Tampereen Ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Teemu Tattari

Opinnäytetyö

Työasema- ja palvelinvirtualisointi vapaan lähdekoodin sovelluksilla

Työn ohjaaja Harri Hakonen
Tampere 06/2010

Tampereen Ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä Teemu Tattari
Työn nimi Työasema- ja palvelinvirtualisointi vapaan lähdekoodin sovelluksilla
Sivumäärä 45
Valmistumisaika 06/2010
Työn ohjaaja Harri Hakonen

TIIVISTELMÄ

Virtualisointi on eräs 2000-luvun suosituimmista teknologioista IT-maailmassa eikä suotta: tarjoaahan se paljon etuja ja vähän haittoja - ainakin paperilla. Virtualisointi säästää rahaa tarjoamalla kustannustehokkaan tavan hallita palvelimia, joten ympäri maailman yritykset ja organisaatiot ovat lähteneet virtualisointi-buumiin mukaan, niin myös meillä Suomessa.

Tämän opinnäytetyön tarkoitus on pureutua edullisiin, toissijaisiin vaihtoehtoihin toteuttaa virtualisointia käyttämällä ilmaisia, avoimen lähdekoodin sovelluksia. Tarkoitus on myös osoittaa, että virtualisointi onnistuu ilman järeää palvelinrautaa myös kotioloissa.

Opinnäyte ei varsinaisesti perustu minkään yrityksen antamaan aiheeseen, vaan on oma aiheeni. Olen kuitenkin hiljattain työni puolesta ollut mukana virtuaalipalvelinjärjestelmän käyttöönotossa ja sitä kautta päässyt näkemään, mitä asioita virtualisointijärjestelmän rakentamisessa on hyvä ottaa huomioon. Tämä työ poikkeaa melko paljonkin siitä, mitä olen omassa työympäristössäni nähnyt ja kokenut, mutta perusasiat pysyvät ennallaan. Tässä työssä pyritään tutkimaan erilaisia virtualisoinnin mahdollisuuksia ja sovellutuksia keskittymättä mihinkään tiettyyn ympäristöön tai kaupalliseen sovellukseen.

Avainsanat virtualisointi avoin lähdekoodi palvelin työasema

TAMK University of Applied Sciences
Business Information Systems

Writer	Teemu Tattari
Thesis	Desktop and Server Virtualization Using Open-Source Software
Pages	45
Graduation time	06/2010
Thesis supervisor	Harri Hakonen

ABSTRACT

Virtualization is one of the biggest IT booms of the 21st century, and for a simple reason: what it offers are a lot of benefits with very few drawbacks - at least on paper. Virtualization technology offers a very cost-effective way to manage servers which in turn saves money, which is the reason why many organizations and companies around the globe have hopped on the virtualization ship - including many Finnish organizations and companies.

The purpose of this thesis is to delve into cheaper, alternative methods to make virtualization happen by using open-source software. One of my aims is to show that virtualization does not necessarily require a lot of computing power and that you can achieve it fairly simply right in your home as well.

This thesis is a result of my own work and was not assigned by any specific company or organization. Recently, however, I have had the opportunity to participate in deploying a virtualized server system at my workplace and have thereby witnessed some of the obstacles present in such a large-scale operation. On one hand, though, this thesis is largely unrelated to what I have encountered in my work environment, but on the other hand virtualization does not differ greatly from that of my work environment. The purpose of this thesis, however, is to explore different kinds of virtualization methods without focusing on any particular type of application, and most notably not focusing on any commercial application.

Keywords virtualization open source server desktop

Sisällysluettelo

1 Johdanto

Virtualisointi on yritysmaailmassa ollut hyvin tuttu termi jo viimeisten viiden vuoden ajan. Syyt virtualisoinnille vaihtelevat, mutta useimmiten taustalla on pyrkimys kustannussäästöihin, optimoida olemassa olevaa järjestelmää, kokeilla uusia asioita tai sekoitus hieman kaikkea. Tässä työssä otetaan selvää virtualisoinnin tarjoamiin mahdollisuuksiin niin koti- kuin työympäristössä. Tarkoitus on selventää, mitä virtualisointi on, mitä erilaisia tyyppejä virtualisointeja on ja lopuksi, kuinka niitä voidaan soveltaa.

Tämä opinnäytetyö käsittää neljä eri virtualisointiohjelmaa, jotka kaikki pohjautuvat avoimeen lähdekoodiin. Avoin lähdekoodi mahdollistaa niin tutustumisen virtualisointiin kuin myös sen, että sitä voidaan tarpeen mukaan hyödyntää tuotannollisissa ympäristöissä - kaikki tämä ilmaiseksi, vailla ohjelmistokustannuksia. Ohjelmistot on valittu siten, että ne eroavat tarpeeksi toisistaan, jolloin voidaan tutkia eri virtualisoinnin tyyppejä ja taustoja.

Käyttäjän ja ympäristön tarpeet määrittävät sen, miten virtualisointi kannattaa toteuttaa. Yleinen tilanne on se, että halutaan luoda virtuaalipalvelinjärjestelmä, jonka alla on useita virtuaalikoneita eli virtuaalipalvelimia. Tyypillisiä ympäristöjä tällaiseen ovat työympäristöt sekä kaupalliset sovellutukset kuten www-ylläpitopalvelut. Tällaiseen tilanteeseen kannattaa luonnollisesti valita enemmän palvelinvirtualisointiin keskittynyt ohjelmisto. Virtualisointi sopii kotikäyttöön myös, ja tämä työ pyrkii ottamaan kantaa myös siihen osoittamalla, että virtualisointi ei vaadi järeää palvelinrautaa tai pitkiä opintoja virtualisointitekniikoista. Kotiympäristön tarve virtualisoinnilla on todennäköisesti vähäisempi, joten jokin työasemapohjainen ohjelmisto voi olla oikea valinta. Virtualisointi voi olla myös tietoturvaa: halutaan hajauttaa jonkin palvelimen eri toimintoja, kuten www-palvelut, tietokannat, käyttäjäkirjautumiset ym. eri lohkoihin. Myös näissä tilanteissa virtualisointi on varteenotettava vaihtoehto.

Kaiken kaikkiaan tämän työn tarkoitus on saattaa kokemattomampikin käyttäjä virtualisoinnin pariin käymällä läpi muutamia eri mahdollisuuksia kertoen niiden hyvistä ja osin huonoista puolista. Tarkoitus ei ole ottaa puolueellisesti kantaa minkään virtualisointisovelluksen puolesta, vaan esitellä erilaisia vaihtoehtoja. Käyttäjän tarpeista

riippuu se, mikä ohjelmisto tai tapa on paras, yksiselitteisiä vastauksia on harvassa. Työssä käydään läpi vaihe vaiheelta virtualisointitekniikkojen taustaa sekä niiden käyttöönottoa, asennusta että käytännön sovellutuksia. Lopussa pohditaan, miten ympäristöistä voitaisiin saada hieman parempi tai kehittyneempi, sekä mietitään tietoturvaa virtuaalikoneiden näkökulmasta.

2 Virtualisointi

2.1 Mitä virtualisointi on

Virtualisointi on tietokonemaailman termi, jolla on useita enemmän tai vähemmän epäselviä tarkoituksia. Yleistettynä virtualisoinnissa on kyse jonkin objektin muuttamisesta näennäiseksi eli virtuaaliseksi, yleensä ohjelmallisesti. Virtualisoinnilla voidaan jakaa fyysisen laitteen resurssit useammalle *virtuaalikoneelle* ilman, että ne tietävät edes olevansa virtuaalisia, jolloin ne pystyvät käyttäytymään kuin oikea, fyysinenkin laite, eivätkä ne ole tietoisia toisista virtuaalikoneista. (Golden, 2007, 10)

Tämän opinnäytetyön kontekstissa virtualisointi käsittää käyttöjärjestelmä- ja palvelinvirtualisointitekniikat, jolloin voidaan käyttöjärjestelmä A:n alaisuudessa suorittaa myös käyttöjärjestelmiä B ja C – yhtäaikaisesti – samalla palvelimella tai normaalilla kotitietokoneella.

Historia lyhyesti

Virtualisointi sinällään ei ole mikään uusi tekniikka, vaan sijoittuu jo 1960 - 1970-lukujen taitteeseen, jolloin IBM:n M44 -projektin yhteydessä luotiin ensimmäiset virtualisoinnin askeleet. Tarkoituksena silloin oli paremmin hyödyntää ison järjestelmän kapasiteettia jakamalla se useampiin *virtuaalisiin* osioihin. Siihen aikaan virtualisointia käytettiin pääasiassa simuloimaan järjestelmän moniajoa (engl. multi-tasking, multi programming), jolloin esim. koodin kirjoittaminen ja korjaaminen onnistuisivat samaan aikaan. (Dittner, 2007, 3 - 4)

Virtualisoinnin tilanne nykypäivänä

Vuonna 2007 Suomessa 41 % isoista organisaatioista oli jo siirtynyt virtualisointiin, tai olivat ainakin tietoisia siitä. Toukokuussa 2009 lukemaksi ilmoitettiin jopa jo 75 % isoista organisaatioista. (Mäkinen, 2009). Joidenkin arvioiden mukaan

maailmanlaajuisesti jopa 50 % ohjelmistoista on virtualisoituja vuoteen 2012 mennessä, kun lukema tällä hetkellä on 16 %. (Gartner 2009).

2.2 Hyödyt ja haasteet

2.2.1 Virtualisoinnin hyödyt

Ehkä merkittävin etu virtualisoinnista pitkällä aikavälillä on helppo yksilöidä: raha. Jos oletetaan tilanne, että yrityksellä on käytössään kolme fyysistä palvelinta ja että yhden palvelimen sähkönkulutus (mukaan lukien ilmastointi) vuotta kohden on n. 700 euroa (Kotilainen, 2010), niin tällöin kolme palvelinta muodostaa yhteensä 2100 euron laskun vuotuisesti.

Kun kaksi näistä palvelimista virtualisoidaan, jäljelle jää vain yksi palvelin, joten teoriassa säästetään 1400 euroa vuodessa jo tällä. Mikäli palvelimia on useita kymmeniä, kuten monissa jo keskikokoisissa yrityksissä on, säästöt nousevat todella huomattaviksi. Ja kun sähköä säästyy, ollaan myös entistä ekologisempia.

Toisaalta on järkevää huomioida myös se, että jos useita palvelimia virtualisoidaan yhteen, nousee yksittäisen palvelimen kuorma ja siten virrankulutus huomattavasti. Ihan puhtaisiin "1400 euron" säästöihin ei siis välttämättä päästä.

Toinen merkittävä virtualisoinnin etu on siirrettävyys, tai tarkemmin virtuaalipalvelimien siirrettävyys. Koska virtuaalipalvelimet ovat ohjelmallisia fyysisen laitteiston sijaan, niitä voidaan helposti ja nopeasti siirtää toisiin virtuaalipalvelinjärjestelmiin esim. laiterikon sattuessa. (Dittner, 2007, 14)

Virtuaalipalvelimia voidaan myös vaivattomasti kloonata, mikä mahdollistaa mm. uusien ohjelmistopäivitysten testauksen turvallisessa ympäristössä, ennen kuin päivitykset ajetaan tuotantoympäristöön. Samaan tapaan virtualisointi sopii myös ohjelmistojen kehittäjille ja testaajille, sillä kehitys ja testaus voidaan toteuttaa identtisissä ympäristöissä, mutta kuitenkin täysin eri virtuaalikoneissa. (Dittner, 2007, 12)

Rahan- ja ympäristön säästöä teknisempi hyöty on myös vähentynyt resurssien hukkaaminen. Useimmat ei-kriittiset palvelimet käyttävät hyvin vähän prosessoritehoa tai muistiresursseja, jolloin voi olla loogista virtualisoida ne johonkin isompaan kokonaisuuteen, jossa ne voivat jatkaa "joutenoloaan" ilman mittavaa resurssien hukkaan heittämistä. (Dittner, 2007, 12)

2.2.2 Haasteet virtualisoinnissa

Siirtyminen virtualisoituun ympäristöön tuo mukanaan haasteita ja myös ongelmatilanteita. Ensinnäkin, virtuaalipalvelinjärjestelmän aloituskustannukset saattavat nousta hyvin korkeiksi, sillä virtuaalipalvelinjärjestelmien tarvitsee olla erittäin tehokkaita. Toiseksi, mukaan kustannuksiin täytyy usein laskea myös levytallennusjärjestelmät sekä mahdolliset verkon aktiivilaitteiden uusimiset, jotta tehokkaiden palvelinlaitteiden hyöty saadaan maksimoitua. (Big Fat Finance Blog, 2009)

Laitteistoinvestointien lisäksi tärkeää on ottaa huomioon myös käyttöjärjestelmien ja mahdollisten muiden asiaan kuuluvien hallintasovelluksien lisensointijärjestelyt. Useissa tapauksissa ohjelmistoja lisensoidaan prosessorimäärän ja/tai -ytimien mukaisesti. Tämä saattaa aiheuttaa epäselvyyttä ja ongelmatilanteita lisenssien kanssa järeimmissä palvelinjärjestelmissä, joissa prosessoreita ja ytimiä on paljon. (Virtualization.info, 2008)

Ylläpidollisesti yksi iso haaste on virtuaalipalvelimien *tehokas* ylläpito. Virtuaalipalvelimia voi olla useita kymmeniä, ja niitä täytyy edelleen valvoa ja ylläpitää. Jotkin virtualisointiohjelmistot tarjoavatkin erillisiä valvontatyökaluja, mutta manuaalisesta työstä ei silti kokonaan päästä eroon.

Kotikäytössä kustannukset ovat tarpeesta riippuen hyvinkin eri luokkaa. Alkuun pääsee, kunhan omistaa suhteellisen modernin laitteiston, joka kykenee virtuaalikoneita ajamaan. Tarkemmin laitteistovaatimuksia on kuvailtu luvussa kolme.

2.3 Virtualisointityypit x86-arkkitehtuurilla

Virtualisointitekniikoita on muutamia erilaisia, ja niiden toteutus sekä niiden puutteet pohjautuvat tapaan, jolla x86-arkkitehtuuri on kehitetty. X86 on Intelin kehittämä mikroprosessoriarkkitehtuuri, joka nykyään löytyy suurimmasta osasta kotitietokoneita ja nykyään myös palvelimista. Muita tunnettuja prosessoriarkkitehtuureja ovat mm. PowerPC- ja RISC-arkkitehtuurit, mutta mitkään näistä eivät ole keskenään yhteensopivia. X86:n lisäksi voidaan vielä puhua IA-32- ja x86-64-varianteista, joilla viitataan bittisyyteen eli 32- ja 64-bitin x86-arkkitehtuurit.

Virtualisoinnin tekniikoista puhuttaessa viitataan yleensä x86-arkkitehtuuriin ja sen tuomiin haasteisiin siinä, kuinka virtualisointia voidaan toteuttaa. Tämän varjolla virtualisointi voidaan jakaa kolmeen päätyyppiin:

- Täysvirtualisointi (nk. ”binäärikäännös”)
- Käyttöjärjestelmätason virtualisointi ja paravirtualisointi
- Laitteisto-avustettu täysvirtualisointi

Täysvirtualisointi ja laitteisto-avustettu täysvirtualisointi ovat paljolti samantyyppisiä menetelmiä. Molemmissa pyritään emuloimaan eli virtualisoimaan itse fyysistä laitteistoa, ei niinkään mitään tiettyä käyttöjärjestelmää. Käyttöjärjestelmätason virtualisointi ja paravirtualisointi keskittyvät puolestaan emuloimaan käyttöjärjestelmiä.

Yllä oleva luettelo on järjestelty siten, että monimutkaisin toteutettava on ensin ja helpoin viimeisenä. Monimutkaisuudella tarkoitetaan ohjelmoinnin haasteita, ei niinkään ohjelmiston käytön monimutkaisuutta. Omat ongelmansa tuovat myös 64-bittiset käyttöjärjestelmät, koska niiden virtualisointi vaatii tukea itse prosessorilta, jolloin se käytännössä onnistuu ainoastaan laitteisto-avusteisella virtualisoinnilla. (VMware, 2007)

2.3.1 Täysvirtualisointi

Vuonna 1999 VMware julkisti ensimmäisen tuotteensa, VMware Workstationin, joka oli ensimmäinen x86-pohjainen virtualisointiohjelmisto, joka pystyi *täysvirtualisointiin*.

VMwaren tiennäyttämisen myötä myös muut valmistajat ovat sittemmin tuoneet julki vastaavanlaisia ohjelmistoja.

Täysvirtualisoinnilla tarkoitetaan tilannetta, jossa pyritään emuloimaan suoraan laitteistoa eikä käyttöjärjestelmää. Käytännössä se mahdollistaa sekalaisten käyttöjärjestelmäyhdistelmien virtualisoinnin, koska emulointi kohdistuu suoraan x86-arkkitehtuuriin. VMware käytti tämän toteuttamiseen tekniikkaa, jota kutsutaan binäärikäännökseksi (engl. binary translation).

Binäärikäännös piti pitkään paikkansa monipuolisimpana virtualisointitekniikkana, ja se on edelleenkin hyvin yleinen ohjelmallinen ratkaisu. Se on myös monimutkaisin ohjelmallisesti toteuttaa, koska koodin täytyy osata mukautua ja reagoida lennossa erilaisiin ongelmatilanteisiin. Toisena haittapuolena monimutkaisuuden lisäksi esiin saattaa nousta suorituskyky, joka kärsii, kun ohjelmakoodia joudutaan mukauttamaan senhetkiseen tilanteeseen sopivaksi. (Intel, 2009)

Täysvirtualisoinnin isoimpia etuja muihin tekniikoihin verrattuna on sen yleisesti hyvä yhteensopivuus isäntä- ja vieraskoneiden välillä, jolloin voidaan ajaa sekalaisia yhdistelmiä eri käyttöjärjestelmistä. Ainoastaan 64-bittiset käyttöjärjestelmäyhdistelmät eivät toimi keskenään x86-arkkitehtuurin ja täysvirtualisoinnin tekniikan rajoitusten vuoksi. Muun muassa tämä ongelma on pyritty ratkaisemaan laitteisto-avusteisella virtualisoinnilla, jota käsitellään paremmin seuraavassa kappaleessa. Täysvirtualisointi edellyttää nk. *hypervisorin* läsnäoloa. Hypervisor on virtualisoinnin kerros eli käytännössä erillisohjelmisto, joka sijoittuu isäntäjärjestelmän ja virtuaalikoneiden väliin ja huolehtii niin virtuaalikoneiden virtualisoinnin hallinnasta kuin myös tietoturvasta (esim. etteivät toiset virtuaalikoneet näe toisiaan). (Vmware, 2007, 2 - 3)

2.3.2 Laitteisto-avustettu virtualisointi

Uusimpana tekniikkana x86-arkkitehtuurin virtualisoinnissa on laitteisto-avustettu virtualisointi (engl. hardware-assisted virtualization). Itse tekniikkana se ei ole kuitenkaan täysin uusi, sillä se pohjautuu jo muinaisiin IBM:n järjestelmiin.

Noin vuoden 2006 tienoilta asti se on kuitenkin ollut saatavilla myös x86-sarjan prosessoreissa (AMD/Intel). Merkittävin ero muihin menetelmiin on se, että prosessori tukee suoraan eräitä virtualisoinnissa tarvittavia käskykantoja ja järjestelmäkutsuja. Tällöin ohjelmakoodin suoritusvastuu pystytään delegoimaan kokonaan prosessorille, eikä ole tarvetta emuloida näitä käskyjä ohjelmallisesti. Tämä näkyy suoraan laajana tukena eri käyttöjärjestelmäyhdistelmille, niin 32- kuin 64-bittisillekin.

Kyseinen tekniikka vaatii AMD-V- tai Intel-VT -tuen prosessorilta, mikä on rajoittanut jonkin verran sen käyttömahdollisuuksia. AMD:llä tuki löytyy lähes kaikista viimeisen 3 - 4 vuoden aikana julkaistuista tuotteista. Intelin prosessoreista tuki nykyään löytyy jo käytännössä kaikista uusista tuotteista, mutta tuki kannattaa varmistaa erikseen etenkin muutaman vuoden vanhojen prosessorien kanssa.

Käytännössä laitteisto-avustettu virtualisointi on suositeltavin vaihtoehto ympäristöihin, joissa on tarve virtualisoida useita ja erilaisia käyttöjärjestelmiä, sillä sen yhteensopivuus on varminta. Kaupallisissa virtualisointiohjelmistoissa se saattaa myös olla ainoa vaihtoehto, jolloin tuotteen asennus ei onnistu palvelimeen, josta kyseistä tukea ei löydy.

Laitteisto-avusteinen virtualisointi ei kuitenkaan poissulje esim. täysvirtualisointia. Tällöin on täysin mahdollista, että ohjelmisto hyödyntää molempia tekniikoita parhaan suorituskyvyn takaamiseksi. Myös laitteisto-avusteinen virtualisointi toteutetaan käyttäen hypervisor-tekniikkaa, kuten täysvirtualisointissa.

2.3.3 Käyttöjärjestelmätason virtualisointi ja paravirtualisointi

Käyttöjärjestelmätason virtualisointi (engl. operating system-level virtualization) on eräs virtualisoinnin muoto, mikä rajoittuu ainoastaan johonkin tiettyyn käyttöjärjestelmään. Tällaisessa tapauksessa ei voida luoda virtuaalikoneita, joissa olisi eri käyttöjärjestelmä kuin mitä isäntäkoneessa. Käytännössä käyttöjärjestelmätason virtualisointia tapaa ainoastaan Linux- ja Unix-pohjaisissa ympäristöissä.

Käyttöjärjestelmätason virtualisointi tapahtuu niin, että luodaan isäntäkoneen sisälle uusi virtuaaliympäristö eli ”säiliö” (engl. container), jolla on käyttöoikeus isäntäkoneen kerneliin eli ytimeen, jonka avulla virtuaalikone operoi. Virtuaaliympäristölle myös voidaan asettaa erilaisia rajoituksia resurssien suhteen. Kyseinen virtuaaliympäristö käyttäytyy kuin fyysinen laite ja näyttääkin käyttäjälle siltä. (Kirill Kolyshkin, 2006)

Ohjelmistosta (esim. OpenVZ, FreeBSD Jail) riippuen, virtuaalikoneella voi olla hierarkinen tiedostojärjestelmä ihan kuin isäntäkoneellakin, jolloin se koostuu kasasta alihakemistoja. Se voi olla myös levykuvatyyppinen (engl. image), ja esim. FreeBSD toteuttaa hierarkisen tiedostojärjestelmämallin, jolloin muutokset virtuaalikoneen tiedostoihin on mahdollista tehdä suoraan isäntäkoneesta. Xen ja OpenVZ käsittelevät virtuaalikoneita levykuvina, jolloin niiden tiedostohierarkia ei ole suoraan muokattavissa isäntäkoneelta.

Käyttöjärjestelmätason tyyppinen virtualisointiratkaisu tarjoaa hyvää suoritustehoa, sillä virtuaalikoneella on suora yhteys isäntäkoneen kerneliin, jolloin välistä putoaa tarve erilliselle virtualisointiohjelmistolle eli hypervisorille. Haittapuolena käyttöjärjestelmäympäristö on hyvin rajattu ja homogeeninen, eikä erilaisia käyttöjärjestelmäyhdistelmiä voida käyttää. Myös muutokset isäntäkoneen kerneliin (päivitykset esim.) saattavat vaikuttaa suoraan virtuaalikoneisiin.

Tässä työssä käsitellään FreeBSD:n *Jail(8)*-nimistä ohjelmistoa, jolla käyttöjärjestelmätason virtualisointi voidaan toteuttaa FreeBSD:llä. Jail on ollut mukana FreeBSD-julkaisuissa jo versiosta 4 asti (eli n. vuodesta 2000).

Toinen vastaava ohjelmisto on *OpenVZ*, joka on kehitetty Linux-jakeluja varten. OpenVZ, kuten Jail, rajoittuu isäntäkoneen käyttöjärjestelmään eikä siten tue eri käyttöjärjestelmäyhdistelmiä.

Paravirtualisointi

Paravirtualisointi eroaa käyttöjärjestelmätason virtualisoinnista siten, että sen avulla voidaan ajaa myös eri käyttöjärjestelmäyhdistelmiä, esim. FreeBSD-Linux-yhdistelmiä.

Sen rajoite on kuitenkin siinä, että se vaatii ohjelmallisia muutoksia käyttöjärjestelmien kerneliin, joten ihan kaikki yhdistelmät eivät ole tuettuja. (Intel, 2009)

Paravirtualisointia ei käsitellä tässä työssä kovinkaan paljoa, sillä uudemmilla prosessoreilla voidaan tukeutua laitteisto-avusteiseen virtualisointiin, jolloin tarve täysin ohjelmallisille toteutuksille vähenee. *Xen*-ohjelmisto, jota tässä työssä käsitellään, pohjautuu kuitenkin paljolti paravirtualisointiin, vaikka se tukee myös laitteisto-avusteista virtualisointia.

3 Laitteisto- ja käyttöympäristövaatimukset

Virtualisointia suunnitellessa on hyvä ottaa huomioon erilaiset käytön tarpeet. Kotiympäristöön tuleva virtuaalipalvelin ei luultavasti vaadi järeätehoista tietokonetta, kun taas työympäristön tehotarve on todennäköisesti jo huomattavasti isompi. Lisäksi isäntäkoneen käyttöjärjestelmävalinta kannattaa miettiä läpi ajatuksen kanssa. Kotikäytössä helpoin tapa päästä alkuun lienee Windows ja VirtualBox -yhdistelmä, koska usein graafinen ympäristö on aloittelijalle helpompi kuin täysin tekstipohjainen ympäristö. VirtualBox tarjoaa myös selkeän oppaan uusien virtuaalikoneiden luomiseksi.

Mikäli tarkoitus on luoda palvelinjärjestelmä ja useita virtuaalikoneita, harkinnan arvoisia ovat Linux/UNIX-järjestelmät, jos vain käyttökokemusta niistä löytyy. Tämä kaikki olettaen, että pysytellään vapaan lähdekoodin ohjelmistoissa. Kaupallisista ohjelmistoista löytyy tarjontaa hyvinkin eri tavalla ja mittasuhteilla.

3.1 Laitteistovaatimukset

Virtualisointi vaatii laitteistolta kolmea asiaa: keskusmuistia, prosessoritehoa ja kiintolevytilaa. Kaksi ensimmäistä kohtaa käytännössä määrittelevät sen, kuinka montaa virtuaalikonetta voidaan samanaikaisesti ajaa. Kiintolevytilaa on syytä olla siinä tapauksessa, jos on tarkoitus ajaa esim. tietokantapohjaisia virtuaalikoneita, sillä kannat saattavat kasvaa yllättävänkin isoiksi.

Koska kukin virtuaalikone varaa käyttöönsä lohkon isäntäkoneen keskusmuistista, on koneessa syytä olla muistia periaatteella "mieluummin enemmän kuin vähemmän". Jos virtualisoinnin tarve on vähäinen (esim. ei tietokantapalvelimia), vähemmälläkin muistilla pärjää. Käyttöjärjestelmätason virtualisointi osaa myös hyödyntää muistin yleensä hieman muita tekniikoita paremmin, koska se on suoraan yhteydessä isäntäkoneen kerneliin (eikä hypervisoriin), joka jakaa resursseja koko laitteistolle.

Toiseksi pullonkaulaksi muodostuu prosessoriteho, sillä virtuaalikoneet tarvitsevat myös suorituskykyä. Tosin tässäkin on oleellista huomioida, että virtuaalikoneissa ei

välttämättä esiinny tasaista käyttökuormaa, vaan ne käyvät joutoajolla ison osan ajasta. Suositeltavaa on kuitenkin, että koneesta löytyy vähintään tuplaydin-prosessori, jolloin suoritustehoa voidaan jakaa säikeittäin useammille koneille yhtäaikaisesti.

Järeämmässä yrityskäytössä useampiytimiset prosessorit sekä useita gigatavuja muistia on käytännön pakko. Etenkin raskaat palvelinohjelmistot syövät muistia helposti useita gigatavuja, jolloin palvelinjärjestelmästä on hyvä löytyä jo kymmeniä gigatavuja muistia. Varsinkin silloin, jos on tarkoitus ajaa useampia, jopa kymmeniä virtuaalikoneita.

Kotikäyttöön riittää suhteellisen uudella prosessorilla varustettu tietokone, josta löytyy vähintään kaksi, mutta kuitenkin mielellään neljä gigatavua muistia. Prosessorivalinnassa on syytä olla tarkkana ja varmistaa, että siitä löytyy virtualisointituki eli joko AMD-V tai Intel VT. Tieto löytyy parhaiten valmistajan kotisivuilta. Joissakin koneissa toiminto pitää vielä erikseen kytkeä BIOS:n asetuksista päälle.

Virtuaalikoneet syövät myös levytilaa, ja useimmissa ohjelmistoissa levytila varataan käyttöön heti virtuaalikonetta luodessa, vaikka sitä ei kaikkea heti täytettäisikään. Yrityskäytössä levytilan yhteydessä tulevat myös RAID-tason varmistukset kyseeseen varsinaisten tilavaatimusten lisäksi. Kuten aiemmissakin kohdissa, levytilan tarve riippuu täysin siitä, kuinka paljon virtuaalikoneita aiotaan asentaa sekä kuinka paljon tietoa täytyy mm. varmistaa.

3.2 Käyttöympäristön vaatimukset

Virtualisointiohjelmistojen käyttö edellyttää jonkin verran käyttöjärjestelmätuntemusta ja -osaamista. Tässä työssä käsitellyistä ohjelmistoista ainoastaan VirtualBoxissa on graafinen käyttöliittymä, joka tekee siitä hieman helppokäyttöisemmän. Tämän työn tarkoitus ei ole opiskella käyttöjärjestelmien käyttöä, mutta esimerkeissä käydään silti seikkaperäisesti läpi tarpeelliset komennot ja asetukset, joilla virtuaaliympäristön saa asennettua.

Tässä työssä on käytetty Windows 7:n ohella GNU Debian/Linux- ja FreeBSD-käyttöjärjestelmiä, ja koska komennot ja asennusohjeet ovat käyttöjärjestelmäkohtaisia jossain määrin, täsmälleen samat komennot eivät välttämättä löydy muista Linux-julkaisuista. Suurin ero tulee pakettihallintasovellusten kohdalla, sillä Debianin käyttämää *Aptitude*-ohjelmaa ei löydy muista kuin Debian-pohjaisista julkaisuista. Sama pätee FreeBSD:n ports-ohjelmistojen hallintaan.

Uudelle käyttäjälle, jolta todennäköisimmin löytyy jokin Windows-käyttöjärjestelmä, helpoin tapa päästä kokeilemaan virtualisointia on asentaa VirtualBox. Tässä työssä aloitetaan myös VirtualBoxilla, koska se mahdollistaa muiden tässä työssä käytettyjen käyttöjärjestelmien virtualisoinnin ilman, että niille tarvitaan erillistä, fyysistä laitteistoa.

Ainoastaan Xen vaatii aidon, fyysisen laiteympäristön, eikä täten suostu toimimaan VirtualBoxin alta. OpenVZ:aa ja FreeBSD:n Jailia voi harjoitella kätevästi VirtualBoxiin asennetuissa virtuaalikoneissa.

4 Virtualisointiympäristöjen ja -ohjelmistojen esittely

Tähän työhön on valittu neljä vapaan lähdekoodin virtualisointiohjelmistoa. Kukin niistä soveltuu hieman eri tarkoituksiin, mutta pääperiaatteiltaan ne toimivat samoin. Ohjelmien valitsemisen kriteerinä oli niiden avoimeen lähdekoodin pohjautuva lisensointi ja yleinen tunnettuus sekä toisistaan eroavat ominaisuudet.

Virtualisointiohjelmistoja on useita muitakin maksuttomia tai avoimeen lähdekoodiin pohjautuvia, mutta ajan- ja tilanpuutteen vuoksi kaikkia on mahdotonta käydä lävitse. Wikipedia tarjoaa kattavan ajan tasalla olevan listan muistakin sovelluksista osoitteessa http://en.wikipedia.org/wiki/Virtual_machine#List_of_virtual_machine_software.

4.1 VirtualBox

VirtualBox on alun perin saksalaisen Innotekin tuote, jonka ensimmäinen julkaisu tapahtui niinkin myöhään kuin tammikuussa 2007. Lyhyessä ajassa VirtualBoxista on kehittynyt hyvin edistynyt ja useita isäntäkäyttöjärjestelmiä tukeva graafinen virtualisointiympäristö. VirtualBox on helppokäyttöinen eikä käytännössä vaadi aiempaa asiantuntemusta. Uusien virtuaalikoneiden luonti on opastettu vaihe vaiheelta, ja edistyneemmille käyttäjille on tarjolla myös tukku lisäasetuksia, joihin voi vielä tarvittaessa vaikuttaa.

VirtualBox soveltuu parhaiten kotikäyttöön sekä esim. ohjelmointiympäristöihin, joissa testejä täytyy ajaa useissa eri käyttöjärjestelmissä tai eristetyissä ympäristöissä. Teknisiltä ominaisuuksiltaan VirtualBox soveltuu myös palvelinvirtualisointiin, mutta käytännössä tähän tarkoitukseen on monipuolisempiakin ohjelmistoja kuten OpenVZ tai Xen.

Nykyään VirtualBoxin omistaa Oracle, jonka vuoksi ohjelman koko nimi on Oracle VM VirtualBox. Se on säilynyt omistajanvaihdoksista huolimatta avoimen lähdekoodin tuotteena.

4.2 FreeBSD Jail

FreeBSD:n mukana kulkeva Jail(8)-ohjelmisto on säiliö-tyyppinen virtualisointiohjelmisto kuten myös seuraavaksi käsiteltävä OpenVZ, mutta nimensä mukaisesti saatavilla ainoastaan FreeBSD-järjestelmissä. Säiliö-tekniikalla tarkoitetaan sitä, että isäntäkone hallinnoi virtuaalikoneita suoraan eikä välissä ole erillistä ohjelmaa eli hypervisoria, kuten VirtualBoxin tai Xenin tapauksessa. Voidaan puhua myös käyttöjärjestelmätason virtualisoinnista, joka tarkoittaa samaa asiaa. Näissä tapauksissa virtuaalikoneita kutsutaan enemmänkin virtuaaliympäristöiksi tai säiliöiksi, mutta teknisesti ne ovat hyvin samanlaisia kuin aidot virtuaalikoneet.

Kukin *jail* eli virtuaaliympäristö asennetaan sille määriteltyyn alihakemistoon, mikä mahdollistaa virtuaalikoneiden muokkauksen myös suoraan isäntäkoneesta. Jail on omiaan erilaisiin testausympäristöihin, mutta ei niinkään prosessi-intensiivisiin virtuaaliympäristöihin sen resurssien hallinnoinnin puutteen vuoksi.

4.3 OpenVZ

OpenVZ vastaavasti on ainoastaan Linux-järjestelmille saatava säiliö-pohjainen, käyttöjärjestelmävirtualisointia hyödyntävä virtualisointiohjelmisto. OpenVZ tukee useampia Linux-julkaisuja nk. mallien (engl. precreated templates) avulla, joiden avulla on mahdollista käyttää mm. Fedora- tai CentOS-julkaisuja, vaikka isäntäkoneessa olisi esim. Debian. OpenVZ:n suurimmat valtit ovat sen hyvin optimoitu suorituskyky sekä mahdollisuus vaikuttaa virtuaalikoneiden resurssien käyttöön, jotta yksikään virtuaalikone ei esim. omi koko prosessoritehoa omaan käyttöönsä.

4.4 Xen

Xen on hieman vanhempi ohjelmisto virtualisointirintamalla, ja sen ensimmäinen versio julkaistiin jo vuonna 2003. Xen on saatavilla ainoastaan Linux- ja UNIX-pohjaisiin isäntäkoneisiin eikä sitä voi ajaa Windowsissa. Se kuitenkin tukee sekä Windows- että

muitakin käyttöjärjestelmiä virtuaalikoneissa versiosta 3 alkaen, kunhan isäntäkoneen prosessorilta löytyy laitteisto-avustettu virtualisointituki. Muissa tapauksissa Xen hyödyntää paravirtualisointi-tekniikkaa, mikä rajoittaa sen saatavuutta eri käyttöjärjestelmäympäristöihin.

Xenissä isäntäkonetta kutsutaan domain 0:ksi (dom0) ja virtuaalikoneita domain U:ksi (domU), mikä saattaa aiheuttaa sekaannusta uusille käyttäjille. Xen tukee resurssien hallinnointia ja jakamista prosessoritehotasolla virtuaalikoneiden kesken, jonka vuoksi se on varteenotettava vaihtoehto Linux- tai UNIX-alustojen virtualisointiohjelmistoksi.

Xenistä on olemassa myös XenServer-niminen versio, joka on erikseen virtualisointia varten räätälöity käyttöjärjestelmä, eikä siten vaadi alleen erillistä Linux- tai UNIX-järjestelmää. Myös XenServer on ilmainen, mutta valitettavasti aika- ja tilarajoitusten vuoksi XenServeriä ei käsitellä tässä työssä.

Xen on nykyään Citrixin omistama, mutta sitä kehittää edelleen The Xen Project ja XenSource Inc.

5 Virtuaalikoneiden asennus ja käyttöönotto

Tämä työ seuraa kronologisesti virtuaalikoneiden asennusta alkaen VirtualBoxiin tehtävien Debian- ja FreeBSD-asennusten myötä. Näiden jälkeen käsitellään vielä Xen-ohjelmiston käyttöönottoa sille tarkoitetulla palvelimessa. Ensin kuitenkin tarvitaan itse VirtualBox-ohjelmisto, joka on saatavissa osoitteessa http://www.virtualbox.org/wiki/Downloads

Tässä tapauksessa VirtualBox asennetaan Windows-koneelle, joten esimerkit ovat sen mukaisia, mutta sivusto tarjoaa tiedostot myös muille käyttöjärjestelmille. Tämän työn kehittyessä VirtualBox ehti päivittyä muutamaankin otteeseen, ja uusin versio kirjoittaessa oli 3.1.8 r61349.

Seuraavaksi tarvitaan Debian- ja FreeBSD-käyttöjärjestelmät. Helpoiten näiden asennus onnistuu hakemalla levykuvat seuraavista osoitteista ja tallentamalla ne koneelle. Kokoa tiedostoilla on n. 650 - 700 MB eli yhden CD:n verran.

64-bittiset versiot:

- http://cdimage.debian.org/debian-cd/5.0.4/amd64/iso-cd/debian-504-amd64-CD-1.iso
- ftp://ftp.fi.freebsd.org/pub/FreeBSD/releases/amd64/ISO-IMAGES/8.0/8.0-RELEASE-amd64-disc1.iso

32-bittiset:

- http://cdimage.debian.org/debian-cd/5.0.4/i386/iso-cd/debian-504-i386-CD-1.iso
- ftp://ftp.fi.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.0/8.0-RELEASE-i386-disc1.iso

Tämän työn puitteissa ei ole merkitystä asennetaanko 64- vai 32-bittinen käyttöjärjestelmä. Jos virtuaalikoneisiin on tarve antaa enemmän kuin 4 gigatavua muistia käyttöön, on järkevää valita 64-bittinen, koska se osaa hyödyntää koko muistialueen.

Asennusta (sekä käyttöä) varten on hyvä, ellei jopa pakollista, olla internet-yhteys, sillä osa tarvittavista asennuspaketeista voidaan ja käytännössä täytyy hakea verkon yli. Tällöin ei ole tarvetta ladata koneelle kaikkia levykuvia, joita on useita kymmeniä

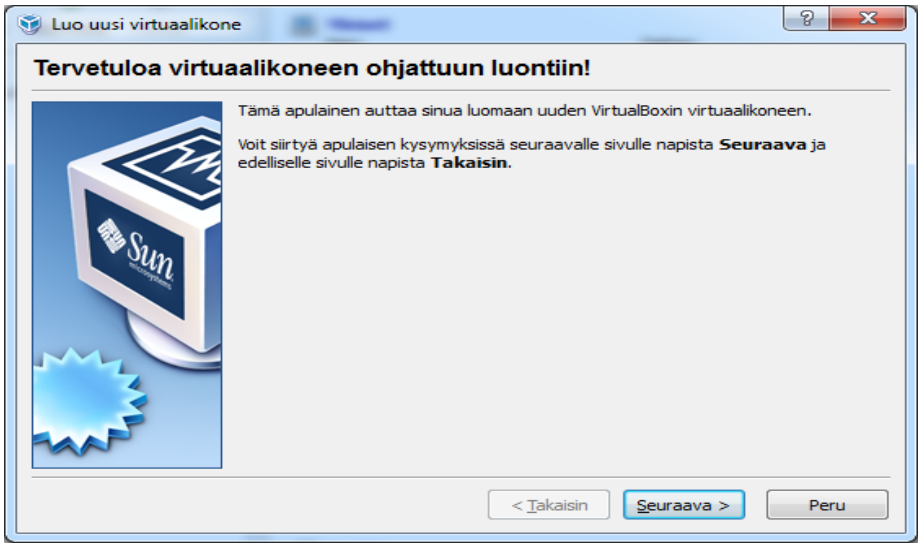
julkaisusta riippuen. Vaihtoehtoisesti asennus voitaisiin tehdä ns. verkkoasennuksena, jolloin kaikki tarvittavat paketit haetaan suoraan verkosta. Tätä varten on olemassa Netinstall- ja Businesscard-levykuvat Debianille. FreeBSD:n asennuksen yhteydessä on mahdollista määrittää myös käytettävä lähde, esim. joko CD/DVD tai verkkoyhteys.

5.1 VirtualBox

Uuden virtuaalikoneen luonti VirtualBoxilla onnistuu parhaiten asennusvelholla, joka käynnistyy ”Uusi”-painikkeen kautta. Asennusvelho käy läpi tarvittavat kohdat virtuaalikoneen luomista varten, jolloin perusasetukset saadaan kohdalleen helposti ja nopeasti.

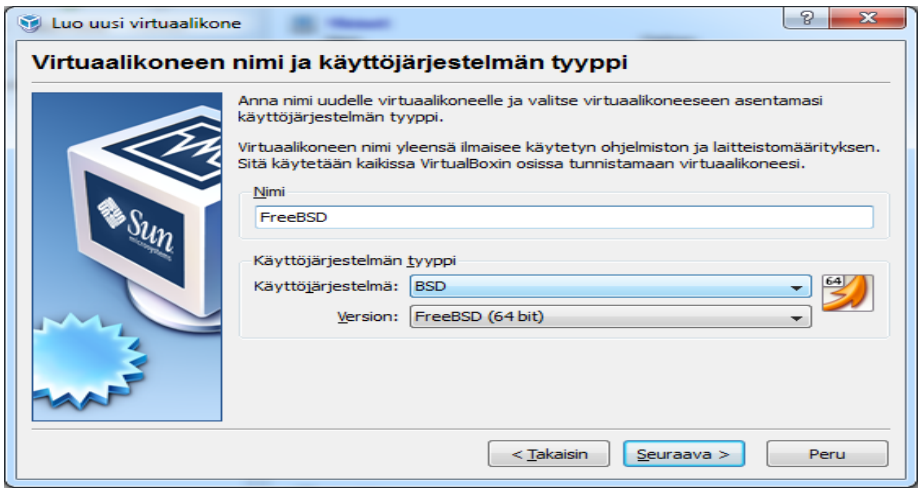
Tarkemmin virtuaalikonekohtaisia asetuksia voi määritellä asennusvelhon valmistuttua, mutta välttämätöntä se ei ole. Seuraavissa kohdissa luodaan virtuaalikoneet FreeBSD- ja Debian-käyttöjärjestelmiä varten.

Asennusvelhon (kuva 1) avulla virtuaalikoneen luonti on käytännössä ”Seuraava”-napin painamista pienin poikkeuksin. Muutokset oletusasetuksista on dokumentoitu ja kuvin varusteltu seuraavissa kohdissa.



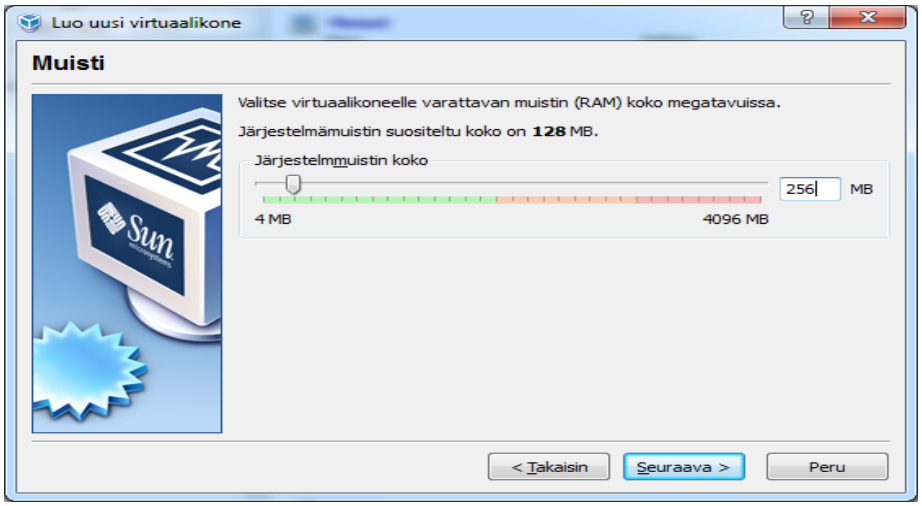
Kuva 1: Asennusvelhon avulla uuden virtuaalikoneen luonti onnistuu helposti.

VirtualBox osaa tarjota yleisimpiä käyttöjärjestelmätyyppejä (kuva 2), jonka mukaan se määrittelee minimivaatimukset muistille ja kiintolevytilalle. Selvyyden vuoksi on hyvä käytäntö nimetä uudet virtuaalikoneet asianmukaisesti, kuten ao. kuvassa.



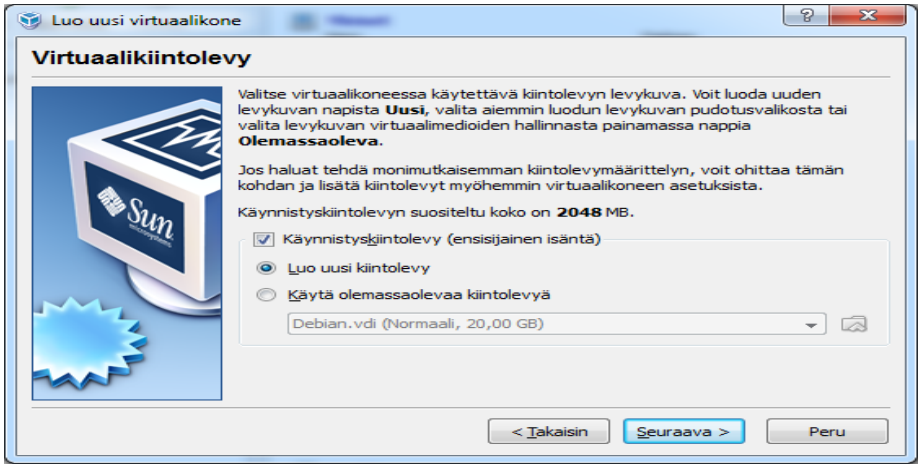
Kuva 2: Määritellään virtuaalikoneen esiasetukset.

FreeBSD-asennukselle VirtualBoxin suosittelema minimi on 128 megatavua, mutta koska tarkoitus on luoda uusia virtuaaliympäristöjä FreeBSD:n sisään, on sille syytä antaa hieman lisää muistia. 256 megatavua (kuva 3) riittää tässä tilanteessa hyvin yhden tai kahden virtuaalikoneen käyttöön. Sama määrä on omiaan myös Debian-asennukselle, mutta enemmänkin voi halutessaan antaa, ja kannattaakin, jos haluaa use-ampia virtuaalikoneita luoda.



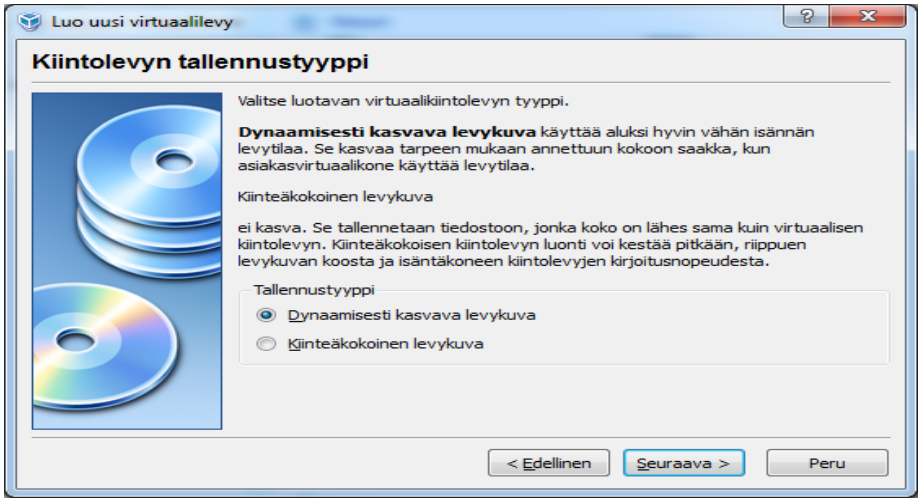
Kuva 3: Virtuaalikoneelle asetettava muistin määrä. 256 megatavua riittää testiympäristölle.

VirtualBox osaa suositella käyttöjärjestelmästä riippuen vähimmäiskokoa kiintolevylle (kuva 4). Asennus on myös mahdollista tehdä jollekin aiemmin luodulle virtuaalikiintolevylle, mutta parempi vaihtoehto on luoda uusi kiintolevy, jolloin yksittäiseen virtuaalikoneeseen tehdyt muutokset on helpompi hallita.



Kuva 4: Virtuaalikiintolevyn tyypin valinta.

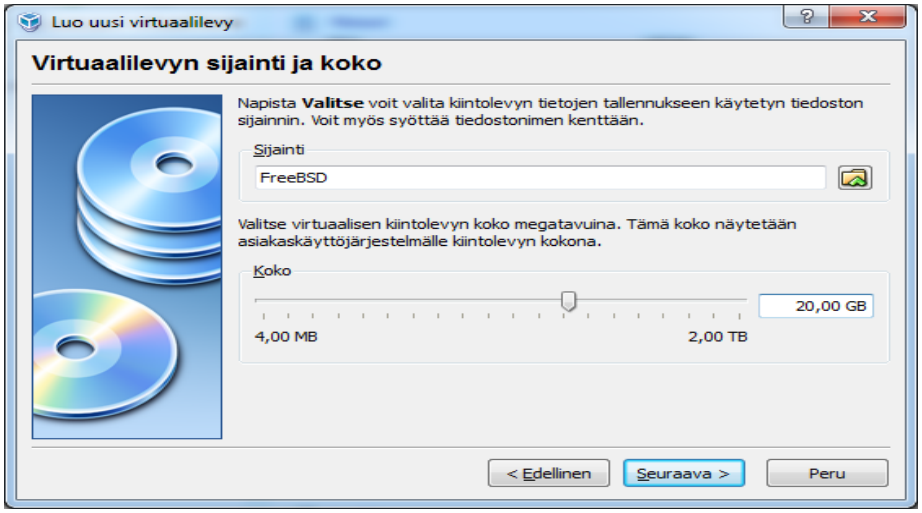
VirtualBox tarjoaa levyn luomiseen kahta eri tapaa: dynaamisesti kasvavaa levykuvaa tai kiinteäkokoista levykuvaa (kuva 5). Dynaamisesti kasvava levykuva on hieman nopeampi luoda asennusvaiheessa eikä se varaa kaikkea levytilaa välittömästi käyttöön. Kiinteäkokoinen levykuva varaa kiintolevyltä koko määrän käyttöönsä saman tien. Dynaamisesti kasvava levykuva on oletusvalinta, ja se on hyvä valinta tämän työn puitteisiin.



Kuva 5: VirtualBox tarjoaa mahdollisuuden käyttää dynaamisesti kasvavaa levytilaa, tai varata kaiken levytilan käyttöön saman tien.

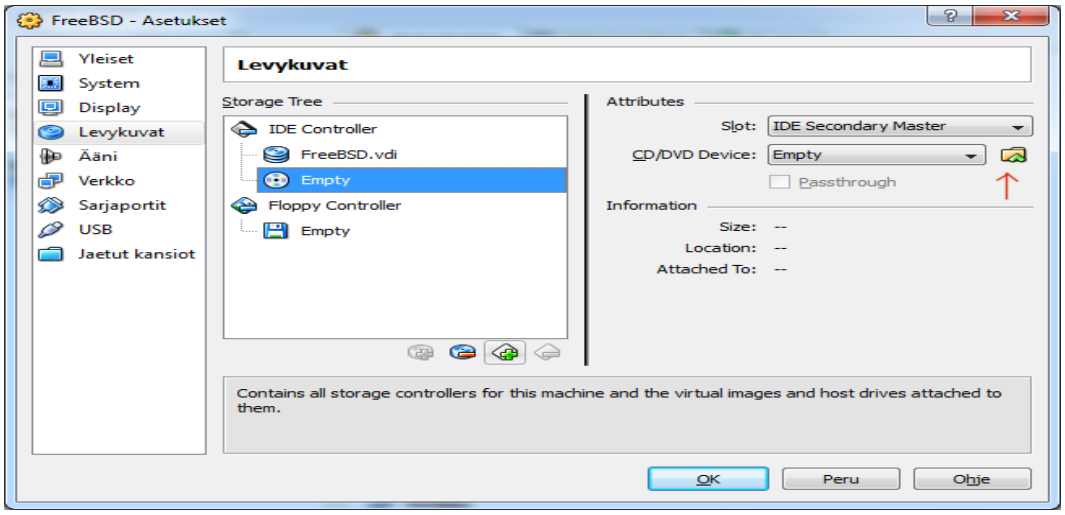
Kiinteäkokoinen levytila on hyödyllinen tilanteissa, joissa halutaan selkeästi rajata virtuaalipalvelimen levykapasiteetti välittömästi, jolloin järjestelmä raportoi aina todellisen jäljellä olevan levytilan.

Viimeisimpänä kohtana valitaan vielä virtuaalilevyn varsinainen koko (kuva 6). Levytilaa kannattaa varata vähintään 20 gigatavua, jotta myöhemmissä vaiheissa ongelmia ei esiinny.



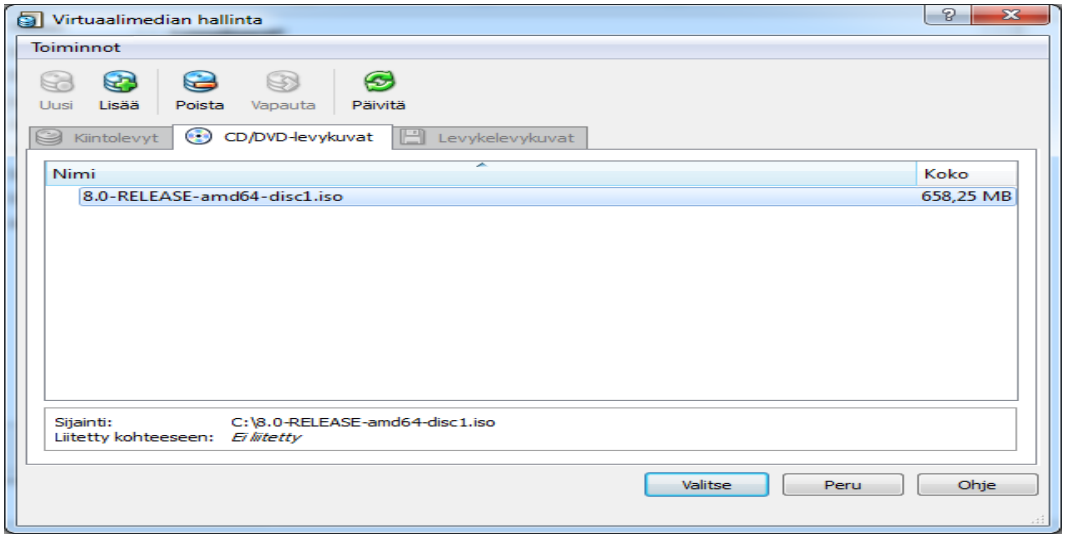
Kuva 6: Levytilan ja tallennussijainnin määrittely.

Kun virtuaalikoneet on luotu, täytyy niihin liittää vielä niille kuuluvat levykuvat, jotta asennus saadaan käyntiin. Punaisen nuolen osoittaman painikkeen takaa (kuva 7) löytyy virtuaalimedian hallinta



Kuva 7: Levykuvan liittäminen virtuaalikoneeseen

Virtuaalimedian hallinnan takaa löytyy lista (kuva 8), jonne voi lisätä levykuvia, joita voidaan käyttää virtuaalikoneissa CD- tai DVD-levyn asemasta.



Kuva *8: FreeBSD:n levykuva liitetty virtuaalikoneen CD-levyksi*

Näiden toimenpiteiden jälkeen virtuaalikoneasennukset voi suorittaa käynnistämällä virtuaalikoneen, valitsemalla käynnistyslähteeksi CD-aseman ja seuraamalla käyttöjärjestelmäkohtaisia asennusohjeita. Seuraavissa kappaleissa on huomioitu joitain yksittäisiä asetuksia, joten varmuuden vuoksi kannattaa lukea eteenpäin vielä ennen asennusten viimeistelyä.

5.2 FreeBSD Jail

FreeBSD:n voi asentaa pääosin asennusohjelman tarjoamin oletusasetuksin. Ainoastaan kannattaa varmistaa, että lähdetiedostot (src) asennetaan myös, jolloin ports-valikoima eli FreeBSD:n mukana kulkeva ohjelmistovalikoima asentuu myös, koska sitä tarvitaan myöhemmissä kohdissa. Mikäli tämän vaiheen ohittaa, on ports-ohjelmistot mahdollista hakea erikseen jälkeenpäinkin *sysinstall*-asennusohjelmalla. Varsinaiseen tuotantokäyttöön asennusta tehdessä asetuksia kannattaa muokata jo tietoturvan ja yleisesti paremman hallittavuuden vuoksi. Joitain vinkkejä käsitellään vielä hieman myöhemmin erillisessä tietoturvaosiossa.

Verkkoasetukset

Oleellista on huomioida, että tässä esimerkissä VirtualBoxin verkkokortti on asetettu NAT (Network Address Translation)-tilaan, jonka vuoksi IP-sarjana käytetään 10.0.2.0-verkkoa erillisen 192.168.1.0-verkon alla, eli käytännössä kaksois-NAT:ia. Alkeellinen kuvaus verkkotopologiasta näyttää suunnilleen tältä:

- ADSL-reititin NAT-tilassa
  - 192.168.1.0-verkko ja VirtualBox-isäntäkone NAT-tilassa
    - 10.0.2.0-verkko ja FreeBSD+virtuaalikoneet

Halutessaan verkkokortin voi laittaa siltaavaan (engl. bridged) tilaan, jolloin FreeBSD voi pyytää IP-osoitteen suoraan esim. kodin ADSL-reitittimeltä, mikä helpottaa mahdollista porttien edelleenohjausta. Tällöin topologiakuvaus muuttuisi seuraavanlaiseksi:

- ADSL-reititin NAT-tilassa
  - 192.168.1.0-verkko ja VirtualBox-isäntäkone sillatussa tilassa
  - 192.168.1.0-verkko ja FreeBSD+virtuaalikoneet

Etuna tässä on se, että nyt sekä isäntäkone että FreeBSD-virtuaalikone ja sen alaiset virtuaalikoneet ovat kaikki samassa verkossa eikä erillisiä reitityssääntöjä tai porttiohjauksia aliverkkoihin tarvitse tehdä. Mutta esimerkin vuoksi tässä asennuksessa käytetään kuitenkin erillistä 10.0.2.0-verkkoa.

Jail(8) ja Ezjail

FreeBSD:n mukana tulee **Jail(8)**-ohjelma FreeBSD:n virtuaaliympäristöjen luontiin ja hallinnointiin, mutta huomattavasti helpommin se onnistuu **Ezjail**-nimisellä lisäohjelmalla, joka on saatavilla FreeBSD:n ports-valikoimasta. Ezjail on erittäin hyvin muokattavissa ja se tarjoaa monia ominaisuuksia, joita kaikkia on mahdotonta käsitellä tässä työssä. Harmillisesti sen oma dokumentaatio on hieman vajanaista, mutta internetissä on runsaasti hyviä resursseja muiden käyttäjien kirjoittamana. Tarkoitus tässä on luoda uusi jail eli virtuaaliympäristö melko perusasetuksin ja osoittaa, että alkuun pääseminen on vaivatonta. Valmis virtuaaliympäristö on oma, eristetty FreeBSD-ympäristönsä omine käyttäjineen ja tiedostojärjestelmineen, jota voi hyödyntää esim. www-ohjelmistojen kuten Apachen tai lighttpd:n pyörittämiseen.

Ezjail löytyy **/usr/ports/sysinstall/ezjail**-hakemistosta (kunhan ports-kokoelma on asennettu) ja se asentuu käskyllä *make install clean*. Asennuksen jälkeen Ezjailin konfiguraatiotiedosto löytyy **/usr/local/etc/ezjail.conf**-hakemistosta, ja sitä muokkaamalla voi mm. vaikuttaa siihen, mihin uudet virtuaalikoneet luodaan sekä tukkuun muita asetuksia, joista manuaalisivu *man ezjail.conf* osaa kertoa paremmin. Ezjail kannattaa myös laittaa käynnistymään automaattisesti koneen käynnistyessä, ja se tapahtuu lisäämällä rivi *ezjail_enable=”YES”* **/etc/rc.conf**-tiedostoon. Tällöin kaikki aktiiviset virtuaalikoneet käynnistyvät automaattisesti.

Asennuksen jälkeen Ezjailille on hyvä tehdä muutama hallinnollinen toimenpide vielä, joista ensimmäinen on ajaa komento *ezjail-admin update -i -p*. Tämä asentaa perussovellukset (i-parametri) ja rakentaa perustan nk. ”basejail”-ympäristölle, jota uudet virtuaalikoneet tulevat hyödyntämään, sekä tarjoaa niille ports-ohjelmistokokoelman käyttöön (p-parametri). On myös mahdollista kääntää tässä vaiheessa koko FreeBSD:n ympäristö uusiksi, jolloin käytetään komentoa *ezjail-admin update -b -p*. Tämä on kuitenkin hyvin pitkällinen toimenpide, ja tuoreessa, vasta-asennetussa järjestelmässä se kannattaa ohittaa, jos mahdollista.

Ezjail asentuu oletuksena **/usr/jails**-hakemistoon, jonka alle myös uudet virtuaalikoneet luodaan. Saman hakemiston alta löytyy myös **flavours**-hakemisto, jonne voidaan määri-

tellä uusien virtuaalikoneiden luonnissa käytettäviä yleisasetuksia eli erilaisia valmiita virtuaalikoneen malleja tai tyyppejä. Kätevä tapa on luoda **/usr/jails/flavours/default/etc**-rakenne, jonne siirretään kaikille virtuaalikoneille yhteiset konfiguraatiotiedostot kuten **resolv.conf** (vaaditaan internet-yhteyksiä varten, löytyy isäntäkoneen **/etc/resolv.conf** -hakemistosta) tai **/etc/localtime** aika-asetuksia varten. Tämän lisäksi täytyy vielä tehdä muutos **/usr/local/etc/ezjail.conf**-tiedostoon ja asettaa default-flavour oletukseksi. Tämän jälkeen **resolv.conf** ja **localtime** kopioituvat automaattisesti kaikkiin uusiin virtuaalikoneisiin. Samaan tyyliin muitakin oletusasetuksia voi lisätä tai muokata halutessaan. Jos käytössä on useita erilaisia tyyppejä, virtuaalikoneen luontivaiheessa on mahdollista määrittää käytettävä tyyppi. Muussa tapauksessa käytetään konfiguraatiotiedoston määrittelemää oletusasetusta.

Ezjail ja verkkoasetukset

Ennen ensimmäisen virtuaalikoneen luontia täytyy sille vielä määrittää IP-osoite verkkoyhteyksiä varten. Tähän on muutamia erilaisia tapoja riippuen järjestelmän asetuksista ja käytössä olevien verkkokorttien määrästä, mutta helpoin tapa yksittäisen verkkokortin tapauksessa on luoda isäntäkoneen verkkokortille nk. alias-osoite eli lisätä sille toinen (tai useampi) IP-osoite. Tämä onnistuu komennolla *ifconfig em0 alias 10.0.2.16 netmask 255.255.255.255*. Lopuksi luodaan testjail-niminen virtuaalikone käskyllä *ezjail-admin create testjail 10.0.2.16*, jonka jälkeen se on käyttövalmis, vaikkakin raakileasetuksin.

Verkkokortin alias-osoite ei säily uudelleenkäynnistyksen läpi, joten se täytyy vielä erikseen lisätä **/etc/rc.conf**-tiedostoon esim. seuraavasti.

echo ’ifconfig_em0_alias0=”10.0.2.16 netmask 255.255.255.255” >> /etc/rc.conf’.

Samalla tapaa onnistuu useampienkin virtuaalikoneiden ja alias-osoitteiden luonti. Ezjail mahdollistaa myös käyttäjätilien ja ryhmien luonnin automatisoinnin, ja esimerkkejä siihen löytyy **/usr/jails/flavours/example**-hakemiston alta. Tässä niitä ei sen tarkemmin eritellä.

Testjail voidaan nyt käynnistää */usr/local/etc/rc.d/ezjail.sh start testjail*-komennolla. Vastaavasti pysäytys toimii vaihtamalla start-sanan tilalle stop. Virtuaalikoneen käynnistyessä sille annetaan tunnistenumero, ja käynnissä olevat virtuaalikoneet sekä niiden IP-osoitteet näkee komennolla *jls*. Jls näyttää tulosteessaan virtuaalikoneiden tunnistenumerot eli ID:t, ja niiden avulla puolestaan onnistuu siirtyminen suoraan virtuaalikoneeseen komennolla *jexec <id> /bin/tcsh*.

Virtuaalikoneen rajoitukset

Vaikka virtuaalikoneet pääasiassa käyttäytyvät kuin aito FreeBSD, on niillä joitain rajoituksia. Muun muassa “raw-socketit” (joita tarvitaan esim. *ping*-komentoon) on oletuksena kytketty pois päältä tietoturvasyistä, ja ne voi kääntää päälle isäntäkoneelta käskyllä *sysctl security.jail.allow_raw_sockets=1*. Tätä ennen virtuaalikoneet täytyy kuitenkin pysäyttää ja käynnistää sen jälkeen uudelleen, jotta asetus tulee voimaan. Tarkemmin isoimpia rajoituksia on käsitelty FreeBSD:n käsikirjassa osoitteessa http://www.freebsd.org/doc/en/books/arch-handbook/jail-restrictions.html.

5.3 OpenVZ

Hieman FreeBSD:n Jailin tyyppinen on myös yleisimmille Linux-jakeluille saatavilla oleva OpenVZ. Kuten FreeBSD:n kanssa, helpointa OpenVZ:aan tutustuminen on VirtualBoxin kautta. Tässä työssä on suosittu Debian-jakelua v5.0.4, koska OpenVZ asentuu vaivatta siihen, ja tuotetuki Debianille on pääsääntöisesti varsin hyvää aktiivisen käyttäjäkunnan puolesta. Debianin asennus on vielä hieman FreeBSD:stä yksinkertaistettu, ja siitä voi suosiolla jättää pois kaiken ylimääräisen kuten tiedostopalvelinkomponentit tai graafista käyttöliittymää koskevat komponentit.

Debianin asennuksessa kannattaa valita GRUB-käynnistyslataajaa, koska se toimii OpenVZ:n kanssa ongelmitta. Samoin on syytä huomioida, että näissä esimerkeissä VirtualBoxin verkkokortti on nyt asennettu siltaavaan tilaan, jolloin Debian pyytää IP-osoitteet suoraan ADSL-reitittimeltä eikä VirtualBoxilta itseltään. Tämä helpottaa virtuaalikoneiden verkottamista myöhemmissä vaiheissa.

Kun Debian on käyttövalmiina, tarjolla olevat OpenVZ-paketit löytyvät suoraan Debianin dpkg-pakettihallinnan kautta, jota voi käyttää esim. Aptitude-ohjelmalla. Käytännössä OpenVZ tarjoaa kaksi erilaista jakelua 32- ja 64-bittisille käyttöjärjestelmille. Tarjontaa on vielä osittain kernel-kohtaisesti, ja parhaiten tällä hetkellä tarjolla olevat paketit näkee komennolla *apt-cache search openvz image*, joista voi valita sopivimman. Tässä työssä asennettu Debian on 64-bittinen, joten on järkevää valita myös OpenVZ:n 64-bittinen versio. Asennus onnistuu komennolla *aptitude install linux-image-2.6.26-2-openvz-amd64*.

Aptitude hakee kaikki tarpeelliset paketit lisäosineen ja asentaa ne automaattisesti, jonka jälkeen Debian on syytä käynnistää uudelleen OpenVZ-kerneliin, jonka pitäisi ollakin oletusvalinta. Kuten FreeBSD:n ja Ezjail:n tapauksessa, OpenVZ vaatii hieman käsittelyä asetuksiinsa, ennen kuin virtuaaliympäristöjä kannattaa lähteä luomaan.

OpenVZ Wiki suosittelee seuraavien muutosten lisäämistä **/etc/sysctl.conf**-tiedostoon (OpenVZ Wiki, 2010):

```
net.ipv4.conf.default.forwarding=1
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter = 1
kernel.sysrq = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
```

Tallennuksen jälkeen sysctl:n uudet asetukset saa käyttöön *sysctl -p* -komennolla. Käytännössä muutokset koskevat isäntäverkon reititystä, jolloin isäntäkone toimii yhdyskäytävänä virtuaalikoneille, jotta ne saavat verkkoyhteyden isäntäkoneen kautta ulkomaailmaan.

Näiden lisäksi Debianin käyttämä iptables-palomuuriohjelmisto vaatii oman NAT-sääntönsä, koska OpenVZ:n virtuaalikoneet käyttävät 192.168.1.0-sarjan NAT-osoitteita tässä esimerkissä. Komento *iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE* on yksi ratkaisu tähän ongelmaan, mutta se suorittaa NAT:n kaikille paketeille eikä esim. vain tietyn IP:n tai IP-verkon paketeille. Tietoturvallisesti se ei ole paras ratkaisu, mutta helpoin tämän esimerkin puitteissa. Helpoin se on myös

siksi, että se toimii kaikkiin luotuihin virtuaalikoneisiin, jolloin iptables-sääntöjä ei tarvitse erikseen muokata jokaista uutta virtuaalikonetta varten. NAT-asetusmuutoksia ei tarvita, mikäli käytössä ovat julkiset IP-osoitteet.

OpenVZ:lla uuden virtuaalikoneen luominen on melko yksinkertaista esiluotujen mallien (precreated templates) avulla. Tässä kohtaa on hyvä huomata, että on myös mahdollista luoda muita kuin Debian-pohjaisia virtuaaliympäristöjä kuten Fedoraa tai Susea. Ajantasainen lista tuetuista käyttöjärjestelmistä löytyy OpenVZ:n wiki-sivustoilta osoitteessa http://wiki.openvz.org/Download/template/precreated.

Valmiin mallipaketin saa haettua esim. wget:llä komennolla *wget http://download.openvz.org/template/precreated/debian-5.0-x86_64.tar.gz*. Huomionarvoista ja tärkeää on se, että paketti täytyy tallentaa **/var/lib/vz/template/cache**-hakemistoon! Tämän jälkeen voidaan luoda Debian 5.0:sta virtuaalikone muutamin komennoin:

vzctl create 101 --ostemplate debian-5.0-x86_64
vzctl set 101 --ipadd 192.168.1.105 --save
vzctl set 101 --nameserver 192.168.1.100 --save
vzctl set 101 --hostname debian_5.0_test --save
vzctl set 101 --onboot yes --save (automaattikäynnistys uudelleenkäynnistyksen jälkeen)
vzctl start 101
vzctl exec 101 passwd (asettaa root-käyttäjän salasanan)

101 on ensimmäinen ID, joka voidaan virtuaalikoneelle antaa, koska sitä aiemmat ovat varattuja. IP-osoite ja nimipalvelin täytyy muokata oman verkon mukaisesti, ja nimeämisen voi luonnollisesti päättää itse. Save-parametri lopussa tallentaa muutoksen pysyväksi. Startin kohdalle voi vaihtaa stop-sanan pysäyttämään virtuaalikoneen.

Kun virtuaalikone on käynnissä, OpenVZ tarjoaa oman terminaalinsa suoraan virtuaalikoneen käyttöön, ja se onnistuu komennolla *vzctl enter <id>*. *exec*-komennolla voidaan käskyttää suoraan virtuaalikonetta ilman erillistä kirjautumista sinne, kuten yo. salasana-esimerkissä. Toinen hyödyllinen komento on *vzlist*, joka listaa kaikki käynnissä

olevat virtuaalikoneet. Paremmin komennot ja niiden parametrit on dokumentoitu vzctl:n manuaalisivuilla.

Tähän asti OpenVZ on ollut hyvin paljon samanlainen kuin Jail, mutta pinnan alta se on hyvin erilainen, monipuolisempi. OpenVZ tarjoaa mahdollisuuden vaikuttaa virtuaalikoneiden resurssienkäyttöön, tai pikemminkin niiden rajoittamiseen. Siinä missä FreeBSD:llä ei ole mahdollista vaikuttaa vaikkapa yksittäisen virtuaalikoneen suorituskykyyn, voi OpenVZ:lla hallita neljää eri pääresurssityyppiä: levykoon rajoituksia, prosessoriajan vuorottelua (CPU scheduler), laiteväylien vuorottelua (I/O scheduler) ja yleistä resurssilaskuria (User Beancounter), jolla voidaan rajoittaa esim. virtuaalikoneen prosessien määrää.

Kun virtuaalikone luodaan, sen levytilarajoitukset voivat olla liian ylä- tai alakanttiin, joten eräs käytännön esimerkki on asettaa virtuaalikoneelle ennalta määritellyt rajat. *vzctl set 101 --diskspace 20000000:30000000 --save* asettaa virtuaalikoneelle 20 gigatavun alarajan ja 30 gigatavun ylärajan. Numeroita käsitellään kilotavuina, jolloin 20000000 kilotavua on yhtä kuin 20 gigatavua. Alaraja on kuitenkin nk. ”soft limit”, jolloin virtuaalikone voi hetkellisesti ylittää alarajan aina määriteltyyn ylärajaan (tässä 30 gigatavua) asti, joka puolestaan vastaavasti on nk. ”hard limit”. Myös ”hetkellisesti”-aikamääreeseen on mahdollista vaikuttaa --quotatime-parametrilla sekuntien tarkkuudella.

Toinen hyödyllinen tapa on rajoittaa prosessoritehoa. Rajoituksia voi jälleen melko tarkkaan asettaa, niin ala- kuin ylärajatkin. Komennolla *vzcpucheck* (-v) saa näkyviin järjestelmän teoreettisen tehon sekä senhetkisen käytön kuin myös sen, mikä virtuaalikone tehoa käyttää (-v-parametrilla). Yksinkertaisimmillaan tuloste näyttää tältä:

Current CPU utilization: 2000 (yksi virtuaalikone käynnissä)
Power of the node: 142288 (koko järjestelmässä saatavilla oleva suoritusteho)

Annetaan virtuaalikone 101:lle puolet saatavilla olevasta tehomäärästä: *vzctl set 101 --cpuunits 71144* (puolet 142288:sta). Nyt *vzcpucheck* näyttää, kuinka virtuaalikone on varannut tehoa käyttöönsä:

Current CPU utilization: 72428
Power of the node: 142288

Käytännössä tämä on huono tapa kuitenkin, koska nyt virtuaalikone saa minimissäänkin 50 % saatavilla olevasta tehosta käyttöönsä, eikä se välttämättä missään välissä edes käytä niin paljoa tehoa. Parempi tapa on asettaa minimiraja alhaiseksi ja maksimiraja johonkin tiettyyn prosenttilukuun, joka voidaan määrittää --cpulimit-parametrilla *vzctl set 101 --cpuunits 2844 --cpulimit 10*. Nyt minimiraja on asetettu n. 2 %:iin ja maksimi 10 %:iin. Tarvittaessa on myös mahdollista vielä määrittää montako ydintä (prosessoria/säiettä) virtuaalikone saa käyttöönsä --cpus-parametrilla. Tätä asetusta kannattaa muuttaa varoen ja ainoastaan siinä tapauksessa, että isäntäkoneesta löytyy neljä tai useampaa ydintä, jolloin voidaan taata esim. kaksi ydintä jonkin virtuaalikoneen käyttöön. Tällöin jäljelle jäävät ytimet jakavat kuormansa tasaisesti eikä mikään virtuaalikone omi kaikkia ytimiä omaan käyttöönsä, mikä yleensä hidastaa koko muunkin järjestelmän etanavauhtiin.

Resurssisääteisyytensä vuoksi OpenVZ sopii mainiosti ympäristöihin, joissa resurssit halutaan jakaa tarkasti ja saada paras mahdollinen lopputulos suorituskyvyn kannalta - etenkin järjestelmissä, joissa ajetaan useita virtuaalikoneita. Eräs käyttötarkoitus voisi olla web-palveluntarjoaja, joka myy eritehoisia paketteja asiakkaan tarpeesta riippuen. Myyntikohteet olisivatkin OpenVZ-virtuaaliympäristöjä, joihin on vain tehty tarkat resurssirajoitukset. Tällöin ei välttämättä tarvitsisi myydä fyysistä palvelinjärjestelmää, vaan asiakkaalle voitaisiin räätälöidä hänen haluamansa kokonaisuus suoraan ohjelmallisesti.

Tässä työssä ei käsitellä erikseen OpenVZ:n tarkimpia resurssi- eikä levyväyläasetuksia, koska ne ovat jo hyvin teknisiä ja vaativat tietoa ja ymmärrystä tietoteknisistä järjestelmistä ja osin jopa ohjelmoinnista. OpenVZ:n wiki-sivusto tarjoaa kuitenkin esimerkkejä näistäkin osoitteessa http://wiki.openvz.org/User_Guide/Managing_Resources.

5.4 Xen

Xen poikkeaa Jailista sekä OpenVZ:sta siinä, että se ei ole säiliö-tyyppinen virtualisointiohjelmisto, vaan samantyylinen kuin VirtualBox eli täysiverinen virtualisointiohjelmisto. Tästä syystä Xen ei toimi VirtualBox:n alla, vaan sitä varten tarvitaan erillispalvelin. Palvelimeksi sopii mikä tahansa uudempi kotitietokone, jonka prosessorilta löytyy tuki virtualisoinnille (eli Intel-VT tai AMD-V). Välttämätöntä se ei ole, mutta ilman sitä käyttöjärjestelmävalikoima on rajoittuneempi, koska virtualisointi toteutetaan paravirtualisoinnilla. Tarkemmin vaatimuksia käsiteltiin alkupuolen Xen-kappaleessa.

Tässä esimerkissä Xen on asennettu Debian-pohjaiseen Linuxiin, jota pyörittää AMD X2 5600+ -prosessori, josta virtualisointituki myös löytyy. Tämä mahdollistaa myös Windows-pohjaiset virtuaalikoneet, vaikka tässä työssä perehdytäänkin ainoastaan Linux- pohjaisiin virtuaalikoneisiin.

Kuten OpenVZ:n tapauksessa, käytössä on Debian 5.0.4 -käyttöjärjestelmä perusasetuksin. Xen löytyy myös suoraan Debianin pakettihallinnasta, mikä tekee asennuksesta hyvin helppoa. Versionumero on kuitenkin jo vanhentunut 3.2.1, kun uusin saatavilla oleva kirjoitushetkellä on jo 4.0. Asennuksen helppouden vuoksi käytetään kuitenkin 3.2.1-versiota, sillä se tarjoaa riittävästi ominaisuuksia tämän työn puitteisiin.

Xen-asennus

Aptitude install xen-linux-system-2.6.26-2-xen-amd64 -käsky asentaa vaadittavat Xen-ohjelmistot, jonka jälkeen kone täytyy käynnistää vielä uudelleen Xenin omaan kerneliin. Tämän jälkeen kannattaa asentaa vielä erilliset työkalut Xenin hallinnointia varten komennolla *Aptitude install xen-tools.* Samoin kuin aiemmatkin virtualisointiohjelmistot, myös Xen vaatii asetusten korjaamista kohdalleen ennen virtualisointiin ryhtymistä.

Xenin konfiguraatiotiedosto löytyy **/etc/xen/xend-config.sxp**-hakemistosta, ja se tarjoaa paljon muokkailtavaa. Alkuun pääsemiseksi tärkein muutos koskee verkkoasetuksia. Muokataan riviä #(network-script network-bridge) siten, että poistetaan risuaita-merkki rivin alusta. Tämän jälkeen isäntäkone osaa luoda ns. sillan verkkoyhteyksiä varten

virtuaalikoneiden välille, jolloin verkkoyhteydet toimivat myös ulkomaailmaan päin virtuaalikoneista.

Konfiguraatioasetukset

Xen-tools asentaa myös oman konfiguraatiotiedostonsa, ja se löytyy **/etc/xen-tools/xen-tools.conf**-hakemistosta. Tähän tiedostoon tehdään oletusasetusmääritykset mm. muistin, IP-osoitteiden ja levytilan suhteen, joten tiedosto kannattaa käydä huolella läpi. Hyvin riittäviä perusasetuksia ovat 4 gigatavun levykoko ja 128 megatavua muistia. Verkkoasetusten helpottamiseksi myös DHCP kannattaa kääntää päälle ja varmistaa, että kyseinen rivi on ”dhcp = 1” ilman kommenttimerkkiä. Järkevää on myös muokata peilipalvelin hakemaan tiedostot suoraan Suomesta: ”mirror = http://ftp.fi.debian.org/debian/”.

Viimeisimpänä, mutta yhtälailla tärkeänä ovat loppupuolella olevat rivit, joiden tulisi näyttää tältä:

```
serial_device = hvc0 #default
# serial_device = tty1
#
disk_device = xvda #default
# disk_device = sda
```

Toinen muutos täytyy tehdä **/etc/inittab**-tiedostoon, jonne lisätään rivit:

```
1:2345:respawn:/sbin/getty 38400 hvc0
2:23:respawn:/sbin/getty 38400 tty1
```

Nämä asetukset koskevat Xenin käyttämää terminaaliyhteyttä virtuaalikoneisiin etenkin Debian-käyttöjärjestelmillä, jolloin syöte ja ulostulo ohjataan oikeisiin terminaaleihin eli käytännössä ikkunoihin.

Näiden asetusten jälkeen kone kannattaa käynnistää vielä kerran uudelleen, jotta kaikki tehdyt muutokset tulevat myös voimaan.

Virtuaalikoneen luonti

Aluksi on hyvä luoda jokin hakemisto, jonne virtuaalikoneet tullaan tallentamaan. Jos Debian asennettiin oletusasetuksin, ei sijainnilla käytännössä ole väliä, koska koko käyttöjärjestelmä on yhtä ja samaa levyosiota. Luodaan esim. **/vms/**-hakemisto *mkdir /vms*, jonne jatkossa luodaan virtuaalikoneet. Virtuaalikoneet luodaan *Xen-tools*-ohjelmalla, joka hakee välttämättömimmät paketit internetistä, muokkaa asetukset käyttäjän määrittelemiksi ja rakentaa konfiguraatiotiedoston, jonka perusteella Xen osaa käynnistää virtuaalikoneen. Koska konfiguraatiotiedostoon on määritelty jo perusasetukset, ainoastaan muutamaan parametriin täytyy kiinnittää huomiota. Seuraava komento luo Debian-virtuaalikoneen.

xen-create-image --hostname=debiantest --dir=/vms --dist=lenny --role=udev

Dist-parametri määrittää käytetyn Debian-version, joista lenny on kirjoitushetkellä uusin. Role=udev-parametri on jälleen Debiania koskeva asetus, joka asentaa udev-palvelun uuteen virtuaalikoneeseen, mikä puolestaan mahdollistaa terminaaliyhteyden suoraan isäntäkoneesta. Lopputuloksena on konfiguraatiotiedosto, joka tallennetaan **/etc/xen**-hakemistoon **debiantest.cfg**-nimellä. Tiedostossa on muutama kohta, joita voi vielä halutessaan muokata.

Eräs ongelma Xenin virtuaalikoneiden kanssa Debianilla on aikasynkronointi, joka usein toimii vähän huonosti ja saattaa täyttää ruudun ”clocksource delta”-viesteillä. Ongelman voi korjata lisäämällä rivin *extra="clocksource=jiffies"* **/etc/xen/debiantest.cfg**-tiedostoon. Lopuksi lisätään vielä rivi *xen.independent_wallclock=1* **/etc/sysctl.conf**-tiedostoon ja ajetaan sysctl -p -komento. Toinen vaihtoehto on asentaa NTP-ohjelmisto virtuaalikoneeseen, jolloin aika synkronoidaan suoraan internetistä ja heittoja ei pitäisi tapahtua.

Virtuaalikone käynnistetään komennolla *xm create debiantest.cfg*, joka käynnistää virtuaalikoneen taustalle. Virtuaalikoneen terminaaliin pääsee suoraan kiinni *xm console debiantest* -komennolla, jolloin voi myös nähdä kuinka se käynnistyy kuin mikä tahansa muukin Debian-järjestelmä. Konsolista pääsee palaamaan Ctrl+]- tai Ctrl+5 -yhdistelmällä. Listan aktiivisista virtuaalikoneista puolestaan saa *xm list* -komennolla,

joka osaa myös näyttää niiden tilan (pysäytetty, käynnissä, jne.) sekä muistin ja prosessorien määrän. Muut xm-pohjaiset komennot on dokumentoitu hyvin manuaalisivuille *man xm*. Yksinkertaisimmillaan virtualisointi Xenillä onnistuu näinkin vaivattomasti, kunhan alun asetusmuutokset vaan saa tehtyä.

Xen tukee myös resurssien rajoittamista, mutta ei niin kehittyneesti kuin OpenVZ. Käytännössä Xen voi hallinnoida prosessoritehoa virtuaalikoneille ja isäntäkoneelle käyttäen joko esiasetettuja rajoituksia tai sitten reaaliaikaisesti. Xenin version 3.2.1 mukana tulee nk. credit scheduler- sekä EDF (Earliest Deadline First)-ajastajat. Näiden suurin ero on se, että credit schedulerissa asetetaan virtuaalikoneille tietyt painotukset, esim. 256 ja toiselle koneelle 512. Tällöin toinen kone saa tuplasti enemmän suoritustehoa eikä muusta järjestelmästä niinkään välitetä. Myös katto voidaan prosentuaalisesti määritellä niin, ettei yksi virtuaalikone voi käyttää kuin maksimissaan esim. 30 % saatavilla olevasta tehosta. EDF-ajastaja taas pohjautuu reaaliaikaiseen seurantaan, jolloin se voi antaa virtuaalikoneille hieman joustoa esim. raskaan yleiskuorman vuoksi. Muuten se toimii samoin painotuksin kuin credit scheduler -mallissa.

Xen on hyvä vaihtoehto OpenVZ:lle mm. jo sen laajemman käyttöjärjestelmätuen vuoksi, mutta suorituskyvyssä se saattaa jäädä hieman jälkeen, mutta normikäytössä eroja todennäköisesti ei huomaa. Loppujen lopuksi kyse on paljolti makuasioista ja tarpeesta: OpenVZ taipuu paremmin säädettäväksi resurssiensa puolesta, mutta on puhtaasti Linux-pohjainen, kun taas Xen tarjoaa laajemmin käyttöjärjestelmätukea, mm. Windowsia tarvittaessa.

6 Virtuaalikoneiden tietoturva

Yleisesti ottaen virtuaalikoneet ovat eristettyjä käyttöjärjestelmiä tai ympäristöjä isäntäkoneesta omine resursseineen, käyttäjineen ja prosesseineen, jolloin esim. tietomurron sattuessa hakkeri ei pääse käsiksi suoraan isäntäkoneeseen vaan saa haltuunsa pahimmassakin tapauksessa ainoastaan virtuaalikoneympäristön. Suurin vaaran paikka on kuitenkin isäntäkoneen saastuminen, sillä isäntäkoneesta on todennäköisesti pääsy kaikkiin virtuaalikoneisiin. Täysvirtualisoinnissa käytettävät hypervisorit vähentävät uhkaa, koska niillä ei ole varsinaista ohjelmistokerrosta, johon hyökkääjä voi päästä käsiksi, mutta eivät poista sitä kokonaan. (Tietokone, 2008)

Virtuaalikone on kuitenkin täysin oma käyttöjärjestelmänsä, ja siihen pätee samat tietoturvariskit kuin fyysiseenkin laitteistoon ja käyttöjärjestelmään, joten virtuaalikoneen tietoturvaan kannattaa suhtautua kuin minkä tahansa muun laitteiston tietoturvaan. Kun virtuaalikoneita on useampia, tulee myös tietoturvan valvominen työläämmäksi. Virtuaalikoneiden virtuaaliset verkot sekä myös hypervisorit ovat ohjelmallisia, ja bugit ohjelmakoodissa saattavat avata tietoturva-aukkoja. Näiltä on hankalampi suojautua muuten kuin pitämällä virtualisointiohjelmistot ja käyttöjärjestelmät päivitettyinä ja ajan tasalla. Palvelimien ja palveluiden hajautus eri virtuaalikoneisiin onkin varteenotettava vaihtoehto niin kuin fyysisellä laitteistolla, eikä esim. SQL-palvelimia aseteta suoraan ulospäin näkyväksi internetiin, vaan sallitaan yhteydet siihen ainoastaan sisäverkon kautta.

Tietoturvaa voi edesauttaa monin tavoin. Vahvat ja pitkät salasanat käyttäjille ja isäntäkoneeseen ainoastaan pääsy niille henkilöille, joiden tarvitsee ylläpitää virtuaalikoneita. Ylimääräiset ohjelmistopalvelut on syytä kääntää pois päältä sekä rajoittaa verkkoyhteyksiä mahdollisimman paljon. (Virtual Machine Security Guidelines, 2007) Toisenlainen suositus verkkoyhteyksien suhteen on se, että vältetään mahdollisimman paljon virtuaaliverkkojen käyttöä ja reititetään virtuaalikoneiden verkkoyhteydet fyysisen laitteen kuten palomuurin kautta, jolloin erilaisia pääsylistoja ja pakettien tutkimista voidaan tehdä liikenteelle helpommin. (Hämäläinen, 2008)

Ulkoisten laitteistojen kuten USB-laitteiden tuki kannattaa kytkeä pois päältä, ellei sille erikseen käyttöä virtuaalikoneessa ole. Kaikkia laitteita emuloidaan ohjelmallisesti, ja kukin ohjelma sisältää aina sen mahdollisuuden, että sen ohjelmakoodissa on jotain, jota hyökkääjä voi hyödyntää. Virtuaalikoneille on myös järkevää olla oma levyosionsa, jotta ne eivät kasvaessaan pääse häiritsemään esim. isäntäkoneen resursseja. Erillinen osio helpottaa myös virtuaalikoneiden ylläpitoa ja siirtämistä esim. isäntäjärjestelmän uudelleenasennuksessa. (Security and Virtualization, 2008)

7 Kehitysideat jatkoa varten

Tätä työtä voi laajentaa käytännössä lähes loputtomiin. Virtualisointiohjelmistojen ominaisuuksista käytiin pääpiirteittäin tärkeimmät lävitse, mutta ne tarjoavat paljon erilaisia työkaluja isommankin virtuaaliympäristön hallitsemiseen. Erilaiset valmismallit, automatisoidut käyttäjien ja esiasennettujen ohjelmistojen ja palveluiden lisäykset sekä migraatiot fyysisistä palvelimista virtuaalisiin ovat kaikki mahdollisia ohjelmistosta riippuen.

Osalla ohjelmistoista virtuaalikoneista voi ottaa nk. snapshoteja eli ikään kuin kuvakaappauksia koko käyttöjärjestelmän tilasta, jolloin niiden tila jäädytetään siihen hetkeen ja tallennetaan levylle. Myöhemmin on mahdollista palauttaa virtuaalikone tuohon tilaan niin kuin mitään ei olisi siinä välissä tapahtunutkaan. Tällaisten ominaisuuksien hyödyntäminen on keskeinen osa nykyistä virtualisointiteknologiaa, ja ne sopivat hyvin varsinkin testiympäristöihin. Luonnollisestikaan järjestelmiä, missä tieto muuttuu jatkuvasti ja replikoidaan muihin palvelimiin, ei ole järkevää lähteä jäädyttämään myöhempää käyttöä varten, sillä tieto vanhentuu nopeasti ja saattaa aiheuttaa tiedon menetystä replikoinnin seurauksena.

Isommissa virtualisointiympäristöissä tulevat kysymykseen myös klusteroinnit eli kuorman ja palvelujen hajauttamiset, joita ei tässä työssä käyty läpi. Myös varmuuskopiot ovat virtuaalijärjestelmien arkipäivää, ja ohjelmistoalustasta riippuen eivät kovin mutkikkaita edes toteuttaa. Virtualisointiympäristön rakentaminen varsinkin yrityskäyttöön ei ole aina ihan yksinkertaista, vaan huomioitavaa riittää lähes jokaisella tietotekniikan osa-alueella.

Kotikäytössä on hyvä harjoitella asioita etukäteen, vaikka joitakin asioita kuten virtuaalikoneen migraatioita ja siirtoja voi olla hankalampi toteuttaa. Tämän työn ohjeilla pääsee alkuun helposti, ja loppu on lähinnä oman mielikuvituksen rajoissa. Virtuaalipalvelimen pystyttäminen kavereille omine virtuaalikoneineen ja tietokantoineen ym. on hyvää harjoitusta, jos vähänkään ylläpitopuoli kiinnostaa. Alkuun pääsee vanhemmallakin tietokoneella, ja itse mm. tein osan tästä työstä käyttäen vanhaa P3 933 MHz:n Debian-palvelintani, jossa on 512 megatavua muistia.

8 Yhteenveto

Virtualisointi saattaa olla käsitteenä hankala hahmottaa, mutta loppujen lopuksi kyse on päällisin puolin hyvin yksinkertaisesta asiasta. Tässä työssä pyrittiin käymään läpi virtualisoinnin eri muotoja, jotta lukija saisi kuvan virtualisoinnista niin historian kuin nykyajankin puolesta. Eri virtualisointitekniikat ovat kehittyneet vuosien varrella ja kehittyvät edelleen jatkuvasti, mutta pääperiaatteet eivät ole muuttuneet, vaikka uusia ominaisuuksia jatkuvasti ohjelmistoihin kehitetäänkin.

Tässä työssä käytiin läpi neljä avoimen lähdekoodin virtualisointiohjelmistoa: VirtualBox, FreeBSD Jail, OpenVZ ja Xen. Tarkoitus oli opastaa virtualisoinnin ensiaskeleet siitä, kuinka kotikäytössä virtualisointi onnistuu helposti VirtualBoxilla ja Windowsilla siihen, miten palvelinpuolen Linux- ja UNIX-ympäristöissä virtualisointi on mahdollista toteuttaa. Virtualisointi tarjoaa niin paljon mahdollisuuksia, että tämän työn puitteissa käytännössä oli mahdollista tutustua niihin ainoastaan pintapuolisesti sekä vähän yrittää tuoda tarjolla olevien ohjelmistojen eroja esiin. Tarkoitus oli olla puolueeton eikä ottaa kantaa siihen, mikä ohjelmisto on paras. Loppujen lopuksi kaikille ohjelmistoille on kuitenkin omat käyttöympäristönsä ja -tarkoituksensa.

Loppupuolella otettiin hieman kantaa virtuaalikoneiden tietoturvaan ja siihen faktaan, että virtuaalikoneisiin tulisi suhtautua kuin mihin tahansa muuhunkin käyttöjärjestelmään tai laitteistoon. Ohjelmat ja tietoturvapäivitykset on syytä pitää ajan tasalla, sekä noudattaa tervettä järkeä ylläpidollisesti mm. käyttäjien salasanojen ja oikeuksien kanssa. Virtuaalikoneissa esiintyy tuon tuosta omia porsaanreikiään, joille loppukäyttäjä ei välttämättä voi mitään, mutta pääasiallisesti ohjelmistot ovat melko tietoturvallisia - ainakin kunhan ne pitää päivitettyinä. Yhtä lailla itse käyttöjärjestelmät on syytä pitää ajan tasalla, ja virtuaalikoneiden kohdalla esim. kloonaukset tai snapshotit tekevät ylläpidosta helpompaa, koska vahingon sattuessa on vanha käyttöjärjestelmäympäristö vaivatonta palauttaa.

Virtualisointitekniikoiden eroja ei välttämättä ole niin tärkeää ymmärtää syvällisesti, vaan yleensäkin tiedostaa, että x86-arkkitehtuurien rajoitusten vuoksi erilaisia ratkaisuja on jouduttu luomaan. Ennen piirivalmistajien mukaantuloa virtualisointi oli hyvinkin

haasteellista paikoin, ja asiat, joita voidaan nykyään pitää itsestään selvinä, eivät aina olleet sitä. Tulevaisuudessa virtualisointi siirtynee entistä enemmän pilvimalliin, jossa yritys voi hankkia palvelinylläpidon ulkoiselta toimittajalta, joka voi myydä palvelun kokonaan virtuaalisena ympäristönä. Puhutaan nk. pilvilaskennasta (engl. cloud computing). Useimmiten tällaiset ympäristöt ovat poikkeuksetta maksullisia ja pohjautuvat johonkin maksulliseen ohjelmistoon kuten VMwareen. Osin siksi myös tässä työssä ei juuri otettu kantaa pilvilaskennan mahdollisuuksiin, koska tarkoitus oli keskittyä avoimen lähdekoodin sovelluksiin. Uusimmissa ohjelmistoversioissa kuitenkin mm. Xen on siirtymässä pilvitekniikkaan, joten tulevaisuudessa tarjontaa tullee löytymään myös avoimen koodin puolelta paremmin.

Kaiken kaikkiaan virtualisointimaailma kehittyy jatkuvasti, ja useampi yritys onkin siihen jo siirtynyt, ja loput seurannevat perässä ennen pitkää. Jatkossa nähdään todennäköisesti myös enemmän ohjelmatason virtualisointia, ja isoimmat verkkolaitteiden valmistajatkin tarjoavat jo omia virtuaalikytkimiään ja -reitittimiään. Virtualisoinnin tulevaisuus on hyvin valoisaa eikä pelkästään palvelinten puolella.

Lähdeluettelo

Big Fat Finance Blog 2009. Virtualization's Darker Side - Five Gotchas. [online] [viitattu 11.4.2010]

http://bigfatfinanceblog.com/2009/12/01/virtualizations-darker-side-five-gotchas/

Gartner Newsroom 2009. Gartner Says 16 Percent of Workloads are Running in Virtual Machines Today [online] [viitattu 11.4.2010]

http://www.gartner.com/it/page.jsp?id=1211813

Golden, Bernard 2007. Virtualization for Dummies. John Wiley & Sons (2007)

Dittner, Rogier 2007. The Best Damn Server Virtualization Book Period. Syngress (2007)

Intel/SharonS-OSM 2009. Roving Reporter: Virtualization - Power of three, or should one prevail? [online] [viitattu 11.4.2010]

http://community.edc.intel.com/t5/Software-Tools-Blog/Roving-Reporter-Virtualization-Power-of-three-or-should-one/ba-p/1453

Intel 2009. Best Practices for Paravirtualization Enhancements from Intel® Virtualization Technology: EPT and VT-d. [online] [viitattu 11.4.2010]

http://software.intel.com/en-us/articles/best-practices-for-paravirtualization-enhancements-from-intel-virtualization-technology-ept-and-vt-d/

Mäkinen, Ville 2009. Virtualisointi teki murron. [online] [viitattu 11.4.2010]

http://www.tietoviikko.fi/kaikki_uutiset/article285493.ece

Kolyshkin Kirill 2006. Virtualization in Linux. [online] [viitatu 11.4.2010]

http://download.openvz.org/doc/openvz-intro.pdf

Kotilainen Samuli 2010. Vihreä it - tietotekniikan tulevaisuus. [online] [viitattu 11.4.2010]

http://www.tietokone.fi/lehti/fallback/vihrea_it_1091

OpenVZ Wiki 2010. Quick Installation [online] [viitattu 2.6.2010]

http://wiki.openvz.org/Quick_installation

The Center for Internet Security. Virtual Machine Security Guidelines [online] [viitattu 25.5.2010]

http://www.cisecurity.org/tools2/vm/CIS_VM_Benchmark_v1.0.pdf

Virtualization.info 2008. Virtualization Industry Challenges - Report 2008. [online] [viitattu 11.4.2010]

http://www.virtualization.info/challenges/

Hämäläinen Pertti 2008. Virtuaalisen uhat ovat todellisia. [online] [viitattu 25.5.2010]

http://www.tietokone.fi/lehti/tietokone_2_2008/virtuaalisen_uhat_ovat_todellisia_978

Vmware 2007. Understanding Full Virtualization, Paravirtualization, and Hardware Assist [online] [viitattu 2.6.2010]

http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

Vmware 2010. Challenges & Obstacles to x86 Virtualization (2010)

http://www.vmware.com/virtualization/history.html

Window Security.com 2008. Security and Virtualization [online] [viitattu 25.5.2010]

http://www.windowsecurity.com/articles/Security-Virtualization.html