioiminen (yleisesti myös replikointi) toiseen konesaliin tai toiseen kaupunkiin, ehkä jopa toiselle mantereelle on mahdollista myös avoimen lähdekoodin tietokantojen avulla. [19, s. 70.]

Järjestelmän varma saatavuus on erittäin haastavaa ja vaikeaa tietokannoille. Tiedon täytyy olla useammassa paikassa samaan aikaan tiedon muuttuessa jatkuvasti. Järjestelmän saatavuuden avulla tietokannan toiminta voi jatkua ilman keskeytyksiä. Häiriön tai palvelimen rikkoutumista varten ei tarvitse palauttaa varmuuskopioita. Tietokannan loppukäyttäjäosapuoli ei tiedä mitään mahdollisista ongelmista, sillä se toimii aina normaalisti. [20.]

7.1 Varatietokannat

Tietokannan vikasietoisuusteknologian avulla tietokanta voidaan replikoida, eli kopioida haluttaessa jopa useampiin konesaleihin. Tietokantakopioita voidaan kutsuta varatietokannoiksi (eng. Standby database). Varatietokantoja voi olla kolmenlaisia:

Fyysinen varatietokanta (eng. Physical standby database). Varatietokanta virkistyy elvytystilassa, jolloin varatietokanta on tietokannan lohkotasolla identtinen ensisijaisen tietokannan kanssa. Lisäominaisuutena varatietokantaympäristössä varmistukset voidaan ottaa varatietokannassa, jolloin ensisijaista tietokantaa ei tarvitse rasittaa varmistuksilla. Fyysinen varatietokanta voi olla jatkuvasti vain lukuoperaatiot sallivassa tilassa (eng. Readonly mode), jolloin sitä voidaan hyödyntää tehokkaasti raportointiin. Varatietokannan jatkuva raportointimahdollisuus on myös usealla tietokannalla.

Looginen varatietokanta (eng. Logical standby database), virkistyy toistamalla ensisijaisen tietokannan SQL-lauseet varatietokannassa. Sen johdosta varatietokanta voi olla rakenteeltaan erilainen ensisijaiseen tietokantaan verrattuna ja sisältää esimerkiksi raportointia tukevia lisäindeksejä. Looginen varatietokanta ei tue kaikkia tietokannan tietotyyppejä eikä kaikkia DDL-toimintoja.

Tilannevedosvaratietokanta (eng. Snapshot standby database) eroaa edellä mainituista siten, että tilannevedostyyppinen varatietokanta on täysin loppukäyttäjien päivitettävissä.

Tilannevedosvaratietokanta perustetaan muuntamalla fyysinen varatietokanta tilannevedostilaan. Tässä tilassa varatietokanta vastaanottaa ja arkistoi ensisijaisessa tietokannassa syntyneitä tietokantatapahtumien tuottamia tapahtumalokeja, mutta tätä tietoa ei fyysisen varatietokannan tapaan automaattisesti kirjoiteta varatietokantaan. Sen sijaan tilannevedosvaratietokanta virkistetään ensisijaisen tietokannan tapahtumilla vasta siinä vaiheessa kun tilannevedos varatietokanta halutaan muuntaa takaisin fyysiseksi varatietokannaksi.

Tilannevedosvaratietokantaa voidaan käyttää esimerkiksi, kun halutaan testata sovellukseen tehtäviä muutoksia aidolla ensisijaisesta tietokannasta kopioidulla tiedolla tiettyyn ajanhetkeen asti (tilannevedos varatietokannan perustamishetki). Kun testaus on suoritettu, voidaan tilannevedosvaratietokanta taas muuntaa fyysiseksi varatietokannaksi, josta muodostuu taas ensisijaisen tietokannan täydellinen ajantasainen kopio. Tarvittaessa tämä sykli voidaan toistaa ja muuntaa fyysinen varatietokanta tilannevedosvaratietokannaksi niin usein kun halutaan. [19, s. 71–72.]

7.2 Laitteiston klusterointi

DRBD-ohjelmisto

DRBD (eng. Distributed Replicated Block Device) on ohjelmisto, joka mahdollistaa kovalevyjen peilaamisen verkon kanssa ja sitä kautta klusteroinnin. SAN-kuitutekniikkaan perustuvalla kiintolevyllä voidaan toteuttaa vikasietoinen klusteri, toinen vaihtoehto verkkotekniikkaan perustava NAS, mutta sen käyttö on rajoittuneempaa. DRBD on myös laajasti käytetty tekniikka. [21.]

DRBD-tekniikka 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. [21.]

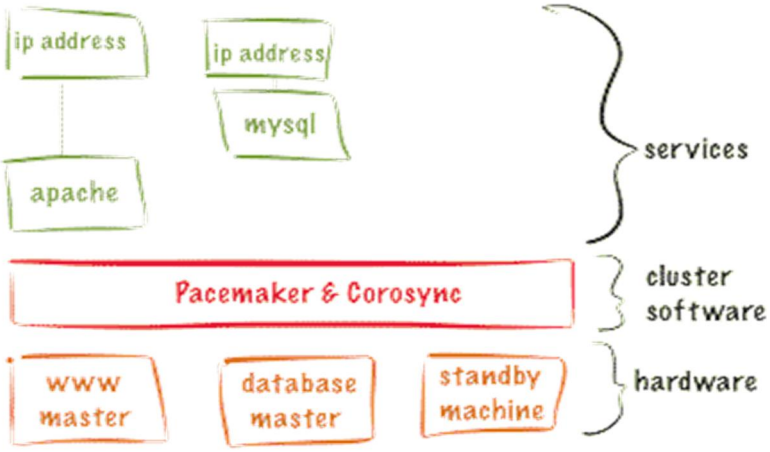
RAID-tekniikkaa hyödyntämällä yhdessä DRBD:n kanssa voidaan välttyä myös osalta ongelmista, jotka kohdistuvat kumpaan tahansa levypalvelimeen. Yksittäisen levyn tuhoutuessa kyseinen levy voidaan vaihtaa ilman, että koko palvelimen toiminta häiriintyy. Palvelimet käytännössä vahtivat toistensa tilaa ja 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 ja toinen on kylmässä valmiustilassa. [21.]

RedHat Cluster Suite -ohjelma

RedHat Cluster Suite (RHCS) on ohjelmistokokonaisuus korkean käytettävyyteen ja kuormantasaukseen. Redhat Cluster Suite sisältää useamman tuotteen. Niitä voidaan käyttää samassa järjestelmässä, vaikka niiden yhteinen käyttö saattaa olla epätodennäköistä. Kaikki tuotteet on aloitettu avoimen lähdekoodin yhteisössä, mutta Red Hat on tuotteistanut sen tuotepaketiksi. Laskennallinen klusterointi ei sisälly RHCS-tuotteeseen vaan Red Hat MRG -tuotteeseen. [22.]

Pacemaker

Pacemakerin kehitys alkoi vuonna 2004 pääosin Red Hatin toimesta. Pacemaker perustuu avoimeen lähdekoodiin. Jos palvelinsolmu (noodi) tai useampi kaatuu klusterissa, niin Pacemaker havaitsee sen ja yrittää automaattisesti käynnistää sen tai jonkin korvaavan sen tilalle. Kuvassa 6 on esitelty Pacemakerin toimintaa. Pacemaker valvoo järjestelmää sekä laitteistojen ja ohjelmistojen vikoja.



Kuva 6. Pacemaker [23.]

Pacemaker yhdistää klusterin resurssienhallinnan (CRM), jonka tehtävänä on käynnistää uudestaan palveluja. Sen avulla on mahdollista pystyttää uudelleen IP-osoitteet, web-palvelimet jne. Pacemaker tukee monia eri käyttötapoja, yksinkertaisin on kahden noodin valmiustilassa oleva klusteri, mutta se myös mahdollistaa 16 noodin active-active-klusterin. Pacemakerilla voidaan vähentää laitteiston määrää ja kuluja. [23.]

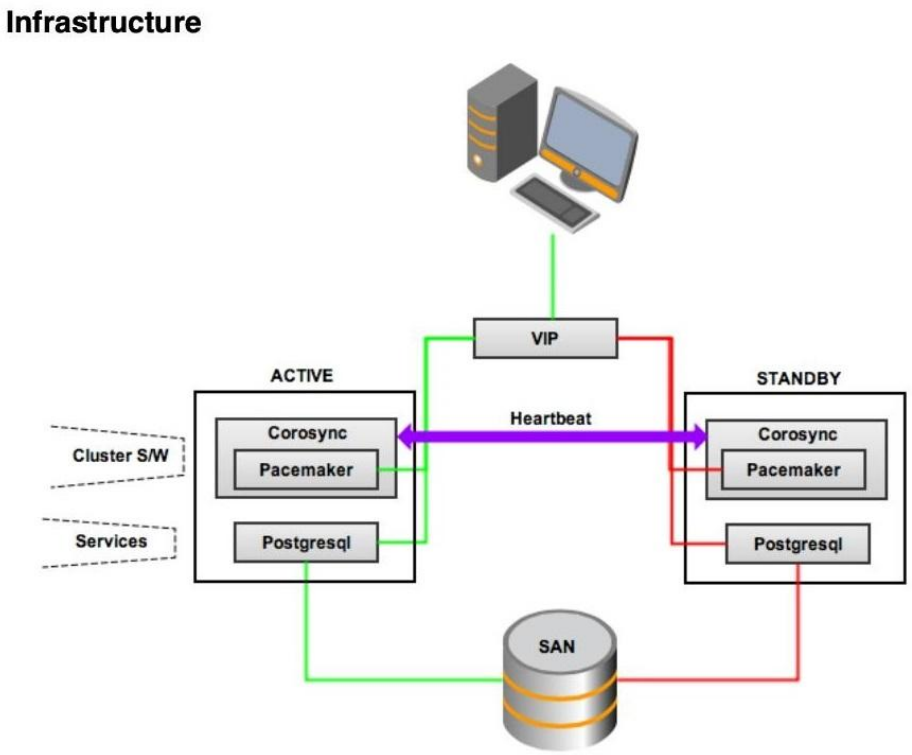
Corosync

Corosync Cluster Engine on avoimen lähdekoodin projekti, joka on BSD-lisenssin alainen. Corosync on johdettu OpenAIS -projektista. Sen tehtävänä on ylläpitää ja kehittää avoimen lähdekoodin sekä kaupallisten klusterihankkeita. Corosync Cluster Engine on ryhmäviestintäjärjestelmä. Se on sisäinen viestien lähettäjä klusterissa, ja sen lisäominaisuuksia käytetään korkean käytettävyyden sovelluksissa. Corosync-

ohjelmisto on suunniteltu toimimaan UDP/IP ja InfiniBand -verkoissa natiivisti eli sisäänrakennetusti. [24.]

Redhat Active-Passive Cluster

Redhat Cluster Suite (RHCS) mahdollistaa luotettavan tietokantaklusterin toteuttamisen. Se on yleiskäyttöinen toteutustapa lähes kaikille tietokannoille. Arkkitehtuuri on yksinkertainen ja perustuu laitteistopohjaiseen klusterointiin. Laitteiston vaatimukset riippuvat tarpeista, mutta käytännössä laitteiden on toimittava Red Hat Enterprise Linuxissa (RHEL) ja SAN-kuitulevyjärjestelmässä sekä RHEL-version on oltava uudempi kuin 5.5. RHCS:ssa suositellaan käytettäväksi GFS-tiedostojärjestelmää sen paremman luotettavuuden vuoksi. GFS (eng. Global File System) eli maailmanlaajuinen tiedostojärjestelmä on toteutettu LVM-tiedostojärjestelmän päälle. LVM (eng. Logical Volume Manager) on Linuxin ytimen loogisten taltioiden hallintakomponentti. [25.]



Kuva 7. Active-Passive Cluster [26.]

Active-Passive Clusterin, joka esitellään kuvassa 7, toteutukseen vaaditaan myös kaksi identtistä palvelinasennusta. Käyttöjärjestelmästä ja tietokannasta on samat versiot

molemmissa. Kahden palvelimen välillä kulkee Heartbeat-sanoma, ja kun noodi1 menee alas niin noodi2 toimii aktiivisena tietyssä ip-osoitteessa. Noodi1:n palautuessa palvelin jatkaa normaalisti. Ip-osoitteen ohjaus toteutetaan virtuaalisella ip-osoitteella eli vip-osoitteella. [25.]

7.3 Replikointi tietokantatasolla

Replikointitapoja on monia, esimerkiksi jaettu levyvaranto, tiedostojärjestelmäreplikointi, transaktiologiin perustuva, trigger -perustainen master standby -replikointi, lausepohjainen, asynkroninen multimaster- ja synkroninen multimaster-replikointi.

Laitteistopohjaista Active/Passive klusteria käytetään useasti tietokannoille, mutta sen suurin haittapuoli on monimutkainen ja vaikeampi laitteiston konfigurointi ja kalliit käyttöönottokustannukset. [27.]

MySQL-replikointi

MySQL:n kaikissa versiossa on jo pitkään ollut vakio-ominaisuutena master-slave-replikointi. Master-slave sanalle ei löydy järkevää suomenkielistä vastinetta, vaan yleisesti käytetään sanaa master-slave. Liitteessä 1 on käytännön esimerkki MySQL-tietokannan master-slave-replikoinnin konfiguroinnista. MySQL-tietokannan oman replikointitoteutuksen lisäksi löytyy monta eri vaihtoehtoista toteutusta, esimerkiksi Tungsten Replicator sekä Perconan ja Oraclen toteutukset. MySQL-replikointi toimii myös WAN-verkossa ja se käyttää TC/IP-tiedostostoprotokollaa. MySQL:n master-slave-replikoinnissa master-noodissa määritellään tietokanta kirjoittamaan binäärilokia ja sen jälkeen luodaan slave-noodille käyttöoikeudet. Tämän jälkeen määritellään slaven noodille masterin noodin tunnus sekä ip-osoite ja binäärilokin kohta, josta replikointi alkaa. Master-slave-replikointi on asynkronista, periaatteessa slavenoodi on pelkästään lukumuodossa. Käytännössä slaveenkin pystyy kirjoittamaan, mutta se ei ole järkevää, koska silloin tieto ei siirry masteriin ja replikointi on epätahdissa. MySQL-replikointi voidaan määritellä Master-Master-muodossa. Tämä konfiguraatio mahdollistaa kummankin tietokantanoodin kirjoittamisen samaan aikaan ja niiden välinen tiedon siirto on synkronista. [28.]

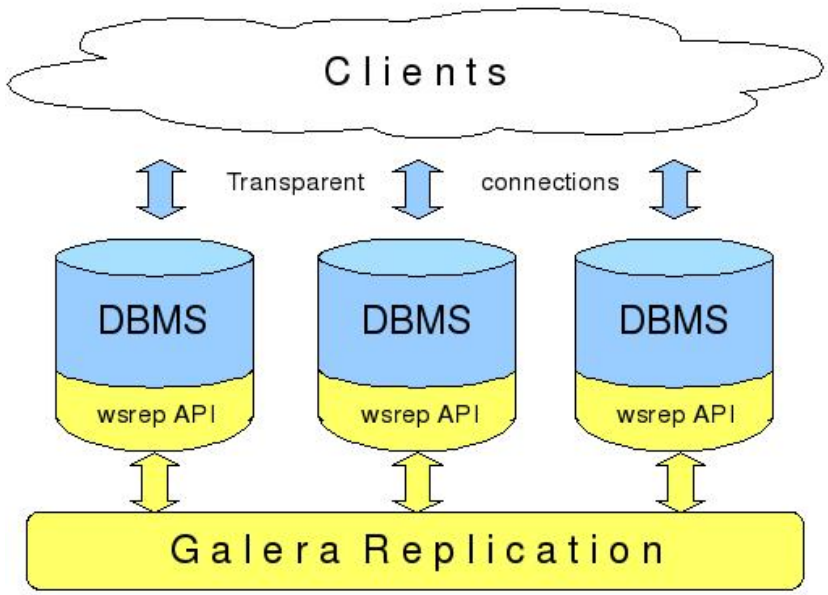
Tungsten-replikointi

Tungsten mahdollistaa epäyhtenäisen tietoliikenteen esimerkiksi MySQL:n, MongoDB:n ja PostgreSQL:n välillä. Tungsten mahdollistaa myös replikoinnin useammalta master noodista yksittäiselle slavelle. [29.]

Tungsten-replikointi korvaa esimerkiksi MySQL:n oman replikoinnin omalla toteutuksellaan. MySQL-tietokanta kirjoittaa binäärilokin ja tungsten-replikointi lukee sitä, käyttäen omaa protokollaa siihen. [30.]

Galera Cluster for MySQL

Galera on MySQL:n sisäinen lisäosa joka korvaa MySQL:n oman replikoinnin. Galera tukee ainoastaan InnoDB-taulumoottoria, MyIsam-taulumoottori on vielä kehitysasteella. Kuvassa 8 on esitelty Galera Cluster. [31.]



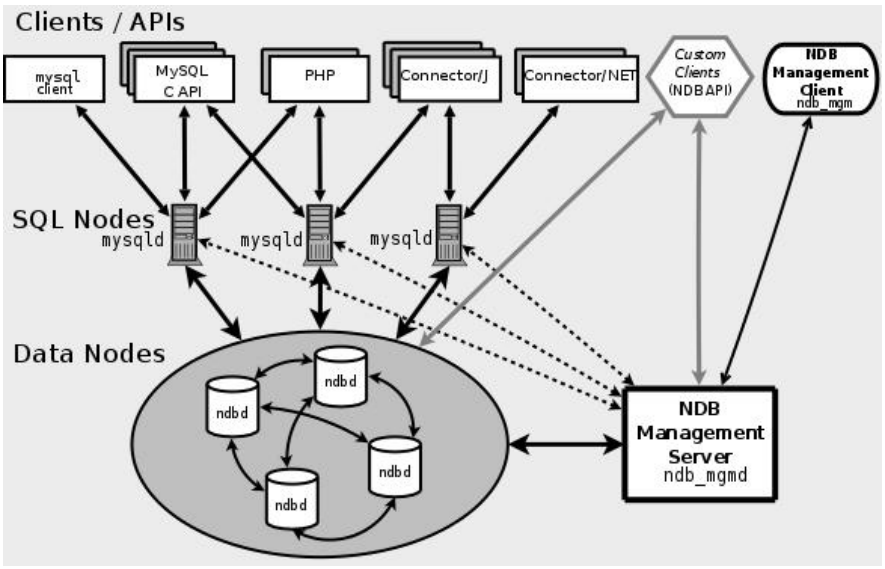
Kuva 8. Galera Cluster for MySQL [31.]

Galera on aito ns. multi-master (active-active-tyyppinen tietokantaklusteri) eli se on täysin synkroninen. Sitä voidaan käyttää myös wan-verkossa. Monisäikeisten slave-noodien käyttötarkoituksilla ei ole rajoituksia. Siinä ei ole slave-viivettä tai muitakaan

integrointiin liittyviä ongelmia. Galeralla on myös automaattinen noodien varaus. Galeran järkevä toteutus vaatii kolme noodia. Galera-noodien edustalle suositellaan kuormantasaajan sijoittamista. Galeralta löytyy oma toteutus, mutta korvaavia avoimen lähdekoodin vaihtoehtoja on useita. [31.] Liitteessä 2 on Galera Cluster konfiguraatio.

MySQL NDB Cluster

MySQL NB Cluster on Oraclen oma tuote MySQL:lle, joka ei toimi MariaDB:n kanssa. MySQL NDB cluster -tuoteperheessä on kolme erilaista solmutyyppiä: SQL, tieto ja hallinta. MySQL-noodi tarjoaa rajapinnan tietoon, tähän on myös olemassa vaihtoehtoinen API-rajapinta. Kuvassa 9 on esitelty MySQL NDB Cluster. [32.]



Kuva 9. MySQL NDB Cluster [32.]

Tietosolmu eli "NDB storage Engine" tarkoittaa, että NDB Clusterissa täytyy käyttää omaa taulumoottoria, esimerkiksi tuotteella InnoDB -taulut eivät ole vikasietoisia ja toimivat erillään klusterista. [32.]

PostgreSQL Streaming replication

Streaming-replikointi (SR) ylläpitää WAL XLOG -kirjaa standby-tietokantapalvelimista ja pitää ne ajan tasalla. Tämä ominaisuus on lisätty PostgreSQL 9.0:sta. Streaming-replikointi mahdollistaa myös hot standby -ominaisuuden. [33.] Liitteessä 3 konfiguroidaan PostgreSQL Streaming.replikointi.

Slony-I

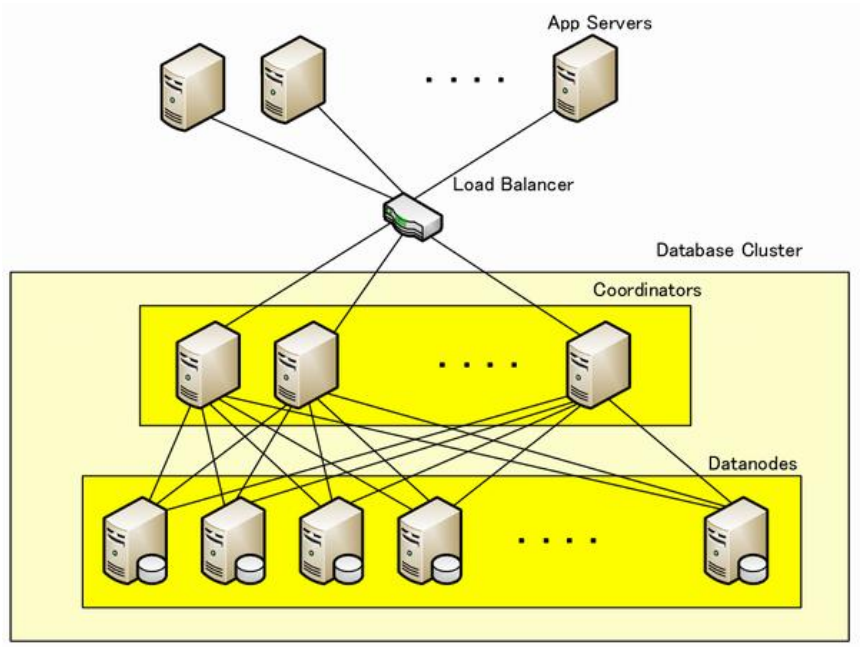
Slony-I on vaihtoehtoinen replikointitoteutus PostgreSQL-tietokannoille. Slony-I soveltuu myös vanhemmille PostgreSQL-versioille, mutta sitä ei enää suositella 9.0-versioille tai uudemmille, koska PostgreSQL:n mukana vakioina tuleva streaming-replikointi soveltuu yleensä paremmin ja se on helpommin ylläpidettävä kuin Slony-I. [34.]

x-DB

EnterpriseDB-yritys on toteuttanut PostgreSQL-tietokannalla oman toteutuksen multimaster-replikoinista. Tuote ei varsinaisesti ole avointa lähdekoodia, mutta se on mainittu siksi, että PostgreSQL-tietokannalle löytyy vaihtoehtoisia multimaster-replikointeja. [35.]

PostgreSQL Multi-Master Replication

Postgres-XC (kuva 10.) on avoimeen lähdekoodiin perustuva ratkaisu PostgreSQL-tietokannan klusterointiin, joka tarjoaa sekä lukuun ja kirjoitukseen skaalautuvuutta.



Kuva 10. Postgres-XC [36.]

Postgres-XC on synkroninen ns. multi-master, eli jokaiseen noodiin pystyy kirjoittamaan samanaikaisesti ja niiden edustalle suositellaan kuormantasaajan sijoittamista. [36.]

PostgreSQL Hot Standby

PostgreSQL-tietokannan Hot Standby -termillä tarkoitetaan, että aktiiviseen tietokantaan tehdään read-only-kyselyitä eikä se häiriinny siitä; tätä voidaan hyödyntää raskaiden raporttiajoihin tuotantoaikana. [37.]

Hot-standby -ominaisuutta voidaan hyödyntää vikasietoisuudessa tai lisäämällä lukukapasiteettia. PostgreSQL-kannalle löytyy monia useita tapoja toteuttaa. Sitä voidaan myös hallita Repmgr-ohjelmistolla. [38.]

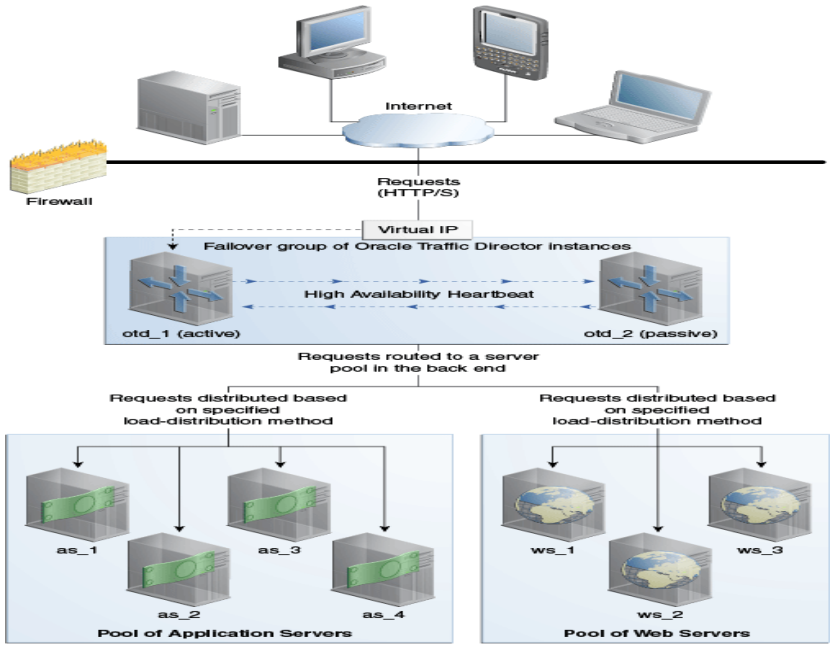
7.4 Kuormantasaus ja yliheitto

Round-robin DNS

Yksinkertaisen kuormantasauksen ja yliheiton voi toteuttaa DNS-nimipalvelimella. DNS-nimipalvelu kääntää ja selvittää IP-osoitteita järjestyksessä listasta, jota permutoidaan (järjestetään) ja vikatilanteessa otetaan häiriintynyt noodi pois listasta. [39.]

VIP

VIP-osoitetta käytetään yleisesti tietokantojen korkeaan käytettävyyteen. Active-Passive-mallissa tietokantaparia ohjataan virtuaalisella IP-osoitteella. Cluster Manager valvoo tietokannan tilaa ja mahdollisen vikaantumisen sattuessa se ohjaa IP-osoitteen aktiiviseen kantapalvelimeen. Kuvassa 11 on esitelty VIP. [40.]



Kuva 11. VIP [40.]

Vikaantuneen palvelimen tullessa takaisin käyttöön palvelin voidaan taas ohjata käyttöön VIP-osoitteella. [40.]

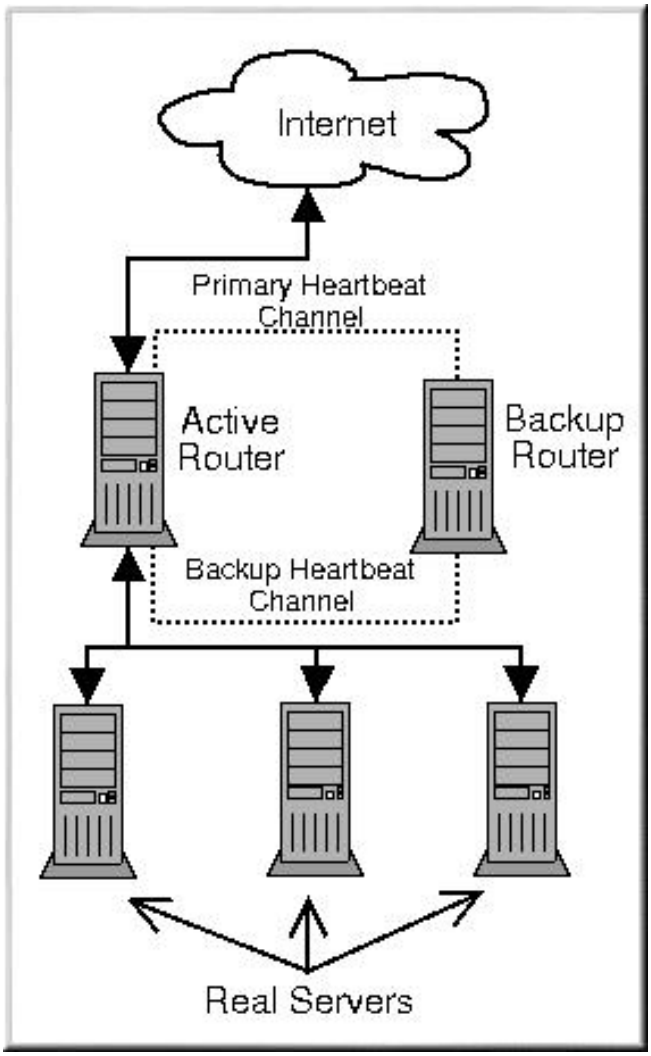
HAProxy

HAProxy toimii avoimen lähdekoodin TCP / HTTP kuormantasaajana, ja se on yleisesti käytetty. Sen tarkoitus on parantaa suorituskykyä www-sivuille hajauttamalla www-palvelimien pyyntöjä useille palvelimille. Sen nimi on lyhenne sanoista High Availability Proxy. Sitä voidaan käyttää myös tietokannoille, mutta siihen käyttötarkoitukseen se on harvinaisempi. [41.]

Piranha

Red Hatin kehittämä Piranha perustuu avoimen lähdekoodin Linux Virtual Server (LVS) -teknologiaan, jota Red Hat on parannellut merkittävästi. Piranha mahdollistaa IP-kuormantasauksen. IP-kuormantausklusterissa yksi palvelin näkyy yhtenä palvelimella ulospäin, mutta todellisuudessa (esimerkiksi www-käyttäjän näkökulmasta) se käyttää isompaa palvelinryhmää. IPLB-klusteri koostuu ainakin kahdesta kerroksesta. Ensimmäinen kerros koostuu kahden samalla tavalla konfiguroidun Red Hat Enterprise Linux AS-Linuxin tai ES-järjestelmien Red Hat Cluster Suite -asennuksilla. Yksi näistä

noodeista toimii aktiivisena IPLB reitittimenä (muut toimivat varmuuskopiona), joka ohjaa Internetistä tulevat pyynnöt ja toinen kerros toimii poolina eli altaana tuleville pyynnöille. [42.]

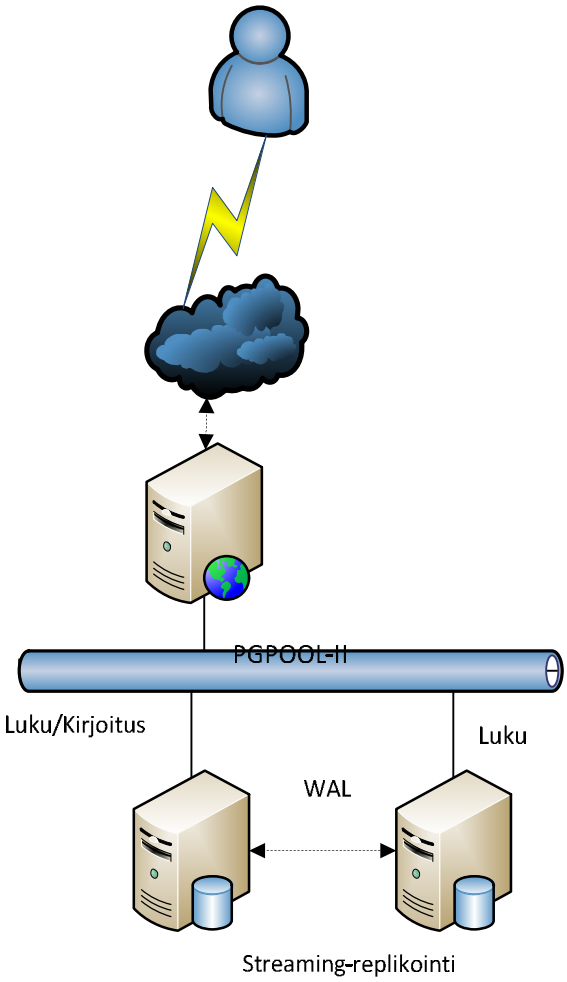


Kuva 12. Piranha [42.]

Palvelimia kutsutaan todellisiksi palvelimiksi (kuva 12.) Todelliset palvelimet tarjoavat kriittisiä palveluja loppukäyttäjille, kun LVS reititin tasapainottaa kuorman näihin palvelimiin. [42.]

PGpool

PGpool (kuva 13.) tarjoaa yhteysaltaan sekä on työkalun replikoinnin hallintaan. Sitä voidaan hyödyntää myös vikasietoisuuden muodostamisessa. [43.]

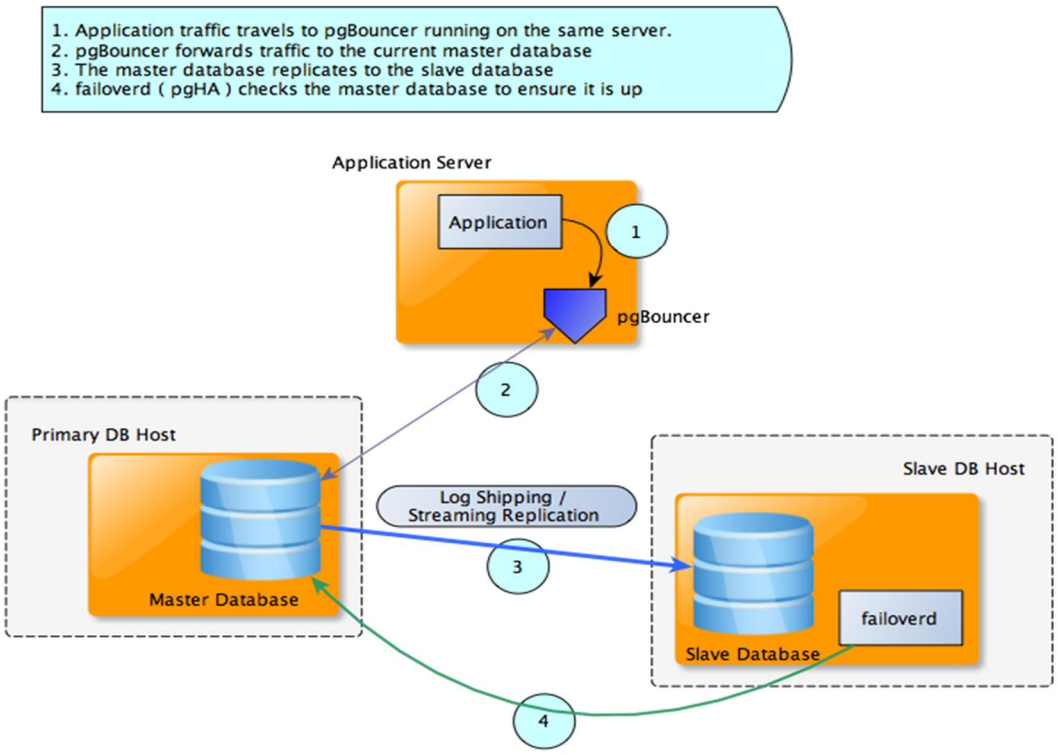


Kuva 13. Yksinkertainen PGpool-ratkaisu [44.]

Replikointitoiminto mahdollistaa reaaliaikainen varmuuskopioinnin luonnin kahteen tai useampaan fyysiseen levyyn, jotta palvelu voi jatkaa pysähtymättä toimintaansa (jos tietokantapalvelimella tapahtuu esim. levyrikko.) [43.]

PGbouncer

PGbouncer on kevyt tietokantayhteysallas (connection pooler) PostgrSQL-tietokannalle. Siinä on kolme tasoa: istunto-, transaktio- ja lausepoolaukset. [45.]



Kuva 14. Tietokantayhteysallas [46.]

PGbounceria voidaan käyttää kevyeen korkeakäytettävyyteen PGpoolin tapaan. [45.]

8 NoSQL

NoSQL ja Big-data (sekä pilvipalvelut) ovat tämän hetken suosituimmat puhenaiheet it-maailmassa. NoSQL-järjestelmien tavoitteena on tietokantakerroksen helppo, pitkälti automatisoitu horisontaalinen skaalattavuus ja tietomallin joustavuus. [47.]

Insinöörityössäni käsittelen tarkemmin yleisintä MongoDB:n NoSQL-tietokantaa, vaikkakin uusia NoSQL-tuotteita tarjoavia startup-yrityksiä ilmestyy jatkuvasti. Esimerkiksi Riak on erittäin kiinnostava NoSQL-ratkaisu, kuten myös Cassandra, jota tituleerataan Big-data -orientuneeksi wide-column store -tyyliseksi NoSQL-tietokannaksi. NoSQL-kantatyyppejä ovat esimerkiksi Wide column store, Key-value store ja Document Store. Kaikkia näitä yhdistää se, että niistä ei ole kuin osa relaatiotietokannoissa olevia ominaisuuksia ja ne ovat suunniteltu ja toteutettu erityistä tehtävää varten. [47.]

MongoDB

MongoDB on 10gen-yhtiön kehittämä ja ylläpitämä avoimen lähdekoodin perustuva dokumenttisuuntautunut NoSQL-tietokanta. MongoDB on yksi monista NoSQL-tietokannoista, jotka eivät perustu perinteisiin relaatiotietokantoihin. MongoDB on tämän hetken suosituin NoSQL-tietokanta. [48.]

Sen sijaan, että data tallettaisiin perinteiseen taulukkomaiseen rakenteeseen perinteisen relaatiotietokannan tapaan, MongoDB tallentaa datan strukturoidussa JSON-formaatissa, jossa on ns. dynaaminen skeema. JSON on lyhenne sanoista *JavaScript Object Notation*, ja se on yksinkertainen tiedonsiirtomuoto, jota on helppo käyttää JavaScript-ohjelmissa. MongoDB kutsuu tätä muotoa BSON-muodoksi**.** BSON on MongoDB-tietokannan käyttämä objektimuoto Tällöin tiedon haku ja ylläpito pitäisi olla helpompaa ohjelmistokehittäjän näkökulmasta. MongoDB-tietokannasta on alusta asti huomioitu korkeakäytettävyys ja siihen liittyvät tarpeet. 10gen alkoi kehittää MongoDB:ta vuonna 2007, jolloin yhtiö rakensi palvelualustan, joka muistutti Windows Azure- tai Google App Engine -palveluita. Maaliskuussa 2010 versio 1.4, MongoDB:sta oli valmis tuotantoon. Uusin vakaa versio, 2.4.5, julkaistiin heinäkuussa 2013. [49.]

MongoDB toimii Unix- ja Windows-käyttöjärjestelmissä. Korkean käytettävyyden osalta MongoDB tukee replikointia ja ja sharding-ominaisuutta. Sharding-ominaisuus tarkoittaa horisontaalista osiointia, joka mahdollistaa yksittäisen tietokannan jakamisen kaikkien klusterin koneiden kesken. MongoDB tukee monien eri ohjelmointikieliä. Valmiit rajapinnat löytyvät C-, C++-, Erlang-, Haskell-, Perl-, PHP-, Python- ja Ruby- ja Scalakielille. MongoDB soveltuu moniin eri käyttötarkoituksiin, esimerkiksi tiedon arkistointiin, tapahtumalokituksiin, dokumenttien tallennukseen, ohjelmistokehitykseen, tosiaikaiseen tilastointiin ja analysointiin, peleihin, mobiiliin ja paikkatietojärjestelmiin. [50, s. 320–350.]

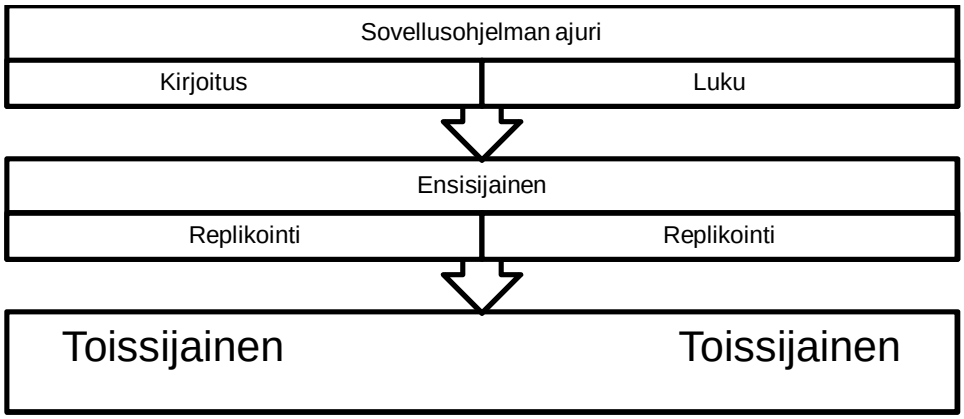
Monet NoSQL-tietokannoista on kehitetty www-sovelluksia varten, eivätkä ne tue join-liitoksia, transaktioita ja muita SQL-kielelle tuttuja ominaisuuksia. Taulukossa 2 on esitelty SQL-termi vs. Mongo-termi.

Taulukko 2. SQL-termi vs. Mongo-termi [51.]

Sql-termi	**Mongo-termi**
Tietokanta	Database
Taulu	Collection
Indeksi	Index
Rivi	Bson Document
Kolumni	Bson field
Group by	Aggregation
Join	Embedding and Linking

MongoDB:n skeeman voi muuttaa lennossa, ilman kannan alasajoa tai muuta ulospäin näkyvää toimenpidettä, mutta tietokanta on kannattavaa. [51, s. 64–67]

MongoDB -tietokannan vakio-ominaisuutena on replikointi, jonka konfiguraatiota kutsutaan Replica Setiksi (kuva 15.)

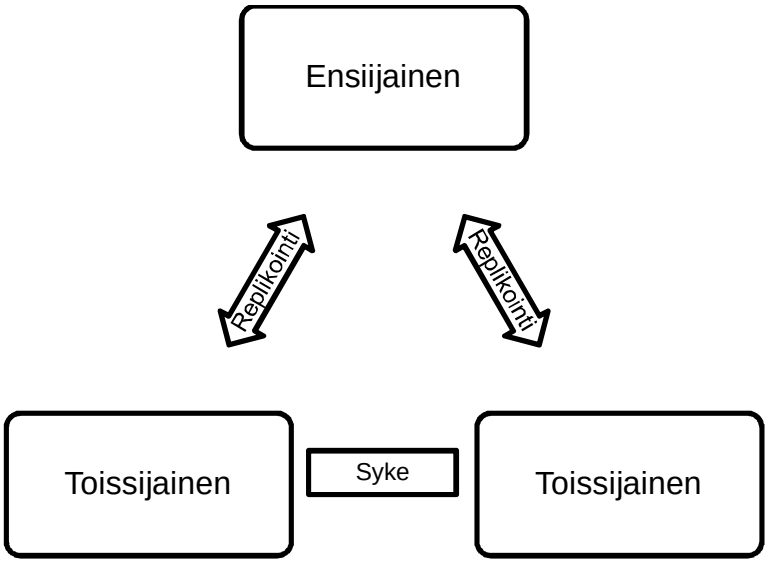


Kuva 15. MongoDB Replica Set [49.]

MongoDB:n Replica Set vaatii vähintään kolme noodia ja sen perusominaisuus on automaattinen vikasietoisuus. Se toipuu täysin automaattisesti häiriö- ja vikatilanteista. [49.]

Entinen ensisijainen noodi tulee toissijaiseksi, kun se toipuu häiriöstä tai vikatilanteesta. Kuvassa 16 on esitelty toissijainen noodi. Siinä näkyy sisäinen heartbeat-syke ja replikointi eri noodien välillä.

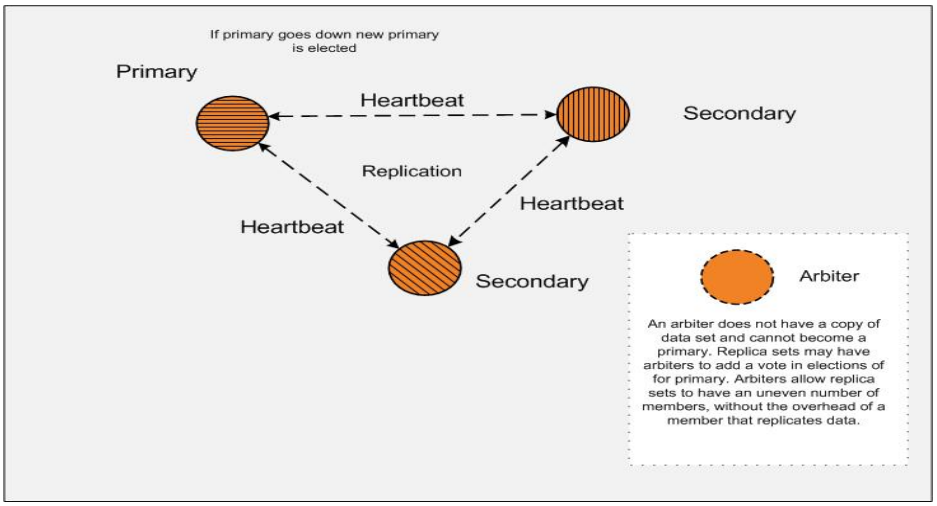
Jokainen noodi ottaa yhteyden automaattisesti muihin noodeihin muutaman sekunnin kuluttua ja varmistaa, että kaikki on kunnossa. [49.]



Kuva 16. Toissijainen noodi [49.]

Kannattaa lukea ensisijaista noodia, koska se on ainoa, joka sisältää uusimmat tiedot täysin varmasti. Kaikki koneet Replica Setissä on yhtä vahvoja selvitäkseen täydestä kuormituksesta. [49.] Liitteessä 4 on käytännön esimerkki MongoDB:n käytöstä sekä replicasetin konfiguroinnista.

Kolmas noodi-tyyppi on olemassa ja sitä kutsutaan sovittelijaksi (arbiter). Kuvassa 17 on esitelty Arbiter-noodi.



Kuva 17. Arbiter-noodi [49.]

Arbiter-noodin tehtävänä on toimia ns. erotuomarina sisäisessä äänestyksessä ensisijaisesta (primary) noodista. Arbiter-noodilla voidaan vähentää sisäistä kuormitusta. [49.]

9 Big data

Liikenne- ja viestintäministeriön teettämässä Kide-ohjelman *Big data Suomessa* -tutkimuksessa kerrotaan, että suomalaisorganisaatiot ovat havahtuneet big dataan, mutta sen rooli niiden omalle toiminnalle on vielä epäselvä. [52.]

Big datalla tarkoitetaan valtavia ja muodoltaan vaihtelevia datamassoja sekä niiden hyödyntämistä. Haasteita on kolme: tiedon valtava määrä, vaihteleva muoto sekä koko ajan kiihtyvä tahti, jolla dataa tuotetaan. [52.]

Viidenneksellä verkkokyselyyn vastanneilla ei ollut selvää big data -visiota. Käytännön kokemukset puuttuivat, sillä 62 prosentilla ei ollut vielä kokemusta Hadoopista, joka on tunnetuin big dataan liittyvä teknologia. Hadoop on avoimen lähdekoodin Apache-lisensoitu ohjelmisto, joka julkaistiin vuonna 2006. Yli puolet vastaajista kertoi kuitenkin big datan nousevan tärkeäksi osaksi liiketoimintaa nyt tai tulevaisuudessa. [52.]

Keskeisin ongelma vastaajien mielestä on osaajapula. Vaikka oppilaitokset ovat ryhtyneet ratkaisujen etsimiseen, konkreettiset tulokset nähdään vasta vuosien päästä. [52.]

Tutkimuksen mukaan big data -ilmiötä hidastavat myös oikeudelliset asiat. Aikaisemmin on hyödynnetty ensisijaisesti oman organisaation tuottamaa tietoa, mutta nyt kuvaan astuu myös datan yhdistely. Kun tiedon tallennus- ja käsittelykyky kasvaa, herää uusia kysymyksiä yksityisyyden suojasta ja tekijänoikeuksista. Kun yhdistellään verkosta vapaasti ladattavia tietoja ja analysoidaan niitä algoritmeilla, tieto saattaa muuttua nykyisen lainsäädännön mukaan liian yksilöiväksi. Lisäksi se voi rikkoa tekijänoikeuksia. [52.]

Euroopan tulisi kiiruhtaa mukaan big data -ilmiöön, joka on tällä hetkellä Yhdysvaltojen ja Intian hallussa. Euroopan unionilla ei ole tällä hetkellä yhteisesti hyväksyttyjä datamassojen käsittelyä koskevia säännöksiä. Se saattaa hidastuttaa alan kehittymistä unionin alueella. [52.]

Osaajapulan ja lainsäädäntöön liittyvien haasteiden lisäksi big dataan liittyviä ongelmia ovat ilmiön ja teknologioiden jäsentymättömyys, kalliit kehityskustannukset sekä epäkypsä ja hajanainen teknologiakirjo. Tutkimuksessa todetaan, ettei kaupallisilla yrityksellä ole varaa odottaa toimintaympäristön kehittymistä. Yrityksiltä vaaditaan investointirohkeutta. Eräs ratkaisu voisikin olla yhteistyö eri aloilla toimivien ja erikokoisten yritysten välillä. [52.]

Julkishallinnossa on tutkimuksen mukaan alettu toteuttamaan avoimeen dataan liittyviä suunnitelmia edistyksellisesti. Onnistuneilla avoimen datan hankkeilla voisi olla vientiarvoa sellaisenaan. [52.]

9.1 Hadoop

Hadoop on uusi ja tunnetuin yksittäinen teknologia. Se analysoi avoimen koodin suuria datamassoja. Hadoop tulkitsee niin kuvat, videot, lokitiedostot, paikkatiedot, ääninauhoitteet kuin Twitter-viestitkin. Käytännössä se havainnoi kaiken sellaisen datan, jonka analysoiminen on perinteisesti ollut hankalaa. Hadoop osaa lajitella sekavan datavuoren ja kaivaa esiin vastauksia. Innoittajina on toiminut Googlen

julkaisemat artikkelit tiedon tallentamisesta (GFS eli Google File System) ja suurten datamassojen hajautetusta analysoimisesta (MapReduce). Hadoopin vastineet näille olivat HDFS (Hadoop Distributed File System) ja MapReduce. Jälkimmäiseen Googlella on Yhdysvalloissa useita patentteja, joiden on tarkoitus myöntämisvuonna 2010 annettujen lausuntojen mukaan suojata yritystä ja sen ydinteknologioita. [53, s. 7–10.]

Moni organisaatio ei välttämättä ole kiinnostunut Hadoopista sellaisenaan. Raakadatasta organisaation toiminnan näkökulmasta hyödyllisiin tuloksiin pääsemiseksi vaatii liikaa kehitystyötä, asetusten säätämistä, integraatiota ja testaamista. Todennäköisesti Hadoopia hyödynnetään joko valmiin jakelun käyttöönotossa tai sitä käytetään osana laajempaa kokonaisuutta. Gartner ennusti vuoden 2013 alussa, että vuonna 2015 jo noin kaksi kolmasosaa kehittyneistä analytiikkaratkaisuista sisältäisi Hadoopin. [53, s. 7–10.]

Suomessa Hadoopin tarjoamiin mahdollisuuksiin on herätty myöhään ja osaajapulasta on tulevaisuudessa tulossa merkittävä pullonkaula big data -hankkeille. [53, s. 7–10.]

9.2 Muistinvarainen analytiikka

Muistinvarainen analytiikka on toinen tunnettu big datan yhteydessä esille tuotu teknologia. Siinä käytetään palvelinklusterin keskusmuistia datan tallentamiseen analyysin ajaksi, jolloin latenssi on mahdollisimman pieni ja analyysi nopeutuu huomattavasti verrattuna tilanteeseen, jos tallennustilana käytettäisiin mekaanista kovalevyä tai SSD-levyä. [53, s. 7–10.]

9.3 Keskeiset tiedontarpeet ja kehityskohteet

Keskeisiä tunnistettuja haasteita big datan kehityksessä ovat erityisesti tiedon jakamisen kanavien rajallisuus ja valtavien tietomäärien käsittelyyn sisältyvät oikeudelliset seikat. [53, s. 7–10.]

Big data -tietouden jakamisessa tiivistyy kaksi toisiinsa kytkeytyvää erittäin keskeistä ongelmaa: koulutuksen puute ja osaajien vähyys. Siirtymä big datan ymmärtämiseen ei

ole sujunut riittävällä vauhdilla. Muutokseen reagoiminen saattaa olla helppoa, mutta opetusohjelmien muutokset erityisesti keskeneräisten ja kehittyvien tieteenalojen osalta ovat osoittautuneet vaikeiksi. Tietotekniikan osalta big data ei ole ensimmäinen tällainen ilmiö. Valitettavasti menetetään usein se merkittävin kilpailuetu, kun odotetaan ilmiön vakiintumista. [53, s. 7–10.]

Vaikka useat yritykset ovat havahtuneet big data -ilmiön kasvuun, ei monessakaan ole panostettu oman osaamisen kehittämiseen riittävästi. Yritysten liiketoiminnan intelligenssi voi olla kypsää, mutta usein siihen sitoutuneet henkilöt ovat jo niin työllistettyjä, ettei heitä ehditä jatkokouluttaa big datan tekniikoihin. Ne harvat, jotka ovat perehtyneet big dataan syvällisemmin, ovat työnantajilleen erityisen arvokkaita resursseja. [53, s. 7–10.]

Big data -ilmiö luo paineen datalähteiden paremmalle ymmärtämiselle. Aiemmin on tallennettu hyvin tunnettua ja strukturoitua dataa omasta liiketoimintakoneistosta relaatiokantoihin ja sitä kautta tietovarasto/BI-käyttöön, mutta nyt ollaan tuntemattomalla alueella tämän ulkopuolisen datan osalta. On katsottava kauemmas ja tutkittava sekä kumppaniverkon tarjontaa että julkisia avoimia tietolähteitä. Ennen kaikkea on löydettävä eri lähteistä ja eri muodoissa tulevia datavirtoja ja yhdistäviä tekijöitä sekä ymmärrettävä yhdistämisen kautta saavutettavia liiketoiminnallisia mahdollisuuksia. [53, s. 7–10.]

Datalähteiden kasvaessa lisääntyy myös dataa tarjoavien tahojen ja teknologioiden valikoima. Uusia yrityksiä syntyy ja valtiolliset rajat ja säännökset hämärtyvät pilvipalveluiden myötä. Vakaita ja kestäviä yrityksiä voi olla vaikea erottaa epärehellisin keinoin menestystä hakevista tahoista. Luotettavuuden mittareita ei välttämättä ole edes olemassa. Datan kuitenkin korostetaan olevan toimialasta riippumatta yksi yrityksen keskeisimmistä ja arvokkaimmista omaisuuseristä. [53, s. 7–10.]

Monelle yritykselle nämä osaamisalueet ovat vaikeasti koottavissa samaan asiantuntijaan tai tiimiin. Asiantuntija-apua on rajallisesti saatavilla, johtuen mm. big data -teknologian nuoresta iästä. Big data-hankkeiden käynnistäminen ja edistäminen tuntuvat hankalalta ja riskialttiilta ilman riittävää asiantuntemusta. Se luo hidasteen koko ilmiön konkretisoitumiselle niin Suomessa kuin muuallakin. [53, s. 7–10.]

Tässä kokonaisuudessa tarvitaan vahvaa osaamista oman liiketoimintaympäristön ja liiketoimintaan keskeisesti liittyvän datan osalta, sekä luovaa ajattelukykyä uusien näkökulmien havaitsemiseksi ja mahdollisten kilpailuetujen tunnistamiseksi. [53, s. 7–10.]

Suurin big data -ilmiötä jarruttava tekijä on juridisesti vaikeaselkoinen toimintakenttä. Tietojenkäsittelyssä on jo aiemmin luotu lainsäädäntöä ja yhteisesti hyväksyttyjä tapoja tiedon käsittelyyn, jakamiseen ja julkaisemiseen, mutta big data herättää tätä kysymystä lisää. Big datan myötä lähdetään kokonaan uuteen ulottuvuuteen datan yhdistelyssä ja prosessoivien volyymien kasvussa. [53, s. 7–10.]

Kun tiedon tallennus- ja käsittelykapasiteetti kasvaa räjähdysmäisesti, niin erityisesti yksityisyyden suoja ja tekijänoikeudet nousevat esille. Tiedon lataaminen, tallentaminen ja yhdistely rikkovat helposti kansallisia tai kansainvälisiä tekijänoikeussäädöksiä. Oikeusturvan valvominen muodostunee yhdeksi suurimmista haasteista tulevaisuuden tietoyhteiskunnassa. [53, s. 7–10.]

EU:lla on tällä hetkellä huomattavia vaikeuksia tuottaa yhteisesti hyväksyttyjä pelisääntöjä datan käsittelyyn suurissa määrissä ja kansalliset rajat ylittäen. Se on ehkä merkittävin tekijä EU:n hitaudessa kasvattaa kansainvälisesti kilpailukykyisiä big data -toimijoita. [53, s. 7–10.]

10 Pilvipalvelut

Yritykset, jotka tarjoavat pilvipalveluita ovat tarttuneet big datan avaamiin mahdollisuuksiin. Rajaton tallennustila ja laskentakapasiteetti, jotka joustavat täysin tarpeen mukaan, sopivat erinomaisesti ratkaisuiksi moniin big datan esittämiin haasteisiin. Suurten julkisten datamassojen laaja levittäminen vaatii keskitettyä ratkaisua ja pilvipalvelut tarjoavat luonnollisen vaihtoehdon avointen ja maksullisten datavarastojen keskitettyyn jakeluun siten, että ne ovat helposti yhdistettävissä dataa käyttävien yritysten pilvipalveluissa säilyttämään omaan dataan. [53, s. 7–10.]

Joustavuuden lisäksi pilvipalvelut tarjoavat hyvän alustan julkisten datavarantojen jakamiselle sekä yksityisten datavarantojen myymiselle yritysten ja muiden organisaatioiden käyttöön. Tallennettavalle datamäärälle ei ole kattoa, ja koska yhä

suurempi osa pilvipalveluiden tarjoajista tarjoaa myös analytiikkaa palveluna, voidaan suuriakin pilveen sijoitettuja datamassoja yhdistellä ja analysoida tehokkaasti. Tulevaisuudessa yhä suurempi osa datasta tullaan tallentamaan pilveen. Jos pilvipalveluista löytyy niin suuria datamassoja, että niiden siirtäminen ei ajallisesti tai kustannusten takia ole mahdollista, niin ei ole muuta mahdollisuutta kuin siirtää oma data samaan pilvipalveluun, jossa massiivinen datakin on. Siirtäminen kuitenkin aiheuttaa viivettä ja synnyttää kustannuksia, joten tämä luo painetta siirtää oma data pysyvästi pilveen, missä se on helppo yhdistää samasta pilvipalvelusta löytyvien muiden datavarantojen kanssa. Pilvipalveluiden haasteena ovat tietoturva, -suoja ja juridiset kysymykset. Tunnetuimmat palveluntarjoajat eivät tarjoa mahdollisuutta rajata datan tallennuspaikaksi yksittäistä kansallisvaltiota, vaan rajaus tapahtuu EU-tasolla eli taataan datan olevan EU-alueella sijaitsevassa palvelinkeskuksessa. Esimerkiksi Amazon ja Microsoft toimivat siten. [53, s. 7–10.]

Suomella olisi tarjottavanaan palvelinkeskusten sijaintipaikkana paljon vahvuuksia, kuten vakaa yhteiskunta, laadukas infrastruktuuri ja koulutettu työvoima. Esimerkiksi Google on rakentanut palvelinkeskuksen Haminaan ja Microsoft ilmoitti syksyllä 2013 investoivansa 200 miljoonaa euroa omaan palvelinkeskukseen Suomessa. [53, s. 7–10.]

Pilvipalvelut-teoksessa Heino (2010) kirjoittaa, että NoSQl ja Hadoop-asioita ajavien mielestä maailmassa on vikaa. Tavanomaiset levyjärjestelmät olisi alun perinkin pitänyt rakentaa niin, että niihin saa petatavukaupalla levykapasiteettia ja että niillä pystyttäisiin palvelemaan miljoonia käyttäjiä. Miksi pitäisi maksaa relaatiotietokantojen valmistajille kovaa hintaa lisensseistä ja niiden tukimaksuista, jos kaikki muut joutuvat it-alalla tinkimään katteistaan? [54, s. 87–91.]

Kun dotcom-kupla puhkesi vuosituhannen vaihteen jälkeen, startup-yrityksiltä jäi vapaaksi reilusti ylimitoitettuja teknisiä ympäristöjä, jotka ovat sen jälkeen kiertäneet käytettyinä ympäri maailmaa. Järjestelmät oli mitoitettu niin, että käyttäjänä olisi voinut olla koko maailma. Tallennuksessa oikein isoihin kapasiteetteihin asti skaalautuviin levylaitteisiin tarvittiin silloin, ja tarvitaan edelleenkin, merkittävä alkuinvestointi. Alkuinvestoinnin jälkeen kapasiteetin laajentaminen on jonkin aikaa edullista, mutta sitten teknologia loikkaa eteenpäin, ja uudempien laitteiden kapasiteetista tulee huomattavasti edullisempaa. Levylaitteiden vaihtaminen uudempiin tai edullisempiin on kivulias operaatio, joka vaatii pilvipalveluntarjoajan jatkuvakäyttöisessä

moniasiakasympäristössä paljon valmistelua ja tekijöiltään erikoisosaamista. Siten joudutaan hankkimaan markkinahintaa kalliimpaa tallennuskapasiteettia. [54, s. 87–91.]

Tallennuslaitteet voi virtualisoida. Tallennuksen virtualisoinnista on jokaisella isolla valmistajalla oma näkemys, eikä siihen ole standardeja. Virtualisointilaitteisto voi mahdollistaa ison ja pienen levylaitteen yhteiskäytön tai kopioinnit eri valmistajien laitteiden välillä. Kustannuksissa tulee huomioida lisenssimaksut. Alennuksilla ja hyvillä sopimuksilla lisenssimaksun vaikutus jää pieneksi, mutta pilvitoimija on sitoutunut lisenssiehtoihin. Ehdot voivat muuttua, ja muutosten jälkeen laitteisto ei välttämättä sovi enää pilvitoimintaan. [54, s. 87–91.]

Tavallisella yritysasiakkaalla levylaitteiden suorituskyky on lähinnä teoreettinen ongelma. Riittää kun laite on ehjä ja siinä on jonkin verran levyjä (levylaitteiden suorituskyky on kiinni levyjen kappalemäärästä). Keskikokoisessa levylaitteessa on kaksi ohjainyksikköä, ja jos toinen niistä vikaantuu, suorituskyky putoaa puoleen. Jos suorituskyvyn kanssa on ongelmia, toimittajan vakiokysymys on: onko asiakkaalla tarpeeksi levyjä. [54, s. 87–91.]

Isot pilvipalveluiden tarjoajat tekevät pilvitallennusjärjestelmänsä itse. Hierarkkisen tallennusjärjestelmän rakentaminen ei ole kovin monimutkaista, jos rakennetaan nimenomaan itselle dedikoitua ympäristöä, jota ei tarvitse toistaa muille asiakkaille. Avoimesta lähdekoodista löytyy monia palasia, joita voi hyödyntää sellaisenaan. Kun rakentaa ympäristön itse, ei tarvitse käyttää aikaa ja energiaa esimerkiksi helppokäyttöisen graafisen käyttöliittymän luomiseen tai monien eri RAID-tasojen tukemiseen ja niiden suorituskyvyn optimoimiseen. Oman tallennusjärjestelmän kohdalla haasteet saattavat olla testaamisessa ja ympäristön saamisessa vikasietoiseksi ja luotettavaksi. [54, s. 87–91.]

Pilvitallennusjärjestelmien rakentaminen lähtee liikkeelle siitä, että tallennuksen perusyksikkönä kannattaa käyttää palvelinlaitteita. Palvelinlaitteiden kaikki komponentit ovat kilpailtuja. Linuxin yleistymisen ansiosta palvelimia saa kokonaan ilman ohjelmistoja, joissa voisi olla toimintaa rajoittavia lisenssiehtoja. Rakennettavan ympäristön kapasiteettina käytetään palvelimien sisäisiä SATA- tai SAS-kiintolevyjä. Tarvitaan vain älykäs ohjelmisto, jolla yhdistetään useita kymmeniä, satoja tai jopa tuhansia palvelimia yhdeksi isoksi tallennusjärjestelmäksi. Palvelimista muodostuvia ja ohjelmistolla yhdeksi loogiseksi laitteeksi liimattavia tallennusjärjestelmiä on tarjolla

myös kaupalliseen tarkoitukseen, ja ne ovat potentiaalisia vaihtoehtoja private cloud -pilveä rakentavien organisaatioiden tallennustarpeeseen. [54, s. 87–91.]

Strukturoimattoman tiedon kasvu on vaatinut uutta tapaa lähestyä tietojen hyödyntämistä. Yksi erityinen haaste on isojen havaintoaineistojen analysointi. Tämä tunnetaan puoliksi vakiintuneella termillä big data. Isoilla aineistoilla tarkoitetaan kooltaan niin suuria massoja, että tavanomaiset tietokantavälineet eivät niihin tehoa. Tämä tarkoittaa petatavuja, eksatavuja sekä tsettatavuja. Näitä aineistoja halutaan käyttää, koska niiden kautta on mahdollisuus liiketoiminnallisten trendien havaitsemiseen, tautien ehkäisemiseen, rikosten estämiseen sekä kaiken kaikkiaan erittäin vaikeiden ongelmien ratkaisemiseen. [54, s. 87–91.]

Googlella, Amazonilla ja Yahoolla on välineet ja tuhansia palvelimia näiden aineistojen pyörittämiseen. MapReduce on Googlen patentoima ohjelmistokehys, jolla isoja havainoaineistoja voidaan käsitellä suurina ryppäinä. Map-prosessi jakaa ongelman pienempiin osiin, ja Reduce-prosessi kerää pienempien ongelmien ratkaisut yhteen. MapReducelle tarjottava aineisto voi olla palvelimen tiedostojärjestelmässä tai relaatiotietokannan sisällä. MapReducella voidaan esimerkiksi lajitella petatavun verran tietoja muutamassa tunnissa. Kovin monella yrityksellä maailmassa ei ole siihen itsellään mahdollisuuksia. Teratavun kokoisen tietokannan lajittelu tai analyysi vie paljon resursseja, mutta siirtämällä prosessointi pilvisovellukselle analyysi voidaan tehdä lyhyessä ajassa. [54, s. 87–91.]

Useimmissa pilviin hajautetuissa strukturoimattoman datan prosessoinnin palveluissa käytetään Hadoopia. Apachen hallinnoima Hadoop-projekti tuottaa tiedostojärjestelmässä asuvan kerroksen, joka hajauttaa sovelluksen suurelle määrälle prosessoivia yksiköitä. Hadoopissa on ”rack awarenes”, joka huomioi, että osa prosessoivista noodeista sijaitsee tietoliikenteen suhteen lähellä ja osa kaukana. Hadoop-projektissa rakennetaan myös omaa Hadoop File System-tiedostojärjestelmää. Yahoo ja IBM ovat Hadoopin tukijoita, ja isoista pilvitoimijoista ainakin AOL:n, Twitterin ja Facebookin kerrotaan käyttävän Hadoopia. MapReduce ja Hadoop saattavat kuulostaa kaukaisilta tavallisen yrityskäyttäjän mielestä, mutta edut on konkretisoitu New York Timesille tehdyn ajon kautta. Lehdelle tehtiin Hadoop-ajo, jossa neljän megatavun TIFF-kuvatiedosto tulostettiin 11 miljoonaksi PDF-dokumentiksi 24 tunnin sisällä. Sadalla EC2-instanssilla tehdyn ajon palvelin- ja tallennuskapasiteetin hinta oli 240 dollaria. [54, s. 87–91.]

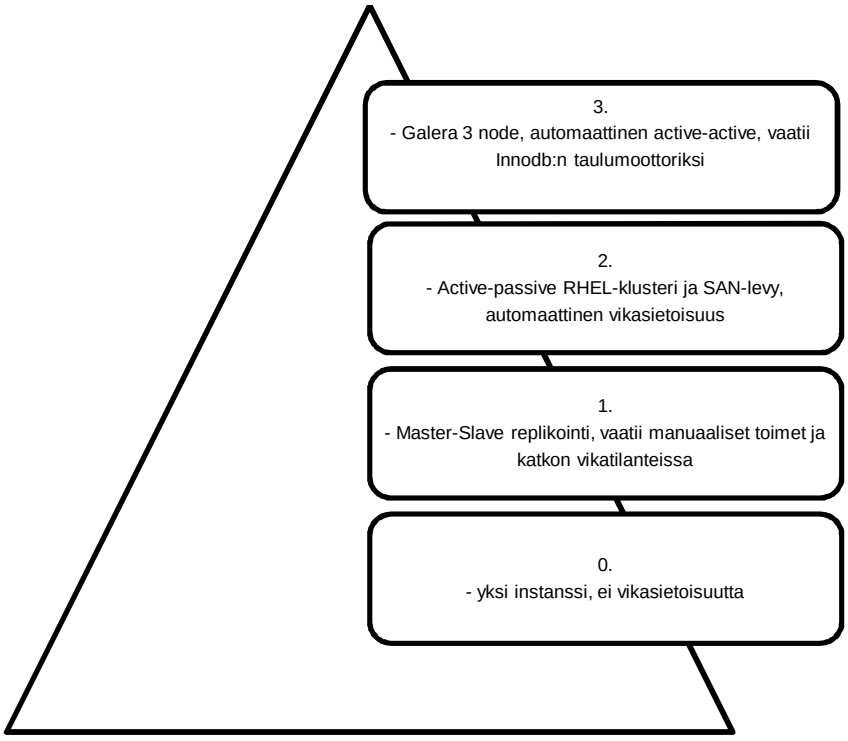
Toinen näkökulma pilvitallennukseen on NoSQL. NoSQL-nimen alla kehitetään tietokantamaisia ohjelmia, joihin on helppo tallentaa strukturoimatonta tietoa ja joilla ei ole SQL-kielen rajoitteita. Myös ilmaisuus on merkittävä seikka. Googlen AppEngine-palvelu tarjoaa asiakkailleen Googlen oman datastore-säilön, jossa on oma kevennetty tietokanta. Myös Amazon tarjoaa yksinkertaistettua SimpleDB-tietokantakorviketta. Tietokantaa tai sen vastinetta tarvitsevat sovellukset saadaan teknisesti toimivalla tavalla hajautettua käytössä oleville tuhansille palvelimille. Ei myöskään tarvitse olla sidoksissa tietokantatoimittajien maksuihin tai ehtoihin. [54, s. 87–91.]

Asiakkailla saattaisi olla kiinnostusta Hadoopin ja NoSQL:n kaltaisiin uusiin rakenteisiin. Amazonin S3-palvelun voi järjestää myös omaan tekniseen ympäristöön ja replikoida omaan S3-ympäristöön talletettuja tietoja Amazonin pilveen. Tallennuksien suhteen pilvipalveluiden kautta markkinoille on saatu uutta. [54, s. 87–91.]

11 Pohdinta

Yrityksen valitsema HA-ratkaisu riippuu täysin budjetista, yrityksen strategiasta ja halutusta uptime:sta. Open Source -vaihtoehtoja on erittäin paljon ja varsin villejä toteutusvaihtoehtoja, joita ei suositella kriittisiin tuotantoympäristöihin. Uusia Nosql-tuotteita ilmestyy jatkuvasti kiihtyvällä tahdilla ja lähes kaikkien Nosql-ratkaisujen taustalla on jokin uusi startup-yritys. Kaupallisia vaihtoehtoja on myös paljon, joten valinta on haastavaa.

Vaihtoehdot SLA:n ja toteutusmahdollisuuksien suhteen voidaan määritellä MySQL/MariaDB -tietokannan osalta hintajärjestykseen kuvan 18 mukaisesti.



Kuva 18. MySQL/MariaDB, HA-toteutus

0. Yksi instanssi eli yksi palvelin. Ei vikasietoisuutta.

1. Master-Slave replikointi. Vaatii vikatilanteissa manuaaliset korjaustoimet.

2. Active-passive RHEL-klusteri SAN-levyllä. Toipuu automaattisesti vikatilanteessa eikä tarvitse manuaalista korjausta.

3. Galera kolmella noodilla. Automaattinen active-active, vaatii Innodb-taulumoottorin tai vaihtoehtoisesti Postgresql:n tai MongoDB:n 3 noodin replikasetin.

Palvelutasosopimuksen (SLA) hinnat määräytyvät noodien mukaan. Mitä enemmän noodeja on käytössä, sitä kalliimpi on myös sopimus. Järjestelmän saatavuus ja klusterointi ei tule korvaamaan varmuuskopiointia tai järjestelmän automaattista valvontaa. Uusia järjestelmiä ilmestyy jatkuvasti, insinöörityössäni on esitelty vain muutamia yleisempiä järjestelmän saatavuuden ratkaisuja.

Lähteet

1. Avoin lähdekoodi. 2013. Verkkodokumentti. Centre for Open Systems and Solutions. <http://coss.fi/avoimuus/avoin-lahdekoodi/>. Luettu 22.4.2013.

2. Järvinen, P. 2003. IT-tietosanakirja. Porvoo: WSOY

3. Open Source. 2013. Verkkodokumentti. AfterDawn Oy. <http://fin.afterdawn.com/sanasto/selitys.cfm/open_source>. Luettu 22.4.2013.

4. Open Source. 2007. Verkkodokumentti. Systeemityö. <http://www.pcuf.fi/sytyke/lehti/kirj/st20072/ST072-04A.pdf>. Luettu 19.7.2013.

5. OSI-malli. 2002. Verkkodokumentti. Tampere University of Technology. <http://www.cs.tut.fi/etaopetus/titepk/luku19/OSI.html>. Luettu 16.7.2013.

6. Tilastokeskus – tietotekniikan käyttö yrityksissä. 2011. Verkkodokumentti. <http://www.stat.fi/til/icte/2011/icte_2011_2011-11-24_tie_001_fi.html>. Luettu 19.4.2013.

7. Hämäläinen, P. 2002. Ryväspalvelimet. <http://www.tietokone.fi/artikkelit/ryvaspalvelimet>. Luettu 16.7.2013.

8. Ahola, V. 2010. Verkkodokumentti. Kymenlaakson ammattikorkeakoulu. Palvelinklusteri ja verkkolevy. <http://publications.theseus.fi/bitstream/handle/10024/23691/ville_ahola.pdf?sequence=3>. Luettu 19.4.2013.

9. High Availability. 2002. Verkkodokumentti. Plan4it. <http://www.plan4it.fi/faq.html>. Luettu 16.7.2013.

10. Ratilainen, T. 2010. Verkkodokumentti. Jyväskylän ammattikorkeakoulu. Palvelinklusterin rakentaminen Debian-ympäristöön. <https://publications.theseus.fi/bitstream/handle/10024/23999/Ratilainen_Tero.pdf?sequence=1>. Luettu 19.4.2013.

11. High Availability. 2013. Verkkodokumentti. Wikipedia. <http://fi.wikipedia.org/wiki/High_availability>. Luettu 19.4.2013.

12. Korkea käytettävyys. 2013. Verkkodokumentti. Cisco Systems. <http://www.cisco.com/web/FI/solutions/datacenter/architecture/usability_home.html>. Luettu 19.4.2013.

13. Palvelutasosopimukset (SLA). 2009. Verkkodokumentti. Helsingin yliopisto. <http://www.cs.helsinki.fi/group/cinco/teaching/2009/soc-seminaari/abstracts/kautto_abstract.pdf>. Luettu 19.4.2013.

14. Heikkinen, T-P., Luoma, M. 2012. Verkkodokumentti. Aalto-yliopisto. <https://noppa.aalto.fi/noppa/kurssi/s-38.3191/.../S-38_3191_sla-2012.pdf>. Luettu 14.8.2013.

15. DB-Engines. 2013. Verkkodokumentti. Solid IT. <http://db-engines.com/en/>. Luettu 22.9.2013.

16. PostgreSQL. 2013. Verkkodokumentti. The Postgresql Global Development Group. <http://www.postgresql.org/docs/9.0/static/hot-standby.html>. Luettu 6.10.2013.

17. MySQL. 2013. Verkkodokumentti. Webopas. <http://www.webopas.net/mysql.html>. Luettu 10.7.2013.

18. MariaDB. 2012. Linux Magazine 125.

19. Hakkarainen, A. 2011. Oracle-tietokannan tehokas hallinta. Helsinki: Readme.fi.

20. Ingo, H. 2013. Verkkodokumentti. How to Evaluate which MySQL High Availability Solution Best Suits You. <http://openlife.cc/system/files/Evaluating%20HA%20alternatives%20MySQL%20-%20PLMCE%202013.pdf>. Luettu 15.9.2013.

21. DRBD. 2011. Verkkodokumentti. LINBIT HA-Solutions. <http://www.drbd.org/>. Luettu 15.9.2013.

22. RedHat Cluster Suite. 2013. Verkkodokumentti. Red Hat, Inc. <https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Cluster_Suite_Overview/s1-rhcs-intro-CSO.html>. Luettu 15.9.2013.

23. Pacemaker. 2013. Verkkodokumentti. Open Source. <http://clusterlabs.org/>. Luettu 10.7.2013.

24. Corosync. 2013. Verkkodokumentti. The Corosync Cluster Engine. <http://corosync.github.io/corosync/>. Luettu 10.7.2013.

25. OpenNode HA Cluster. 2013. Verkkodokumentti. OpenNode Cloud Platform. <http://opennodecloud.com/documentation/howtos/opennode-ha-cluster-howto/>. Luettu 10.7.2013.

26. Bambling, S. 2013. Active-Passive Cluster. <http://snozberry.org/images/cluster1.pdf>. Luettu 16.9.2013.

27. PostgreSQL 9.0. 2013. Verkkodokumentti. Replication. <http://www.postgresql.org/docs/9.0/static/warm-standby.html>. Luettu 20.9.2013.

28. MySQL. 2013. Verkkodokumentti. Replication. <http://dev.mysql.com/doc/refman/5.0/en/replication.html>. Luettu 21.9.2013.

29. Tungsten replicator. 2013. Verkkodokumentti. Google Project Hosting. <http://code.google.com/p/tungsten-replicator/>. Luettu 16.9.2013.

30. Tungsten replicator. 2012. Verkkodokumentti. Percona Inc. <http://www.percona.com/live/mysql-conference-2013/sessions/using-tungsten-replicator-solve-replication-problems>. Luettu 16.9.2013.

31. Galera replication. 2013. Verkkodokumentti. Codership. <http://codership.com/products/galera_replication>. Luettu 16.9.2013.

32. MySQL Cluster. 2013. Verkkodokumentti. Oracle Corporation. <http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-overview.html>. Luettu 20.9.2013.

33. Streaming replication. 2013. Verkkodokumentti. Postgre SQLWiki. <http://wiki.postgresql.org/wiki/Streaming_Replication>. Luettu 20.9.2013.

34. Slony-I. 2010. Verkkodokumentti. Slony Development Group. <http://slony.info/>. Luettu 20.9.2013.

35. x-DB replication. 2013. Verkkodokumentti. EnterpriseDB Corporation. <http://www.enterprisedb.com/products-services-training/products-overview/xdb-replication-server-multi-master>. Luettu 16.9.2013.

36. Postgres-XC. 2013. Verkkodokumentti. Source Forge. <http://sourceforge.net/apps/mediawiki/postgres-xc/index.php?title=Main_Page>. Luettu 20.9.2013.

37. PostgreSQL Hot Standby. 2013. Verkkodokumentti. The PostgreSQL Global Development Group. <http://www.postgresql.org/docs/9.0/static/hot-standby.html>. Luettu 20.9.2013.

38. Repmgr. 2012. Verkkodokumentti. 2nd Quadrant Limited. <http://www.repmgr.org/>. Luettu 6.10.2013.

39. Brisco, T. 1995. Verkkodokumentti. Rutgers University. <http://tools.ietf.org/html/rfc1794>. Luettu 20.9.2013.

40. VIP. 2013. Verkkodokumentti. Oracle Corporation. <http://docs.oracle.com/cd/E23389_01/doc.11116/e21036/ha002.htm>. Luettu 21.9.2013.

41. HAProxy. 2011. Verkkodokumentti. Severalnines AB. <http://www.severalnines.com/resources/clustercontrol-mysql-haproxy-load-balancing-tutorial>. Luettu 21.9.2013.

42. Piranha. 2012. Verkkodokumentti. Red Hat, Inc. and others. <http://www.redhat.com/software/rha/cluster/piranha/>. Luettu 15.9.2013.

43. PGPool. 2013. Verkkodokumentti. PgPoolWiki. <http://www.pgpool.net/mediawiki/index.php/Main_Page>. Luettu 20.9.2013.

44. PGPool-II. 2011. Verkkodokumentti. <http://pgpool.projects.pgfoundry.org/contrib_docs/simple_sr_setting/>. Luettu 20.9.2013.

45. PGbouncer. 2013. Verkkodokumentti. FusionForge. <http://pgfoundry.org/projects/pgbouncer/>. Luettu 15.8.2013.

46. PostgresHA. 2013. Verkkodokumentti. Open Source Consulting Group. 2013. <http://www.openscg.com/2013/04/postgresql-clustering/>. Luettu 15.8.2013.

47. NoSQL. 2013. Verkkodokumentti. Gofore. <http://gofore.com/tag/nosql-tietokanta/>. Luettu 10.9.2013.

48. MongoDB. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/MongoDB>. Luettu 12.9.2013.

49. MongoDB. 2013. Verkkodokumentti. MongoDB, Inc. <http://docs.mongodb.org/manual/core/replication-introduction/>. Luettu 12.9.2013.

50. Linux Magazine. 2013. Issue 153

51. Linux User & Developer. 2013. Issue 129

52. Lyytikäinen, S. 2013. Verkkodokumentti. Tietoviikko. <http://www.tietoviikko.fi/kaikki_uutiset/haluatko+toita+big+data+osaajista+huutava+pula/a931262>. Luettu 22.9.2013.

53. Alanko, M. Salo, I. 2013. Big data Suomessa. Liikenne- ja viestintäministeriön julkaisu

54. Heino, P. 2010. Pilvipalvelut. Talentum Media Oy

Liite 1. MySQL MasterSlave -konfigurointi

Testejä varten asennettiin neljä Ubuntu Server 13.04 virtuaalipalvelinta Virtualbox-ohjelmistolla

Taulukko 3. Palvelintopologia

Nimi	IP-osoite
Dbvserv1	192.168.1.79
Dbvserv2	192.168.1.80
Dbvserv3	192.168.1.81
Lbserv1	192.168.1.82

Luodaan testikanta komennolla:

create database replika;

Tämän jälkeen dbserv1-palvelimen my.cnf-tiedostoa muokkaamalla määritetään server-id ja asetetaan binäärilogitus replika-tietokannalle päälle.

server-id	=	1
log_bin	=	/var/log/mysql/mysql-
bin.logexpire_logs_days	=	10
max_binlog_size	=	100M
binlog_do_db	=	replika
binlog_ignore_db	=	mysql

show master status;

Master-tietokannasta otetaan kopio mysqldump-komennolla. Tämän jälkeen määritetään replikointikäyttäjä master-palvelimella.

GRANT REPLICATION SLAVE ON *.* TO 'slave_user'@'%' IDENTIFIED BY 'slave_user';

FLUSH PRIVILEGES;

Show master status;

Slave-palvelin:

Relay-logitus on hyvä laittaa päälle.

server-id=2

relay-log-index=slave-relay-bin.index

relay-log=slave-relay-bin

datadir=/home/sami/mysql/slave/data

CHANGE MASTER TO MASTER_HOST='192.168.1.79',MASTER_USER='repl_ user',

MASTER_PASSWORD= 'repl_user',MASTER_LOG_FILE='',MASTER_LOG_POS=4;

Start slave;
SHOW SLAVE STATUS \G

Replikointi on toiminnassa, jos slave-palvelimella näkyvät seuraavat rivit:

Slave_IO_Running: Yes

Slave_SQL_Running: Yes

Liite 2. Galera Cluster -konfiguraatio

/etc/init.d/mysql start

mysql -u root

mysql> DELETE FROM mysql.user WHERE user='';

mysql> GRANT ALL ON *.* TO root@'%' IDENTIFIED BY 'root';

mysql> UPDATE mysql.user SET Password=PASSWORD('root') WHERE User='root';

mysql> GRANT ALL ON *.* to galer@'%' IDENTIFIED BY 'Passu';

update-rc.d mysql defaults

MySQL dbserv1

/etc/mysql/conf.d/wsrep.cnf

wsrep_provider=/usr/lib/galera/libgalera_smm.so

wsrep_cluster_address="gcomm://"

wsrep_galer_auth=galer:Passu

wsrep_galer_method=rsync

/etc/init.d/mysql restart

MySQL dbserv2

/etc/mysql/conf.d/wsrep.cnf

wsrep_provider=/usr/lib/galera/libgalera_smm.so

wsrep_cluster_address="gcomm://192.168.1.80:4567"

wsrep_galer_method=rsync

wsrep_galer_auth=galer:Passu

/etc/init.d/mysql restart

MySQL dbserv3

/etc/mysql/conf.d/wsrep.cnf

wsrep_cluster_address="gcomm://192.168.1.81:4567"

wsrep_galer_method=rsync

wsrep_galer_auth=galer:Passu

wsrep_provider=/usr/lib/galera/libgalera_smm.so

/etc/init.d/mysql restart

uudestaan dbserv1

/etc/mysql/conf.d/wsrep.cnf

wsrep_cluster_address="gcomm://192.168.1.79:4567"

mysql -u root -p

mysql> set global wsrep_cluster_address='gcomm://192.168.1.79:4567';

Ympäristömuuttujat

mysql> show status like 'wsrep%';

Liite 3. PostgreSQL Streaming – replikointi

Minimiasennus replikoinnista:

Master-palvelin, 192.168.1.79
Slave-palvelin, 192.168.1.80

Muokataan konfiguraatio-tiedostoa
/etc/postgresql/9.1/main/postgresql.conf
#listen_addresses = 'localhost' # what IP address(es) to listen on;
listen_addresses='*'
#wal_level = minimal # minimal, archive, or hot_standby
wal_level = hot_standby
#max_wal_senders = 0
max_wal_senders = 3

Luodaan replikointikäyttäjä
root@dbserv1:~# psql -h localhost -U postgres -W -c "CREATE USER ruser WITH REPLICATION PASSWORD 'password';"

Laitetaan oikeudet kuntoon yhteydenottoa varten tiedostoon
/etc/postgresql/9.1/main/pg_hba.conf
host replication ruser 192.168.1.80/32 md5

Pysäytetään master-palvelin komennolla /etc/init.d/postgresql stop

Tämän jälkeen muokataan slave-palvelimen tiedostoa
/etc/postgresql/9.1/main/postgresql.conf
#listen_addresses = 'localhost'# what IP address(es) to listen on;
listen_addresses = '*'
#hot_standby = off
hot_standby = on
Tämän jälkeen pysätetään slave-palvelin ja poistetaan vanhat tiedot
#/etc/init.d/postgresql stop
#cd /var/lib/postgresql/9.1/main/

#rm -rf *

Tämän jälkeen muokataan
/var/lib/postgresql/9.1/main/recovery.conf
primary_conninfo = 'host=192.168.1.79 port=5432 user=ruser password=password'
standby_mode = on

Kopioidaaan kaikki data
rsync -av /var/lib/postgresql/9.1/main/* 192.168.1.80:/var/lib/postgresql/9.1/main/

Käynnistetään molemmat palvelimet
/etc/init.d/postgresql start

Tämän jälkeen voidaan tarkistaa, toimiiko replikointi
psql -h localhost -U postgres -W -c "select * from pg_stat_replication;"

Liite 4. MongoDB:n replicat-asennus

Tämä tehdään kolmelle palvelimelle:

sudo apt-get install mongodb-server

#Muuta portti ja julkinen ip-osoite

/var/lib/mongodb #hakemiston voi tyhjentää

dbserv1 /etc/mongofb.conf

bpath=/var/lib/mongodb

bind_ip = 127.0.0.1

port = 27017

replSet = setti

Käynnistä nodet uudelleen

sami@dbserv1:/var/lib/mongodb$ mongo --host 192.168.1.79

MongoDB shell version: 2.2.4

connecting to: 192.168.1.79:27017/test

> rs.conf()

Null

> rs.initiate()

{

"info2" : "no configuration explicitly specified -- making one",

"me" : "192.168.1.79:27017",

"info" : "Config now saved locally. Should come online in about a minute.",

"ok" : 1

}

> rs.conf()

{

"_id" : "setti",

"version" : 1,

"members" : [

{

"_id”: 0,

"host" : "192.168.1.79:27017"

}

[

}