Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Markus Ahonen

Suurthiheyksinen käyttöjärjestelmätason palvelinvirtualisointi OpenVZ:llä

Insinöörityö 4.11.2009

Ohjaaja: Matti Puska
Ohjaava opettaja: Matti Puska

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä	Markus Ahonen
Otsikko	Suurtiheyksinen käyttöjärjestelmätason virtualisointi OpenVZ:llä
Sivumäärä	42 sivua
Koulutusohjelma	tietotekniikan koulutusohjelma
Tutkinto	insinööri (AMK)
Ohjaaja	yliopettaja Matti Puska
Ohjaava opettaja	yliopettaja Matti Puska

Insinöörityössä tutustuttiin virtualisointiin yleisesti ja tarkemmin käyttöjärjestelmätason virtualisointiin OpenVZ:llä. Työn tavoitteena oli selvittää voidaanko kyseistä järjestelmää käyttää jopa satojen virtuaaliympäristöjen yhtäaikaiseen suoritukseen, kuinka sitä hallitaan ja voitaisiinko sitä hyödyntää Metropoliassa.

Taustatietojen selvityksen jälkeen OpenVZ asennettiin tavalliseen pöytätietokoneeseen jota kuormitettiin testattaessa http_load ja siege-ohjelmilla vasteaikojen selvittämiseksi.

Tuloksista pääteltiin, että kyseisessä laitteisto- ja ohjelmistokokoonpanossa luvattu olematon virtualisointi-rasite (overhead) ei pidä paikkaansa. Vaikka virtuaaliympäristöjä kyettiin suorittamaan jopa 100 ilman, että vasteajat olisivat olleet sietämättömiä, ei siitä kyetty tekemään yleiselle tasolle vietäviä johtopäätöksiä. Tämän teki mahdottomaksi laitteistokokoonpanojen kirjo ja käytettävien palvelinohjelmien erilaisuudet, pienikin muutos saattaisi muokata tuloksista täysin erilaisia. Tuloksien perusteella voitiin kuitenkin olettaa, että OpenVZ:n rasite on paljon pienempi kuin laitteistoemuloivien virtualisointiratkaisujen. Virtuaaliympäristöjen hallintaa kokeiltiin muutamilla eri työkaluilla, mutta mikään niistä ei osoittautunut kaikenkattavaksi. Lopulta järjestelmää hallinnoitiin kolmella työkalulla. Metropoliassa virtualisointialustalta vaadittiin käyttöjärjestelmäitsenäisyyttä. Tästä syystä koko Metropolian järjestelmiä ei voida korvata OpenVZ:llä. Jotkin vähämerkityksiset palvelut olisi kuitenkin mahdollista toteuttaa järjestelmällä. Hyödyiksi laskettiin vikasietoisuus ja edulliset käyttöönottokustannukset.

Hakusanat	OpenVZ, virtualisointi, virtuaali, vasteaika

Helsinki Metropolia University of Applied Sciences **Abstract**

Author Title Number of Pages Date	Markus Ahonen High-density operating system-level virtualization with OpenVZ 42 4 November 2009
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor Supervisor	Matti Puska, Principal Lecturer Matti Puska, Principal Lecturer

This thesis explores virtualization with the main focus on the operating system-level virtualization platform OpenVZ. The goal was to determine whether it was possible to run hundreds of virtual environments simultaneously, how to manage them and whether it would be possible to apply the solution to Metropolia University.

After the background had been studied, OpenVZ was installed on a regular desktop computer. The system was benchmarked with http_load and siege to determine how response times changed in comparison with the amount of virtual systems.

The results indicated that this particular hardware- and software-combination did not yield near-zero virtualization overhead. Even though the system was able to handle 100 virtual environments without response times becoming inacceptable, there was no way to draw general conclusions. This was due to the fact that hardware- and software-combinations can greatly vary and even a minute change could alter the results drastically. The conclusion was drawn that OpenVZ overhead was far slighter than with systems running hardware emulating virtualization platforms was drawn. The management of the virtual environments was tested with a few tools, however no single tool provided a total solution to the management and eventually the system ran with three tools in use. Metropolia University's environment requires a virtualization platform that supports different operating systems. Therefore it is impossible to replace the entire virtualization system with OpenVZ. Some trivial systems could, however, be implemented with OpenVZ. The advantages were error-redundancy and cheap implementation expenses.

Keywords	OpenVZ, virtualization, virtual, response time

Sisällys

Lyhenteet, käsitteet ja määritelmät

Asiakasohjelmisto	Isäntäohjelmiston alaisuuteen asennettu ohjelmisto virtualisoinnissa (guest). Usein joko käyttöjärjestelmä tai yksittäinen ohjelma.
Papulaskurit (beancounterit)	OpenVZ:n käyttämä sana resurssien laskureille, rajoille ja takuille virtuaaliympäristöä kohden.
CPU-vuorottaja	Järjestelmä, joka päättää, mikä prosessi saa prosessoriaikaa.
dentry	*directory entry*, edustaa yhtä osaa polusta
Emulointi	Tekniikka, jolla imitoidaan jotain tietokonetta tai elektronista järjestelmää toisen tietokoneen tai järjestelmän avulla.
fair-share scheduling	Tietokoneen vuorotusjärjestelmä, jossa prosessoriaikaa jaetaan tasapuolisesti käyttäjien tai ryhmien kesken sen sijaan että se jaettaisiin tasapuolisesti prosessien kesken.
inode	*index node*, tavanomaisessa Unix-käyttöjärjestelmän tiedostojärjestelmässä inode sisältää metadataa – kuten omistajan – tiedostosta, johon se viittaa. Jokaisella tiedostolla on oma uniikki inode-numero, jonka kautta ydin pääsee itse tiedostoon käsiksi.
IP-osoite	Tunniste, jolla yksilöidään verkossa oleva piste johonkin fyysiseen laitteeseen kuten tietokoneeseen.
IPC	*Inter-Process Communication*, tekniikoita, joilla prosessien tai prosessin eri säikeet voivat keskustella keskenään.
Isäntäohjelmisto	Virtualisoinnin kannalta fyysiseen tietokoneeseen asennettu käyttöjärjestelmä (host). Isäntäohjelmiston alaisuudessa suoritetaan asiakasohjelmistot. Asiakasohjelmistoja ei voi olla olemassa ilman isäntäohjelmistoa.
Ydin (kerneli)	Käyttöjärjestelmän ydin. Kerneli on vastuussa käyttöjärjestelmän resursseista eli laitteiston ja ohjelmiston kommunikoinnista.
Linux	Avoimeen lähdekoodiin perustuva käyttöjärjestelmä. Linuxista on olemassa useita eri versioita, joiden ominaisuudet on muokattu erilaisiin palvelin- tai pöytäkonetarkoituksiin. Käyttöjärjestelmästä on olemassa myös suljetun lähdekoodin ratkaisuja.

OpenVZ	Parallels-yhtiön avoimen lähdekoodin käyttöjärjestelmätason virtualisointiratkaisu.
Rasite (overhead)	Tietotekniikassa overheadilla tarkoitetaan mitä tahansa ylimääräistä tai epäsuoraa prosessoriaikaa, muistia, kaistaa tai muuta resurssia vaativaa toimintaa, joka viivästyttää prosessorin toimintaa.
PID	*Process IDentifier*, joissakin käyttöjärjestelmissä käytössä oleva prosessin tunnistustapa. Jokaisella prosessilla on oma uniikki prosessi-numero.
Resurssi	Fyysinen tai virtuaalinen kokonaisuus, jolla on rajoitettu saatavuus.
RPM	*RPM Package Manager*; joissain Linux-käyttöjärjestelmissä käytettävä paketinhallinta työkalu. Järjestelmän kehitti RedHat.
Allekirjoitus (signature)	Matemaattinen skeema, jolla voidaan todentaa dokumentin aitous.
Simulointi	Simuloinnilla tarkoitetaan jonkin todellisen asian, tilan tai prosessin imitoimista.
Mallipohja (template)	Malli, jossa perusasiat ovat valmiiksi määriteltynä.
Virtuaaliympäristö	*Virtual Environment (VE),* eristetty suoritusympäristö. Laitteistoemuloinnissa jokainen asiakasjärjestelmä on oma virtuaaliympäristönsä. Käyttöjärjestelmätason virtualisoinnissa virtuaaliympäristö on koteloitu ja eristetty ohjelma.
VMM	*Virtual Machine Monitor*; laitteistoemuloinnissa isäntäjärjestelmä
VMware	Virtualisointiratkaisuja tarjoava markkinajohtajayritys

1 Johdanto

Virtualisointi on mielenkiintoinen ja jatkuvasti enemmän esille tuleva alue tietotekniikassa. Palvelujen moninaisuus ja näin ollen oman ympäristön tarve asettavat omat vaatimuksensa laitteistolle. Toisaalta taas palvelujen samankaltaisuus on luonut tarpeen laitteiston paremman käyttöasteen saavuttamiselle. Moninaisuuteen vastaa hyvin laitteistoemuloiva virtualisointi. Samankaltaisilla palveluilla järjestelmä on mahdollista jakaa kevyesti kaikkien kesken, tällaista tarvetta palvelee käyttöjärjestelmätason virtualistointi.

Tässä insinöörityössä perehdytään käyttöjärjestelmätason virtualisointiin avoimeen lähdekoodiin perustuvalla OpenVZ:llä. Työssä selvitetään virtualisoinnin perusteet ja keskitytään OpenVZ:n tarjoamiin erikoisuuksiin. Työssä läpikäydään sekä teoria että yksinkertaisin tapa asentaa järjestelmä. Työn tarkoituksena on selvittää, voiko yhdessä laitteistossa suorittaa yhtäaikaisesti jopa satoja virtuaalipalvelimia ja minkälaiset resurssivaatimukset yhtäaikainen suoritus aiheuttaa laitteistolle. Kuinka tällaista palvelinrypästä hallinnoidaan? Mitä mahdollisuuksia tällaisella järjestelmällä on tulevaisuudessa ja pystyttäisiinkö järjestelmää hyödyntämään Metropoliassa? Laitteistona työssä käytetään tavanomaista pöytätietokonetta, johon on asennettu RPM-pohjainen Linux-käyttöjärjestelmä.

Käyttöjärjestelmätason virtualisoinnilla saavutetaan teoreettisesti huomattavasti parempi tehokkuus kuin tavanomaisilla virtuaalikoneilla. Toisaalta käytettäessä OpenVZ:aa täytyy sekä isäntä- että asiakaskoneen olla varustettu Linux-käyttöjärjestelmällä.

2 Virtualisointi

“Virtualisointi”-termi keksittiin 1960-luvulla tarkoittamaan virtuaalista laitetta. Sillä tarkoitetaan tekniikkaa, jolla jokin fyysinen resurssi piilotetaan muilta järjestelmiltä, sovelluksilta ja loppukäyttäjältä. Fyysisellä resurssilla tarkoitetaan esimerkiksi muistia

tai prosessoria. Fyysisen resurssin sijaan näytetään emuloitu resurssi, jonka tarkoitus on imitoida oikeata resurssia. Tällä tavoin yksi fyysinen resurssi voi toimia monena loogisena resurssina. Samoin useampi fyysinen resurssi voi toimia yhtenä loogisena resurssina. Looginen resurssi viittaa fyysiseen resurssiin ja kuvaa sen ilmentymää käyttäjälle. [1.]

Virtualisointi suoritetaan isäntäohjelmalla, joka luo simuloidun ympäristön asiakasohjelmistoa varten. Simulointi on jonkin todellisen asian imitoimista. Usein asiakasohjelmistoa ei ole rajoitettu käyttäjäohjelmiin, vaan jopa käyttöjärjestelmiä pystytään suorittamaan virtuaalisessa ympäristössä. Asiakasohjelmisto kuvittelee olevansa suoraan yhteydessä fyysiseen laitteistoon. [1.]

Käyttöjärjestelmä itsessään on eräänlainen virtuaaliympäristö. Useiden ohjelmien ollessa käynnissä samassa tietokoneessa käyttöjärjestelmä huolehtii siitä, kuinka laitteistoresursseja käytetään ja jaetaan. Käyttöjärjestelmä koteloi laitteistoresurssit niin, että usea ohjelma pystyy käyttämään niitä yhtäaikaisesti. [2, s. 10.]

2.1 Virtualisoinnin tarve

Suurin yksittäinen syy virtualisoinnille on raha. Virtualisoinnilla saatetaan jo pienissäkin ympäristöissä saavuttaa huomattavia säästöjä. Laitteistokustannuksissa säästetään, koska fyysisiä laitteita tarvitaan vähemmän. Toisaalta myös käyttöjärjestelmien lisenssejä ei useinkaan tarvita kuin yksi keskusyksikköä kohden.

Tietokoneiden huoltaminen on aikaa vievää ja kallista. Yhdessä fyysisessä laitteessa olevien useiden virtuaalipalvelimien hallinta on huomattavasti helpompaa. Viallisten osien vaihtaminen, päivitysten asentaminen, tehokkuuden seuranta ja varmistukset vaativat vain vähän työaikaa kun kyseessä on yksi laite. [3, s. 6.]

Virtualisointi vähentää myös tilantarvetta. Palvelinhotellin pitäminen ilman virtualisointia vaatisi nopeasti valtavan tilan. Lisäksi yksittäiset erikoissovellukset vaativat usein oman palvelimen. Tällaisen tilan ylläpitämisestä aiheutuisi massiiviset

sähkökustannukset, ja tilan jäähdytys olisi haastavaa. Ensimmäinen yksinkertainen askel järkevään yhtenäistämiseen on inventoida palvelimet, joissa suoritetaan yksittäistä erikoissovellusta. Virtualisoinnilla fyysisten palvelimien tarve voidaan mahdollisesti pudottaa pieneen osaan alkuperäisestä. [4.]

Käytön maksimointi liittyy palvelinten yhtenäistämiseen. Palvelimet saattavat usein käyttää vain muutamia prosentteja prosessoritehosta. Kun yksittäistä sovellusta palvelevat palvelimet kerätään yhteen, tulee laitteiston käytöstä tehokkaampaa. [4.]

Virtuaalikoneet tarjoavat erinomaisen kehitysympäristön [2, s. 177]. Koska virtuaalikone saadaan emuloimaan oikeaa ympäristöä, voidaan esimerkiksi ohjelmistopäivitykset ensin testata tällaisessa ympäristössä ja vasta toimivuuden toteamisen jälkeen aloittaa päivityksen yleinen jakelu. Käytännössä kaikenlaiset testaukset, joissa halutaan käyttää tuotantoympäristön kloonia ilman vaaraa, voidaan toteuttaa virtuaalikoneilla.

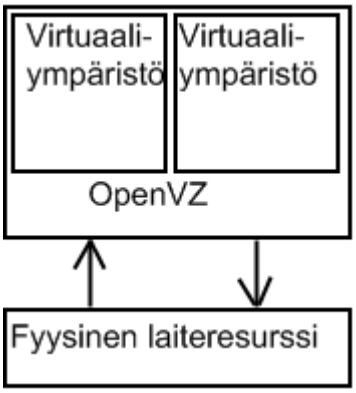
Tarpeen vaatiessa uuden virtuaalikoneen teko on helppoa verrattuna fyysisen laitteen hankintaan. Fyysisen laiterikon tapahtuessa virtuaalikoneen siirto onnistuu nopeasti ja vaivattomasti ja vikatilanteesta palautuminen on ripeää. [2, s. 70.]

2.2 Virtualisointitavat

Virtualisointiin on olemassa useita eri lähestymistapoja. Kaksi tyypillisintä virtualisointitapaa ovat palvelinvirtualisointi ja levytilavirtualisointi [5]. Palvelinvirtualisoinnilla yksittäisellä palvelimella voidaan suorittaa useita toisistaan eristettyjä virtuaalipalvelimia. Levytilavirtualisoinnilla tarkoitetaan loogisen tilan erottamista fyysisestä. Seuraavissa luvuissa tutustutaan kahteen erilaiseen palvelinvirtualisointitapaan. OpenVZ on käyttöjärjestelmätason virtualisointialusta. Alan markkinajohtajan, VMwaren, desktop-tuote on laitteistoa emuloiva virtualisointialusta.

2.2.1 Käyttöjärjestelmätason virtualisointi

Käyttöjärjestelmätason virtualisointi tapahtuu olemassa olevan käyttöjärjestelmän päällä. Usein näitä virtuaalikoneita kutsutaan ”säiliöiksi”. Virtualisointiohjelmisto tarjoaa kirjastot, joiden avulla ohjelmistot kommunikoivat käyttöjärjestelmän kanssa. Tällä luodaan illuusio siitä, että laite on omistettu yksinomaan kyseiselle ohjelmistolle. Ohjelmisto näkee vain omassa virtuaaliympäristössään olevat muut ohjelmistot ja kommunikoi vain oman virtuaalisen käyttöjärjestelmänsä kanssa. Ohjelmisto ei ole tietoinen muista samassa laitteistossa suoritettavista virtuaaliohjelmistoista. Tämä tekniikka toimii hyvin silloin, kun kaikki virtualisoitavat ohjelmistot toimivat kyseisen käyttöjärjestelmän päällä luonnollisesti. Virtualisointiympäristön ja fyysisen resurssin välinen kommunikointi havainnollistuu kuvassa 1.



Kuva 1. Kommunikointitapa käyttöjärjestelmätason virtualisoinnissa.

Tyypillisesti käyttöjärjestelmätason virtualisointia käytetään webhotelleissa. Jokainen webpalvelin-prosessi suoritetaan omassa säiliössään, jolloin se luulee omistavansa koneen kaikki resurssit, vaikka todellisuudessa tällaisia säiliöitä saattaa olla useita kymmeniä samassa laitteistossa. [3, s.7.]

Käyttöjärjestelmätason virtualisoinnista aiheutuu hyvin vähän prosessorikuormaa, koska laitteistoresursseja ei tarvitse jyvittää säiliöille. Näin ollen säiliöille jää enemmän resursseja käytettäväksi. [3, s. 7.]

Säiliöinti-tekniikalla toteutettu virtuaaliympäristö kärsii kuitenkin suuremmista ohjelmistorajoitteista kuin myöhemmin esitelty laitteistoemulointi. Käyttöjärjestelmä täytyy valita tarkasti, jotta säiliöissä suoritettavat ohjelmat toimivat. Mikäli säiliöissä suoritetaan eri ohjelmia voi tästä syntyä ongelmia. Usein ohjelmat vaativat toimiakseen jonkin tietyn version käyttöjärjestelmästä ja jopa tietyt käyttöjärjestelmä- ja ohjelmistopäivitykset. Tällöin laitteiston hallinnoinnista tulisi sietämättömän hankalaa. [3, s. 8.]

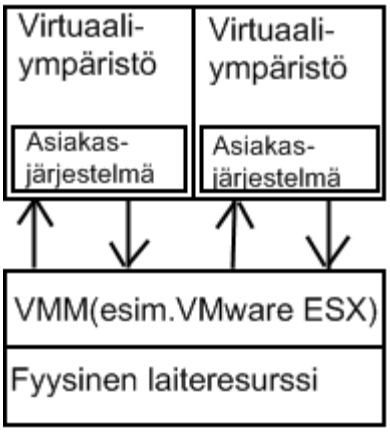
2.2.2 Laitteistoemulointi

Laitteistoemuloinnilla tarkoitetaan sellaista virtuaalista järjestelmää, jossa laitteistokokonaisuus simuloidaan täydellisesti. Täydellisellä simuloinnilla tarkoitetaan tilannetta, jossa virtuaalikone simuloi isäntäkoneen laitteistokokonaisuuden. Jos ohjelmisto voidaan suorittaa kyseisellä laitteistokokoonpanolla, voidaan se suorittaa virtuaalikoneessa riippumatta siitä, voitaisiinko se suorittaa isäntäkoneen käyttöjärjestelmässä. Muut virtualisointitekniikat mahdollistavat vain tiettyjen tai muokattujen ohjelmistojen suorituksen virtuaalikoneessa. Tyypillisesti isäntäohjelmistoa kutsutaan *VMM*:ksi eli Virtual Machine Monitoriksi tai *hypervisor*iksi, sen vastuulla on huolehtia kaikesta, mitä virtuaalikoneiden sisällä tapahtuu, ja joko antaa tai kieltää pääsy johonkin resurssiin. [6.]

VMM:t on jaoteltu kahteen luokkaan. Tyypin II VMM:t ovat sellaisia, joita suoritetaan jonkin käyttöjärjestelmän päällä, ja ne suorittavat korkeamman tason virtuaalikoneita. Esimerkiksi Java- ja .NET-ympäristöt ovat tyypin II VMM:eja. Tyypin I VMM:t toimivat ilman isäntäkäyttöjärjestelmää. Ohjelmisto asennetaan joko suoraan laitteistoon, tai ohjelmisto on sulautettu laitteistoon. Esimerkiksi VMware ESX on tyypin I VMM. [7.]

VMM tarjoaa standardoidun laitteistoympäristön, jonka kanssa asiakasjärjestelmä voi kommunikoida ja jonne se on asennettu. VMM ja asiakasjärjestelmä muodostavat yhdessä siirrettävän paketin. Paketti voidaan siirtää jopa tietokoneeseen, jonka laitteisto eroaa alkuperäisestä. Edellisen kaltaisessa tilanteessa hypervisor tulkitsee VMM:n

kutsut laitteistolle sopiviksi. [3, s. 8.] Kuva 2 havainnollistaa tyypin I VMM:n kommunikoinnin fyysisen resurssin kanssa.



Kuva 2. Tyypin I VMM:n kommunikointikaavio

Laitteistoemuloinnilla voidaan saavuttaa tosiasiallisesti eristetyt asiakasjärjestelmät. Laitteistoemulointi tukee sekä useiden käyttöjärjestelmien käyttämistä että useiden eri versioiden käyttämistä samasta käyttöjärjestelmästä. Käyttöjärjestelmät saattavat poiketa toisistaan vaikkapa vain yhden päivityksen verran. [3, s. 8.]

Suurin hyöty laitteistoemuloinnin kaltaisesta virtualisoinnista saadaan ohjelmistokehityksessä ja laadunvarmistuksessa. Kehitettyä ohjelmistoa voidaan nopeasti testata useilla eri käyttöjärjestelmäversioilla ja käyttöjärjestelmillä. Erilaisten laitteistoympäristöjen testaus on myös helppoa, koska testauksen kannalta merkittävä komponentti tarvitsee vaihtaa vain yhteen tietokoneeseen. [3, s. 9.]

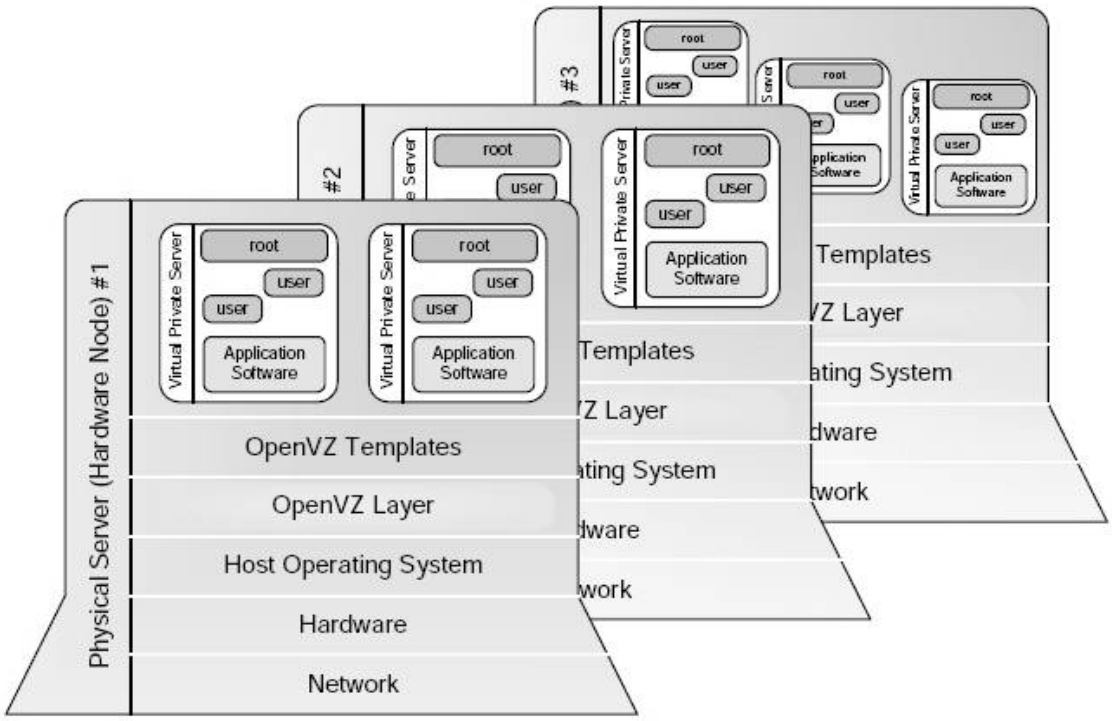
Laitteistoemulointi on raskasta. Usein asiakasjärjestelmän suorituskyky on huomattavastikin heikompi, kuin jos järjestelmää ajettaisiin suoraan tietokoneessa. Toinen haittapuoli laitteistoemuloinnissa on laitteistoajurien päivitysten vaikeus. Laitteistoajurit sijaitsevat joko VMM:ssä tai hypervisorissa, joten niitä ei voida päivittää tavanomaisella tavalla. Mikäli tietokoneessa on jokin resurssi, johon virtualisointisovelluksella ei ole ajuria, ei virtualisointisovellusta voida suorittaa. [3, s. 9.]

3 OpenVZ

Käyttöjärjestelmätason virtualisointisovellus OpenVZ on modifioitu Linux-ydin. Ytimeen on lisätty tuki virtualisoinnille ja eristämiselle, resurssien hallinnalle ja tilan tallennukselle. [8, s. 2.] Linux-ytimen tehtävä on yhdistää ohjelmistot laitteistoon.

3.1 Perusteet

OpenVZ luo useita turvallisia ja eristettyjä säiliöitä (virtuaaliympäristöjä) yhdelle fyysiselle palvelimelle. Ratkaisulla saadaan palvelimelle parempi käyttöaste ja taataan, että ohjelmistot eivät ole ristiriidassa. Jokainen säiliö toimii aivan kuten erillinen palvelin. Säiliö voidaan käynnistää uudelleen, sillä voi olla omat käyttäjät, IP-osoitteet, muisti, prosessit, tiedostot, ohjelmat, systeemikirjastot ja konfigurointitiedostot. [8, s. 2−3.] OpenVZ-arkkitehtuuri on havainnollistettu kuvassa 3. Kuvassa oletetaan, että fyysisiä palvelimia on useita, mutta yhtä lailla niitä voisi olla vain yksi. Fyysistä palvelinta kutsutaan OpenVZ maailmassa Hardware Nodeksi.



Kuva 3. OpenVZ-arkkitehtuuri [9, s. 15].

OpenVZ on avoimen lähdekoodin sovellus GNU GPL:n alaisuudessa. Parallelsin Virtuozzo Containers on kaupallinen projekti, jonka pohjana OpenVZ on. Parallels tukee myös OpenVZ:aa. [10.] GNU GPL määrittelee alaisensa tuotteet ja niistä johdetut vapaasti levitettäviksi.

Virtualisointia säiliöimällä ei kuitenkaan sanota todelliseksi virtualisoinniksi. Virtualisointialustan päällä toimivat sovellukset paketoidaan omiksi erillisiksi säiliöiksi. Tästä syystä järjestelmä on rajoitetumpi kuin laitteistoemuloivat järjestelmät. Samasta syystä järjestelmä on teoriassa huomattavasti nopeampi, koska virtualisoinnista ei aiheudu samaa rasitetta (overhead). Virtualisoinnissa rasitteella tarkoitetaan prosessorin emuloinnista aiheutuvaa kuormaa ja eri ympäristöjen välillä vaihtelusta aiheutuvaa muistin tyhjennys- ja täyttö- operaatiota.

OpenVZ voidaan asentaa joko RPM-pohjaiseen tai Debian-pohjaiseen Linuxiin. RPM-pohjaisia Linuxeja ovat muun muassa Fedora, Red Hat Enterprise Linux ja CentOS. RPM on eräs Linuxin paketinhallintajärjestelmä. Debianin Etch ja Lenny ovat tuettuja. OpenVZ paketin mukana tulee ainoastaan muokattu ydin (kerneli). Toimintaa varten tarvitaan vielä käyttäjätason työkalut ja säiliön mallipohja (template cache). Mallipohjan voi tehdä myös itse. [11.] Mallipohja on lista paketeista, jotka tarvitaan virtuaaliympäristön luomiseen.

3.2 Ytimen lisäominaisuudet

Muokatun Linux-ytimen lisäominaisuudet ovat tuki virtualisoinnille ja ympäristöjen eristäminen, resurssien hallinta ja ympäristön tilan tallennus [8, s. 2].

3.2.1 Virtualisointi ja eristäminen

Virtualisointituen avulla yksi ydin saadaan ajamaan montaa virtuaaliympäristöä. Jokaisella virtuaaliympäristöllä on oma prosessipuunsa. Virtuaaliympäristön PID-numerot alkavat numerosta 1 aivan kuten aidossakin järjestelmässä. PID-numero on

käyttöjärjestelmän prosessille antama yksilöllinen tunnistusnumero jonka kautta siihen voidaan viitata. Sillä on oma tiedostojärjestelmä erillään isäntäjärjestelmän tiedostojärjestelmästä. Ympäristölle luodaan myös omat virtuaaliset laitteet. Verkkoadapterit, IP-osoitteet ja reitityspöydät ovat erillisiä. Jopa IPC-objektit ovat virtuaaliympäristön omia. IPC on joukko tekniikoita, joilla voidaan jakaa tietoa eri prosessien tai säikeiden välillä. [8, s. 2−3; 12.]

3.2.2 Resurssien hallinta

Resurssien hallinta on asia joka erottaa OpenVZ:n muista samantyyppisistä käyttöjärjestelmätason virtualisointiratkaisuista. Muokattuun ytimeen on lisätty käyttäjän papulaskurit (beancounterit), kaksitasoinen levykiintiöjärjestelmä ja tasapuolinen CPU-vuorottaja.

Resurssien hallinta on erittäin tärkeää käyttöjärjestelmätason virtualisointiratkaisuissa. Yhden ytimen resurssit ovat rajalliset ja ne pitää jakaa useiden virtuaaliympäristöjen kesken. Resurssien jakoa tulee hallita niin, että virtuaaliympäristöt voivat tulla toimeen toistensa kanssa eivätkä vaikuta toisiinsa. [8, s. 3.]

Käyttäjän papulaskurit

Jokaiselle virtuaaliympäristölle määriteltäviä resurssilaskureita, -rajoitteita ja -takuita kutsutaan käyttäjän papulaskureiksi. Papulaskureiden avulla estetään joltain virtuaaliympäristöltä isäntäkoneen resurssien omiminen. [8, s. 3.] Jokainen papulaskureilla määriteltävä resurssi on nähtävissä kansiossa /proc/bc/, ja jokaiseen on assosioitu viisi arvoa: tämänhetkinen käyttö (current usage), maksimikäyttö (maximum usage), väliraja (barrier), loppuraja (limit) ja epäonnistumislaskuri (fail counter). Mikäli joku resurssi ylittää välirajan, kasvatetaan epäonnistumislaskuria, jonka avulla järjestelmän hallinnoija pystyy havaitsemaan tulevat ongelmat. Kansion sisällä on jokaisella virtuaalikoneella oma hakemistonsa, jonne tallennetaan kyseisen virtuaalikoneen papulaskurit ja niihin liittyvät tiedot. [12.] Taulukossa 1 esitellään osa arvoista, joille yllä mainitut viisi arvoa voidaan määritellä.

Taulukko 1. Papulaskureita

Arvo	Merkitys
lockedpages	Muistin määrä sivuina, jota ei saa siirtää virtuaalimuistiin.
shmpages	Jonkin VE:n jaetun muistin kokonaismäärä sivuina.
privvmpages	Ohjelman omaan käyttöön varatun muistin määrä.
numfile	Kaikkien VE-prosessien avoimien tiedostojen yhteenlaskettu lukumäärä.
numflock	Kaikkien VE-prosessien luomien tiedostolukitusten lukumäärä.
numproc	VE:n sisäisten prosessien maksimilukumäärä.
numpty	Epäaitojen (kuten ssh-sessiot) terminaalien lukumäärä.
numsiginfo	Siginfo struktuurien lukumäärä. Parametrillä säädetään signaalijonon kokoa.
dcachesize	Muistiin lukittujen dentry- ja inode-struktuurien koko.
physpages	VE:n sisäisten prosessien käyttämän muistin kokonaismäärä.
numiptent	IP-pakettien filtteröintimerkintöjen lukumäärä.

Levykiintiöt

Jokaisella virtuaaliympäristöllä voi olla oma levykiintiö. Kiintiö mitataan levyblokkeina ja inodeina, joilla tarkoitetaan tiedostojen lukumäärää. Virtuaaliympäristön sisällä voidaan käyttää Linuxin standardeja käyttäjä- ja ryhmäkohtaisia levykiintiöitä. [8, s. 3; 12.]

Tasapuolinen CPU-vuorottaja

OpenVZ:n tasapuolinen CPU-vuorottaja on kaksitasoinen toteutus Fair-Share Schedulingista. Ensin vuorottaja päättää, mille virtuaaliympäristölle annetaan aikajakso. Päätös perustuu ympäristölle annettuun cpuunits-arvoon. Toiseksi Linux-käyttöjärjestelmän oma vuorottaja päättää, mille prosessille ympäristössä vuoro annetaan. Tähän vaikuttavat tavanomaiset prosessien prioriteetit. [8, s. 3; 12.]

3.2.3 Tilan tallennus

Virtuaaliympäristö voidaan jäädyttää ja tallentaa levylle kokonaisena. Tällöin koko ympäristö voidaan siirtää toiseen tietokoneeseen ja palauttaa toimintaan. Koko prosessi vie vain muutaman sekunnin eikä se asiakkaan näkökulmasta näytä palvelun

tavoittamattomuutena. Jäädyttäminen tallentaa myös avoinna olevat verkkoyhteydet, joten siirto vaikuttaa vain viiveeltä. [8, s. 4; 12.]

3.3 Mallipohjat

Mallipohja on OpenVZ:ssä virtuaaliympäristön rakennuspalikka. Käyttöjärjestelmä-mallipohja on joukko paketteja, joita tarvitaan virtuaaliympäristön ajamiseen. Usein mallipohjat luodaan suoraan hardware nodeen. Luomiseen tarvitaan vain työkalu (vzpkg) ja mallipohjan metadata. [8, s. 4.]

Metadata on tietoa jostakin käyttöjärjestelmämallipohjasta. Mallipohjan metadata sisältää neljä asiaa: lista paketeista, jotka sisältyvät kyseiseen mallipohjaan, verkossa olevien pakettien tallennuspaikkojen sijainnin, jakelukohtaiset komentojonot jotka täytyy suorittaa templaten asennuksen yhteydessä sekä julkiset gpg-avaimet joilla voidaan tarkastaa pakettien allekirjoitus (signature) eli aitous. Pakettien tallennuspaikasta voidaan hakea ja asentaa ohjelmistopaketteja. Metadatatieto on muutamissa tiedostoissa, jotka löytyvät kansiosta /vz/template/<käyttöjärjestelmän nimi>/<käyttöjärjestelmän julkaisu>/config/. Esimerkiksi metadata Fedora 11-järjestelmälle löytyy kansiosta /vz/template/fedora/11/config/. [9, s. 17.]

Mallipohjan metadata sisältää tarvittavan tiedon käyttöjärjestelmä-mallipohjan tekemiseen. Tarvittavat paketit ladataan verkon tallennuspaikasta hardware nodelle mallipohjan luonnin aikana. Ne asennetaan väliaikaiseen virtuaaliympäristöön joka pakataan yhdeksi gzipatuksi tar-palloksi. Palloa kutsutaan mallipohja välivarastoksi (template cache). Mallipohja välivarasto on käytännössä esiluotu virtuaaliympäristö ja sitä käytetään nopeaan virtuaaliympäristöjen tekemiseen. Luonti nopeutuu, koska virtuaaliympärisöä varten ei käyttöjärjestelmää tarvitse asentaa vaan mallipohja välivarasto-pallo puretaan käyttövalmiiksi. Mallipohja välivarasto tiedostot tallennetaan /vz/template/cache/-kansioon. [9, s. 17.]

Kun jakelusta julkaistaan päivityksiä, käyvät ajan myötä kaikki mallipohja välivarastot vanhoiksi. Mallipohja välivaraston päivitys on kuitenkin verrattain helppoa.

Päivitystapa on jakelukohtainen. Useimmiten päivitystä varten luodaan väliaikainen säiliö, jossa suoritetaan normaali päivityskäsky, esimerkiksi apt-get tai yum. Päivityksen jälkeen mallipohja välivarasto pakataan ja vanha korvataan uudella. [9, s. 17.]

3.4 Asennus

OpenVZ-ydin asennetaan jonkin tuetun Linux-jakelun ytimen tilalle. Ydin pyrkii tukemaan samoja laitteita kuin alkuperäinenkin ydin. Laitteistovaatimukset riippuvat suoritettavien virtuaaliympäristöjen määrästä. Minimivaatimuksena on 128 MB käyttömuistia, 4 GB kovalevytilaa ja verkkokortti. Suoritettavien virtuaaliympäristöjen määrää rajoittaa suoraan keskusyksikön teho. Toisaalta virtuaaliympäristöissä suoritettavien ohjelmien vaatimukset riippuvat suoraan käyttömuistin määrästä. Jokainen virtuaaliympäristö tarvitsee myös jonkin verran kovalevytilaa. [9, s. 21.]

Jos virtuaaliympäristölle tarvitaan levykiintiöiden määrittelyjä, on suositeltavaa, että säiliöiden omille hakemistoille käytetään omaa loogista levyä. Oletuksena omat kansiot ovat hakemistossa /vz/private/<veid>. Säiliökohtaisten levykiintiöiden käyttö vaatii ext2- tai ext3-tiedostojärjestelmän. [11.]

Muokattu ydin noudetaan pakettien tallennuspaikasta jakelun tarjoamilla työkaluilla, esimerkiksi yumilla. Ytimen asennuksen yhteydessä voidaan päättää, minkätyylinen se on. Ydintyylit on optimoitu joko uniprosessorille tai multiprosessorille ja tietylle muistimäärälle. Oikean ydintyylin valitsemisella luvataan saavutettavan 5−15 % parempi suorityskyky. Ydin on myös mahdollista kääntää itse, jos valmiiksi tehdyt eivät jostain syystä käy. [11.]

Ytimen asennuksen jälkeen muokataan bootloader vastaamaan oikeata ydin-imagea. Selvyyden vuoksi otsikko muokataan alkuperäisestä jakelusta joksikin kyseistä ydintä kuvaavaksi. Bootloader-ohjelmista Grub osaa muokata itsensä automaattisesti. [11.]

Ennen ensimmäistä uudelleenkäynnistystä pitää /etc/sysctl.conf-tiedostoa muokata. Tiedostossa määritellään internet-yhteyteen liittyviä asioita, kuten pakettien uudelleen reititystä. Ensimmäisen käynnistyksen jälkeen tulee asentaa OpenVZ-käyttäjätason työkalut. Tärkein työkalu on vzctl, jolla hallinnoidaan säiliöitä. [11.]

Viimeisenä käynnistetään OpenVZ-ytimen moduulit /sbin/service vz start-komennolla. Komento käynnistäisi, mikäli sellaisia olisi määritelty, myös kaikki automaattisesti käynnistyväksi määritellyt säiliöt. Jatkossa komento suoritetaan itsestään käynnistyksen yhteydessä. [11.]

Asennuksen jälkeen voidaan alkaa luoda käyttöjärjestelmämallipohjia ja säiliöitä.

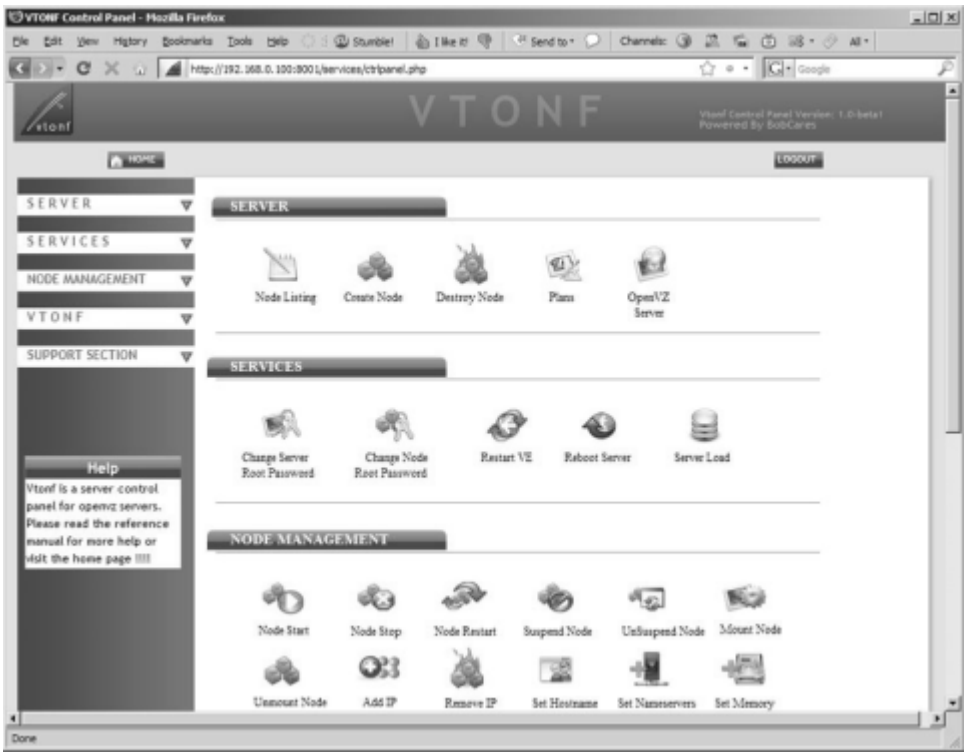
3.5 Hallinnointi

Käyttäjätason työkalupaketin vzctl on tärkein työkalu OpenVZ-hallinnointiin. Tekstipohjaisena sen käyttö on kuitenkin haastavaa ja tarpeettoman monimutkaista normaalikäyttöön. Työkalupaketissa on tarjolla monenlaisia komentorivipohjaisia työkaluja, vzquota on kuitenkin ainoa vaadittava vzctl:n lisäksi. Työkalujen komennot muistuttavat normaaleja Linux-käskyjä, mutta ne alkavat kirjaimilla vz.

OpenVZ-hallinnointitiedostoja on kahdenlaisia, yleinen hallinnointitiedosto /etc/sysconfic/vz ja virtuaaliympäristöjen hallinnointitiedosto /etc/sysconfig/vz-scripts/vpsid.conf. Yleinen hallinnointitiedosto määrittelee yleiset ja oletusasetukset virtuaaliympäristön toiminnalle, kuten lokitiedoston asetukset, oletusasetustiedoston ja käyttöjärjestelmä-mallipohjan tiedot. Virtuaaliympäristön hallinnointitiedosto puolestaan määrittelee kyseisen ympäristön asetukset, kuten levytilan käytön, resurssirajoitteet ja IP-osoitteen. Saman määrittelyn löytyessä molemmista tiedostoista on virtuaaliympäristökohtaisella tiedostolla suurempi prioriteetti. [9, s. 18.]

Asetustiedostot luetaan, kun OpenVZ tai säiliö käynnistetään. Osaa asetuksista voi kuitenkin muokata lennosta vzctl-komennolla. Tällöin muutoksen voi joko tallentaa tai olla tallentamatta asetustiedostoon. [9, s. 18.]

Virtuaaliympäristöjen hallintaa helpottamaan on kehitetty useita graafisen käyttöliittymän kautta toimivia sovelluksia. Osa sovelluksista on ilmaisessa jakelussa. Kuvassa 4 näkyvä vtonf-sovellus on yksi näistä ilmaissovelluksista.



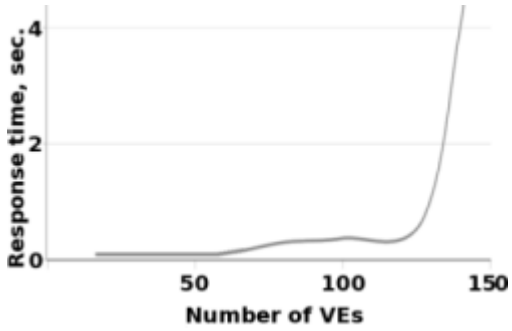
Kuva 4. vtonf-hallinnointiohjelman päänäkymä

Virtuaaliympäristöjen päivittäminen onnistuu komentorivikomennolla vzpkgcache, jolloin kaikki säiliöt päivittyy yhtäaikaisesti.

3.6 Suorituskyky

OpenVZ:n säiliöivän toimintatavan takia virtualisoinnista aiheutuva overhead on usein olematon. Säiliöitä voidaan luoda tiheästi, ennen kuin hardware noden toiminta hidastuu. Teoriassa OpenVZ pystyy isännöimään satoja säiliöitä. Keskusyksikön teho ja muistin määrä rajoittavat suorituskykyä. Kuvassa 5 esitetty kuvaaja on saatu tietokoneella, jossa on 768 MB keskusmuistia. Kuvaajan palvelimen säiliöidyt

webpalvelimet oli asetettu palauttamaan pyytäjälle staattinen verkkosivu. Säiliöiden määrän kasvaessa vasteaika kasvaa käyttömuistin loppuessa. [12.]



Kuva 5. Virtuaaliympäristöjen lukumäärä suhteessa vasteaikaan [12].

Kyseisellä tietokoneella voidaan siis suorittaa yhtäaikaisesti noin 120:tä tällaista virtuaaliympäristöä vasteajan pysyessä siedettävänä. Käyrä ekstrapoloituu lineaarisesti, joten vastaavasti tietokoneella jossa on 2 GB käyttömuistia pitäisi pystyä suorittamaan 320:tä vastaavaa virtuaaliympäristöä. [12.]

HP-laboratorioiden suorittama vertailu Xenin ja OpenVZ:n suorituskyvyissä todistaa, että overhead OpenVZ käytössä on hyvin vähäinen. Xenin vasteaika kasvoi sietämättömäksi hyvin pian palvelimien lisäämisen yhteydessä, kun taas OpenVZ palveli käytännössä yhtä nopeasti kuin erillinen palvelin. Suorituskyky ei ole kuitenkaan ainoa ratkaiseva tekijä virtualisoinnissa. OpenVZ:n virtualisointitapa on rajoitetumpi kuin Xenin. HP:n testissä virtualisointi rasitteiden erojen suurimmakasi syyksi arvellaan L2-välimuistin huteja. OpenVZ:n, koska sen resurssit on rajoitettu papulaskureilla eikä eristämällä, ei tarvitse tyhjentää välimuistia, kun vaihdellaan virtuaaliympäristöjä. [13, s. 11]

3.7 Tulevaisuus

On epätodennäköistä, että OpenVZ kilpailisi tulevaisuudessa merkittävästä markkinaosuudesta virtualisoinnissa. VMwaren merkittävin kilpailija tällä hetkellä on Microsoft [14]. Parallels-emoyhtiön mukaan heidän kaupallinen Virtuozzonsa on kuitenkin markkinajohtaja käyttöjärjestelmätason – joka on osa-alue virtualisoinnissa –

virtualisoinnissa [15]. OpenVZ kehitys kuitenkin jatkuu yrityksen omien sunnitelmien mukaisesti. Ytimen osalta tulevaisuudessa ollaan parantamassa tasapuolisen cpu-vuorottajan toimintaa, korjaamassa toisen tason levykiintiöiden vikoja, muokkaamassa API-tukea tukemaan VMwarea ja Xeniä ja virtualisoimassa IPsec ja VPN. [16.] Koska OpenVZ on kaupallisen Parallels-yrityksen tukema, heijastuvat yhtiön tapahtumat myös ilmaiseen sivuprojektiin.

Parallels on kehittämässä tyypin 1 hypervisoria. Kehitetty tuote tulee olemaan kuitenkin erillinen eikä se vaikuta OpenVZ kehitykseen. OpenVZ ei voi olla tyypin I VMM, koska se pohjautuu Linuxin ytimeen. Säiliöivä virtualisointitapa on tekniikkanakin sellainen, että sovelluskohteita tällaiselle hypervisorille on vaikea keksiä. [17.]

4 Järjestelmän testaaminen

Olemassa olevien tutkimusten perusteella voidaan olettaa, että OpenVZ-säiliöitä voidaan suorittaa useita satoja normaalilla työpöytäkäyttöön tarkoitetulla tietokoneella. Testausta varten asennettiin CentOS 5.3 i386 Linux-käyttöjärjestelmä Dell Optiplex GX620-tietokoneelle. CentOS perustuu Red Hat Enterprise Linuxiin. CentOS on poistanut jakelupaketista Red Hatiin viittavat kuvat ja paketit ja muokannut osaa paketeista.

CentOS valittiin testauskäyttöjärjestelmäksi palvelinpohjaisuuden takia. Toinen vaihtoehto, Fedora, on ennemminkin loppukäyttäjille tarkoitettu Linux-jakelu, jossa on uusimmat testiversiot ohjelmistoista. Oletettavasti CentOS on vakaampi käyttöjärjestelmä. [18.]

Käytettävässä jakeluversiossa oli ytimen versio 2.6.18-128-el5. OpenVZ-ytimen versio oli 2.6.18-128.2.1-el5.028stab064.4PAE. Mallipohja välivarasto luotiin HyperVM sovelluksen esittelyversiolla, josta käytössä oli versio 2.0.7993 Stable. Säiliöt luotiin vzctl-työkalulla komentojon kautta sen vaivattomuuden vuoksi. HyperVM:ää käytettiin

sen mukana tulevien valmiiden mallipohja välivarastojen vuoksi. Sen esittelyversio on kuitenkin rajoitettu viiteen säiliöön, joten niiden hallinnointi tehtiin vtonf-ohjelmalla.

Palvelinkoneessa oli 3,0 GHz:in Intel Pentium D tuplaydin-suoritin, 1 GB normaalia DDR2-533MHz muistia ja 74 GB:en kovalevy. Muistin lisäksi käyttöjärjestelmälle partitioitiin 2 GB virtuaalimuistia.

Käyttöjärjestelmän ja OpenVZ asennuksen jälkeen valittiin sopiva mallipohja välivarasto. Siihenkin valittiin käyttöjärjestelmäksi CentOS, jotta isäntäjärjestelmän ja asiakasjärjestelmien välillä olisi mahdollisimman vähän eroja. Mallipohja välivarastona käytettiin HyperVM:n mukana tullutta centos-5-i386-full-mallipohjaa. Testattava sovellus, Apache, toimi tässä kokoonpanossa. Mallipohja välivarasto piti vielä muuttaa tähän testiin sopivaksi niin, että Apache oli käynnissä ja virtuaaliympäristöt saivat jokainen oman IP-osoitteensa. Myös papulaskurit asetettiin sopiviksi. Testausta varten ei ollut mielekästä asettaa kaikelle mahdolliselle rajoituksia. Muistin määrä ja prosessoriaika olivat ainoat rajoitetut resurssit.

Isäntäjärjestelmästä on poistettu palomuuri ja SElinux käytöstä. Myöskään asiakasjärjestelmissä näitä ei käytetä. Palomuurin käyttäminen aiheuttaisi ylimääräistä prosessorikuormaa, joka oikeassa ympäristössä pitää ottaa huomioon. Testin kannalta se oli kuitenkin järkevämpää jättää kokonaan kytkemättä päälle. Palomuurikonfiguraatiot voivat olla minkä tahansa kaltaisia, minkä vuoksi yhden palomuurikonfiguraation testaaminen antaisi vääristävän kuvan järjestelmän suorituskyvystä.

4.1 Yksittäisen palvelimen suorituskyky

Järjestelmän suorituskyvyn testaus suoritettiin samalla Linux-palvelimella, johon OpenVZ on asennettu. Yksittäistä palvelinta testattaessa yhtään virtuaaliympäristöä ei ollut käynnistettynä ja palvelupyynnöt kohdistettiin isäntäjärjestelmän web-palvelimeen. Apache määritettiin vastaamaan palvelupyyntöön sivulla, jonka koko oli 5 kB. Vastaussivu sisälsi tekstiä. Jotta koneen resurssien rajat tiedettäisiin etukäteen, suoritettiin suorituskyvyn testaus ensin palvelimella toimivaan yhteen Apache-palvelimeen. Palvelimen kuormittuessa tutkittiin, mikä resurssi muodostuu

ensimmäisenä pullonkaulaksi. Kovalevy oli kiinni SATA300-väylässä, jonka teoreettinen tiedonsiirtonopeus on 300 MB/sekunti ja keskimääräinen hakuaika alle 9,5 millisekuntia. Koska haettava sivu oli sama ja kovalevyssä on 8 MB välimuistia, on hyvin epätodennäköistä, että siirtonopeus muodostuisi pullonkaulaksi. Muiden suorittamien testauksien perusteella oli myös oletettavaa, että fyysiset resurssit olisivat riittävät kovaankin kuormaan. Todennäköisesti joko Linuxin tai Apachen asetukset aiheuttaisivat vasteaikojen kohtuuttoman kasvun ja näitä muokkaamalla saavutettaisiin huomattavastikin parempi suorituskyky.[19.]

Testi suoritettiin paikalliselta tietokoneelta. Tietokoneelta lähetettiin ohjelmalla rajoitettu määrä kyselyitä satunnaisesti virtuaaliympäristöille. Apachessa käytettiin muokattua konfiguraatiota. Oletusarvoilla Apache ei pysty palvelemaan kuin muutamaa sataa asiakasta yhtä aikaa. Samalla muutettiin avoimena olevien tiedostojen maksimimäärä 1024:stä 8196:een.

Suorituskyvyn testauksessa oli tarkoitus selvittää virtualisointi rasitteen vaikutus vasteaikoihin. Tästä syystä ei ollut erityisen merkityksellistä koettaa saada maksimisuorituskykyä esimerkiksi Apachesta. Oletusarvoisessa Apachessa on mukana useita moduuleita, joita tähän staattisen verkkosivun palautukseen ei käytetä.

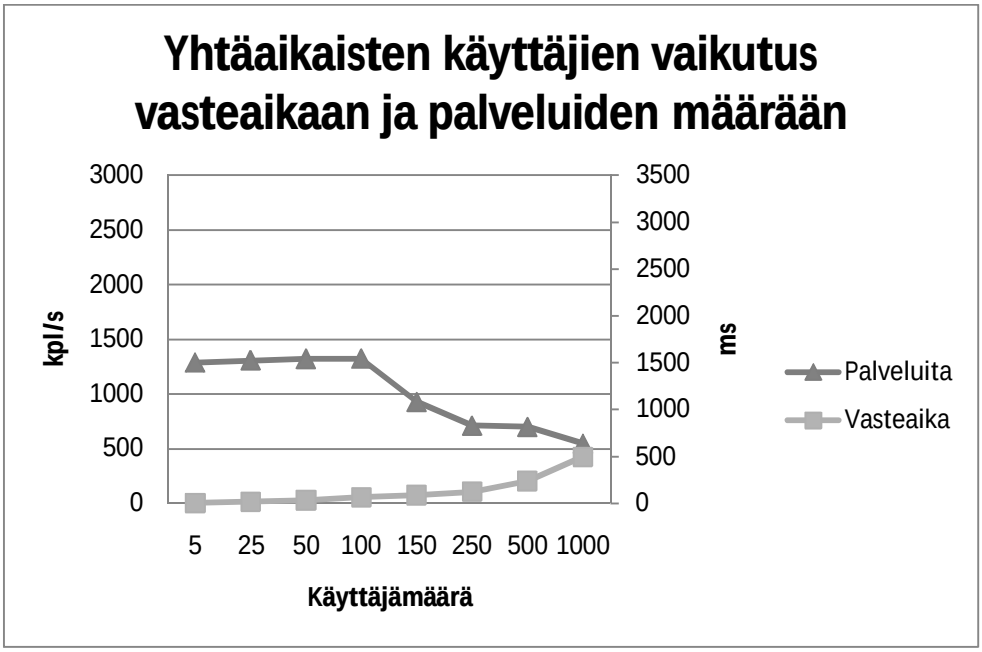
4.1.1. http_load

Järjestelmän suorituskyky testattiin ajamalla http_load-ohjelmaa kymmenen kertaa erilaisilla pyyntömäärillä. http_load on webpalvelimen testaukseen kehitetty ohjelma, joka lähettää useita pyyntöjä yhtä aikaa palvelimelle. Sovellus toimii yhdessä prosessissa, joten sen vaikutus järjestelmän resursseihin pitäisi olla vähäinen. Käytettävä versio oli 12mar2006. Testauksen ajan tarkkailtiin muistin ja prosessorin käyttöastetta top- ja vmstat- ohjelmilla. Testaus suoritettiin parametreillä ’./http_load -parallel {5,25,50,100,150,250,500,1000} -fetches 12000’. Parallel-parametrillä määritetään, montako yhteyttä palvelimelle luodaan. Fetches-parametri kertoo, montako kertaa www-dokumentti näillä yhteyksillä haetaan. Ohjelma kertoo loppuraportissaan erilaisia tietoja palvelimen suorituskyvystä, mutta mielenkiintoisia arvoja ovat

keskimääräinen vasteaika ja moneenko palvelupyyntöön palvelin vastaa sekunnissa. Esimerkki http_load-ohjelman loppuraportista on kuvassa 6. Palvelimen kuormitustestin tulokset on esitelty kuvassa 7.

```
12000 fetches, 100 max parallel, 5.6724e+07 bytes, in 11.6891 seconds
4277 mean bytes/connection
1033.68 fetches/sec, 4.88619e+06 bytes/sec
msecs/connect: 0.478909 mean, 88.105 max, 0.042 min
msecs/first-response: 76.748 mean, 1003.53 max, 1.249 min
HTTP response codes:
   code 200 -- 12000
```

Kuva 6. Esimerkki http_load-ohjelman loppuraportista



Kuva 7. Yhtäaikaisen käyttäjien määrän vaikutus ilman virtualisointia, http_load

Kuvasta on havaittavissa, että edes tuhannella yhtäaikaisella käyttäjällä vasteaika ei ollut sietämätön. Kun palvelupyynnöt jaetaan useisiin eri virtuaaliympäristöissä toimiviin palvelimiin, on yhden prosessin taakka hieman pienempi. Vastaava kuvaaja virtuaaliympäristöjen kohdalla, mikäli OpenVZ aiheuttaa virtualisointirasitetta, kasvaa nopeammin. Testin aikana prosessorin käyttöaste molemmilla ytimillä kävi melkein sadassa, kun yhtäaikaisia käyttäjiä oli viisi. Toisaalta I/O-operaatioita ei suoritettu lainkaan, joten muistia oli riittävästi.

4.1.2. Siege

Järjestelmää testattiin myös toisella samanlaiseen tarkoitukseen tehdyllä ohjelmalla, Siegellä. Siegen toimintaperiaate on sama kuin http_load:ssa. http_load-ohjelmalla tarkasteltiin tässä yhteydessä palvelimen suorituskykyä. Siege-ohjelmaa käytettiin virtualisoinnin rasitteen selvitykseen. Jotta rasitetta voitaisiin mitata, täytyisi säiliöimättömän palvelimen laitteisto saada samanlaiseksi kuin yksittäisen säiliön resurssit ovat. Koska yksittäisen sovelluksen muistinkäytön rajoittaminen on hankalaa Linuxissa, varsinkin kun Apache tekee lapsiprosesseja kuormaa tasatakseen, muutettiin virtuaalikoneiden konfiguraatio vapaammaksi. Muisti- ja prosessoriaikarajoitukset poistettiin. Kun perusjärjestelmä oli testattu täydellä raudalla testattiin virtuaaliympäristö ilman rajoituksia. Siege-ohjelmasta oli käytössä versio 2.69.

Perusjärjestelmän vasteaika mitattiin ajamalla siege kymmenen kertaa kohteenaan paikallinen palvelin. Ohjelma käynnistettiin komennolla ’siege -b -c100 -r20 http://127.0.0.1’. Parametreillä säädetään viivästystä, yhtäaikaisten käyttäjien määrää ja testikertojen määrää. Viivästyksellä on tarkoitus simuloida ihmisten käyttäytymistä. Tällaisen testauksen kannalta se ei ole merkityksellistä, joten käytettiin ’benchmark’-tilaa. Esimerkki Siege-ohjelman raportista kuvassa 8. Järjestelmän vasteajat on eritelty taulukossa 2.

```
** SIEGE 2.69
** Preparing 100 concurrent users for battle.
The server is now under siege..      done.
Transactions:                   2000 hits
Availability:                 100.00 %
Elapsed time:                   0.83 secs
Data transferred:               9.02 MB
Response time:                  0.03 secs
Transaction rate:            2409.64 trans/sec
Throughput:                    10.86 MB/sec
Concurrency:                   68.43
Successful transactions:        2000
Failed transactions:               0
Longest transaction:            0.72
Shortest transaction:           0.00

12000 fetches, 100 max parallel, 5.6724e+07 bytes, in 11.6091 seconds
4277 mean bytes/connection
1033.68 fetches/sec, 4.88619e+06 bytes/sec
msecs/connect: 0.478989 mean, 88.105 max, 0.042 min
msecs/first-response: 76.748 mean, 1003.53 max, 1.249 min
HTTP response codes:
   code 200 -- 12000
```

Kuva 8. Esimerkki Siege-ohjelman raportista

Taulukko 2. Perusjärjestelmän vasteajat

Testiajo	Vasteaika/sek
1	0,01
2	0,01
3	0,01
4	0,02
5	0,01
6	0,02
7	0,02
8	0,01
9	0,01
10	0,02
ka	**0,014**

Testin mukaan rajoitettu perusjärjestelmä palvelee sataa yhtäaikaista asiakasta keskimäärin 14 ms:n vasteajalla. Mikäli virtualisoinnista ei aiheutuisi rasitetta, tulisi yksittäiselle säiliölle tehdyn vastaavan ajon antaa sama tulos.

4.2 Virtuaalipalvelimien suorituskyky

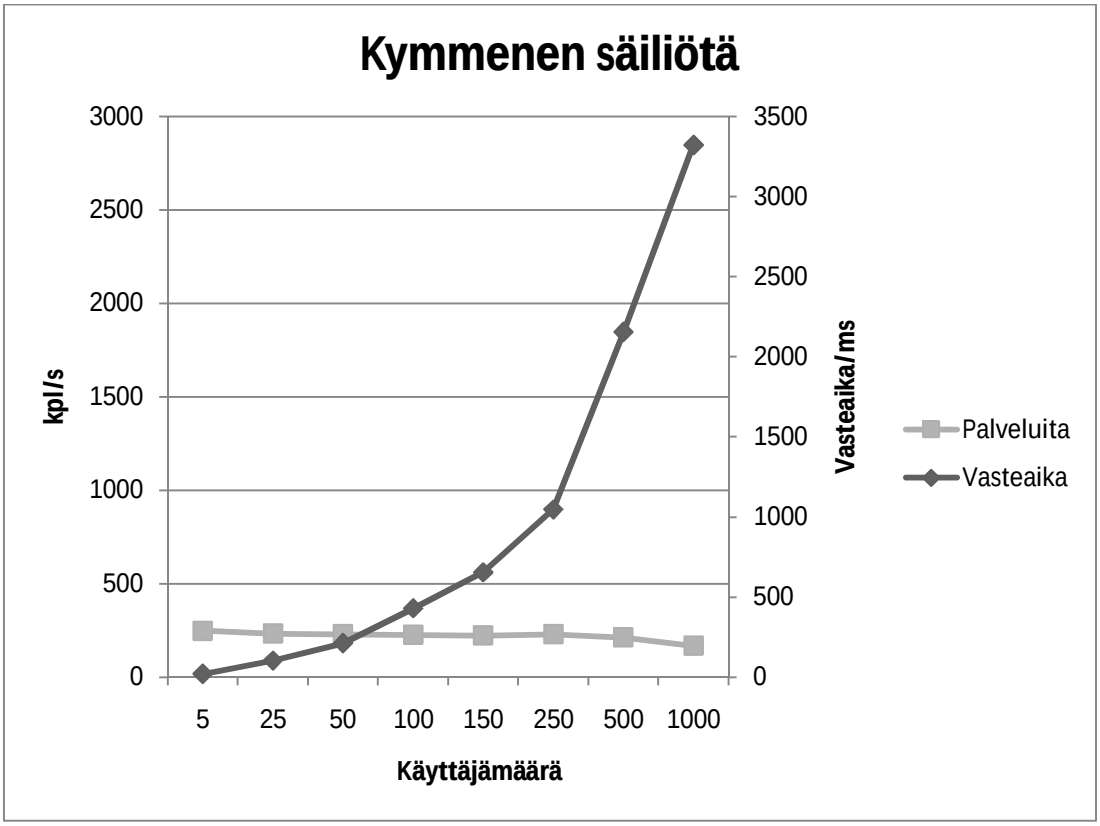
Tietokoneen muistin maksimimäärän ollessa 1 GB voidaan virtuaaliympäristöjä luoda rajoitettu määrä. Jokaisen virtuaaliympäristön prosessoriaika oli rajattu maksimissaan 5 %:iin. Muistin määrä rajoitettiin niin, että luotettavasti voitiin suorittaa noin kolmeakymmentä virtuaaliympäristöä. Jokaiselle säiliölle varattiin 32 MB muistia. Asetus tehtiin vtonf-hallintaohjelman plan wizard -moduulilla, joka rakentaa asetuksista mallipohjan. Mallipohja voidaan sittemmin kohdistaa mihin tahansa virtuaalikoneeseen. Asetus privvmpages säädettiin arvoon 65536 ja asetus guaranteed memory arvoon 32768. Asetuksella voidaan teoriassa suorittaa kuuttakymmentä virtuaaliympäristöä, mutta tällöin osa prosesseista saattaa tuhoutua, mikäli muisti loppuu kesken. Käytännössä raskaan kuormituksen takia säiliöitä ei kuitenkaan ole mielekästä suorittaa niin montaa. Koska säiliön piti tulla toimeen näin pienellä muistimäärällä, sammutettiin tarpeettomat palvelut. Järjestelmäpalvelujen lisäksi vain httpd ja sshd jäivät päälle. sshd:tä tarvittiin, jotta säiliötä päästiin tarvittaessa pääteyhteyden kautta muokkaamaan.

Jokaiselle virtuaaliympäristölle annettiin yksi yksityinen IP-osoite. Tämän osoitteen kautta ympäristöjä testauksessa satunnaisesti kuormitettiin. Molemmat testausohjelmat sallivat osoitetiedoston käytön, josta poimitaan satunnaisesti osoite palvelupyyntöä varten.

Testi suoritettiin neljällä eri virtuaaliympäristömäärällä. Määrät olivat 10, 30, 50 ja 70. Oletettavasti tiukat resurssirajat tuovat ongelmia vähäisten säiliöiden määrän kanssa. Viidelläkymmenellä säiliöllä ei pitäisi tulla ongelmia, ellei yksittäisen palvelimen kuormitus kasva sietämättömäksi. Tätä suuremmat säiliömäärät todennäköisesti kärsivät muistin loppumisesta, jolloin jouduttaisiin käyttämään virtuaalimuistia. Kaikki virtuaaliympäristöt luotiin välittömästi, mutta niitä käynnistettiin vain testin tarvitsema määrä. Testi tehtiin komentojonolla, joka suoritti saman http_load-käskyn kymmenen kertaa, testiajojen välissä pidettiin minuutin tauko. Testiajon jälkeen palvelin käynnistettiin uudelleen ja odoteltiin, että kuormitus – käytännössä säiliöiden latautuminen – laskee normaalille tasolle. Tulokset tallennettiin tiedostoon niiden analysoinnin helpottamiseksi.

4.2.1. http_load

Virtualisoidun järjestelmän suorituskyvyn testaamiseksi ajettiin http_load ohjelma samoilla parametreillä uudelleen erilaisilla virtuaalipalvelinmäärillä. Kymmenellä virtuaalipalvelimella vasteajat pitenivät huomattavasti. Tämä ei johtunut virtualisoinnista vaan jokaiselle säiliölle jyvitetystä muistin ja prosessoriajan määrästä. Yksittäinen säiliö saavutti nopeasti resurssiensa rajat, kun pienimuistista palvelinta pommitettiin suurella käyttäjämäärällä. Kuvassa 9 esitetään vasteaikojen ja palvelumäärän kuvaajat.



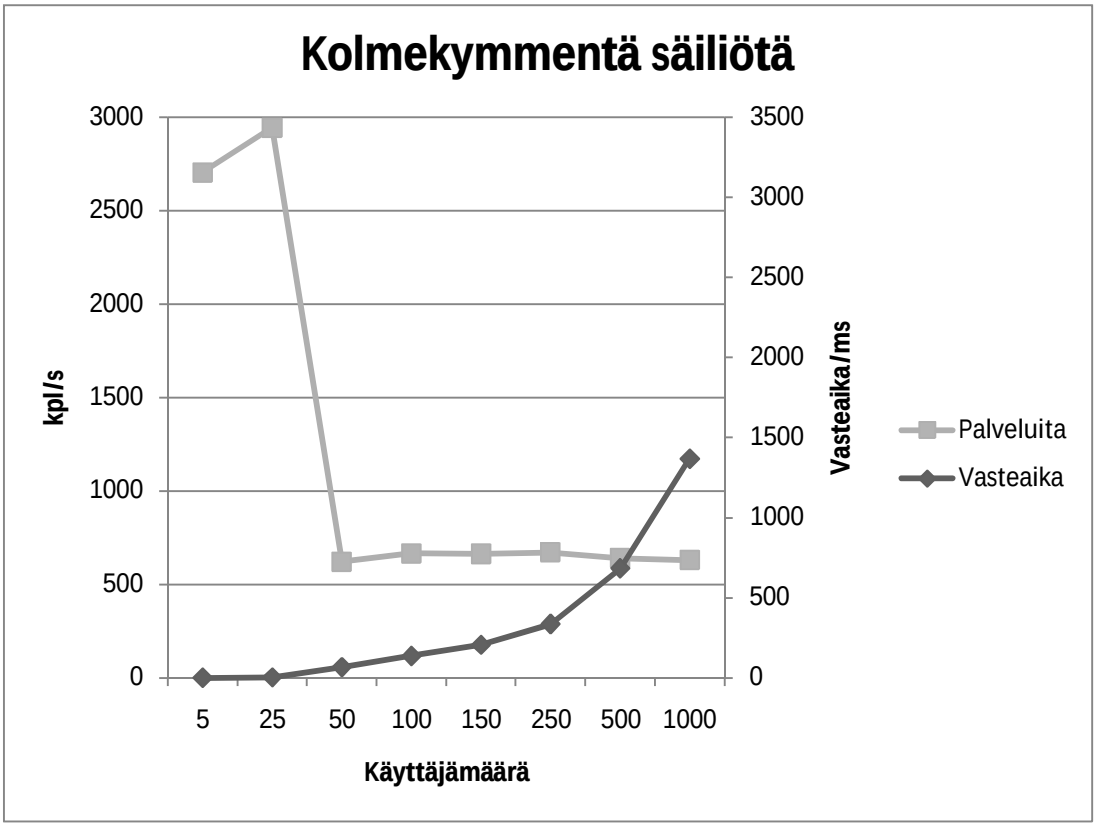
Kuva 9. Vasteajat ja palvelumäärä kymmenellä säiliöllä.

Palveluiden määrä on vähentynyt dramaattisesti omistautuneen palvelimen (dedicated server) tapauksesta ja vasteajat nousivat sekuntiin, kun yhtäaikaisia käyttäjiä oli 250. Kun yhtäaikaisia käyttäjiä oli tuhat, saatiin palvelin osittain ylikuormitettua DoS-hyökkäyksen tapaisen kyselymäärän takia. Säiliöiden muistirajoituksiin ei kuitenkaan koskettu, koska suuremmalla palvelinmäärällä yksittäisen palvelimen kuormitus

vähenee. Edellinen vasteaikojen valtava kasvu johtui käytännössä ainoastaan muistin määrän rankasta rajoituksesta. Palvelinten suorituskyky muistin määrään nähden oli kuitenkin varsin hyvä.

Vasteaikoihin vaikuttaa myös se, että kyselyt tehtiin usealle palvelimelle yhden sijaan. Omistautuneen palvelimen tapauksessa kaikki yhtäaikaiset yhteydet olivat samalle palvelimelle. Verrattuna kuvaan 7 huomataan, että kuvan 9 vasteajat ovat huomattavasti suurempia. Koska teoriaosiossa käsitellyn testauksen testiympäristöä ei tunneta muuten kuin laitteiston osalta, ei vertailu ole järkevää.

Tasautuneen kuorman ansiosta vasteajat kolmellakymmenellä säiliöllä olivat varsin hyviä. Kuvassa 10 esitellään kuvaajat kolmellakymmenellä säiliöllä.

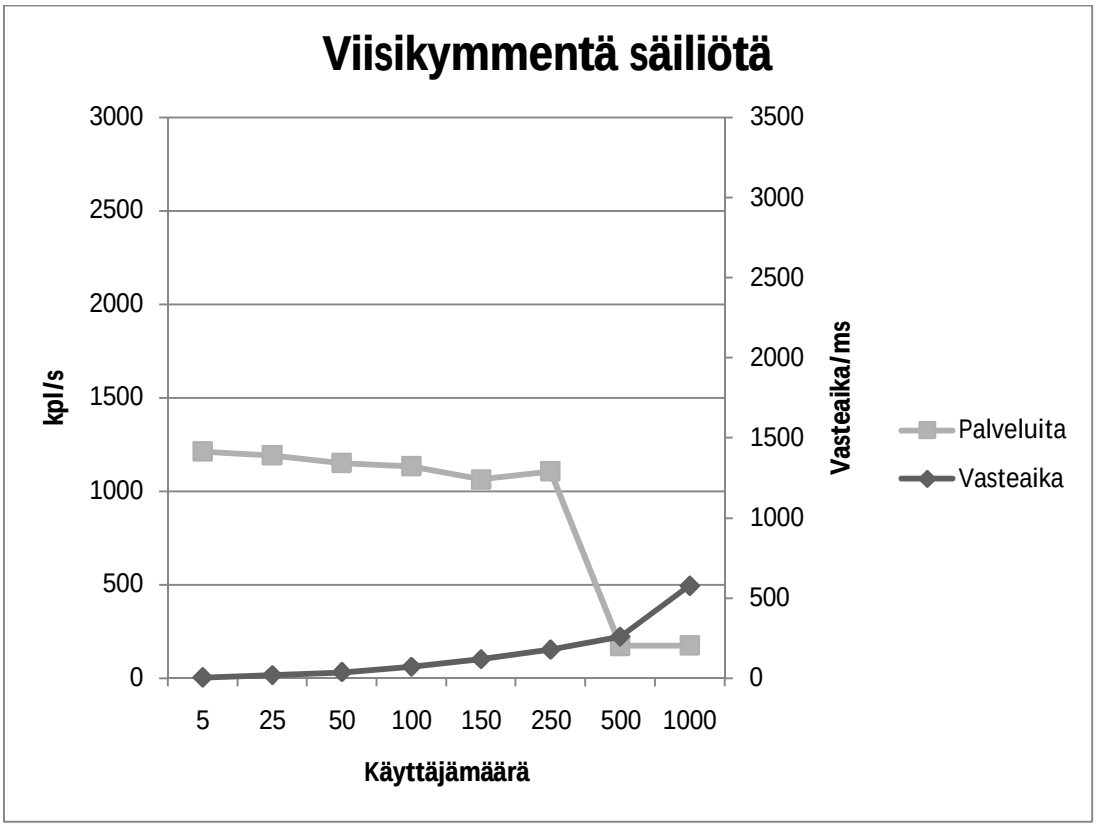


Kuva 10. Vasteaikojen ja palvelumäärän muutokset kolmellakymmenellä säiliöllä.

Kun yhtäaikaisten käyttäjien lukumäärä ylitti sata, alkoi vasteaika kasvaa verrattuna omistetautuneen palvelimen tapaukseen. Vasteaika 150 yhtäaikaisella käyttäjällä oli

noin kaksinkertainen. Vasteajat olivat kuitenkin siedettäviä lukuun ottamatta viidensadan ja tuhannen yhtäaikaisen käyttäjän tapauksia.

Säiliömäärän noustessa viitenkymmeneen säiliöön saatiin kuvan 11 mukainen kuvaaja.



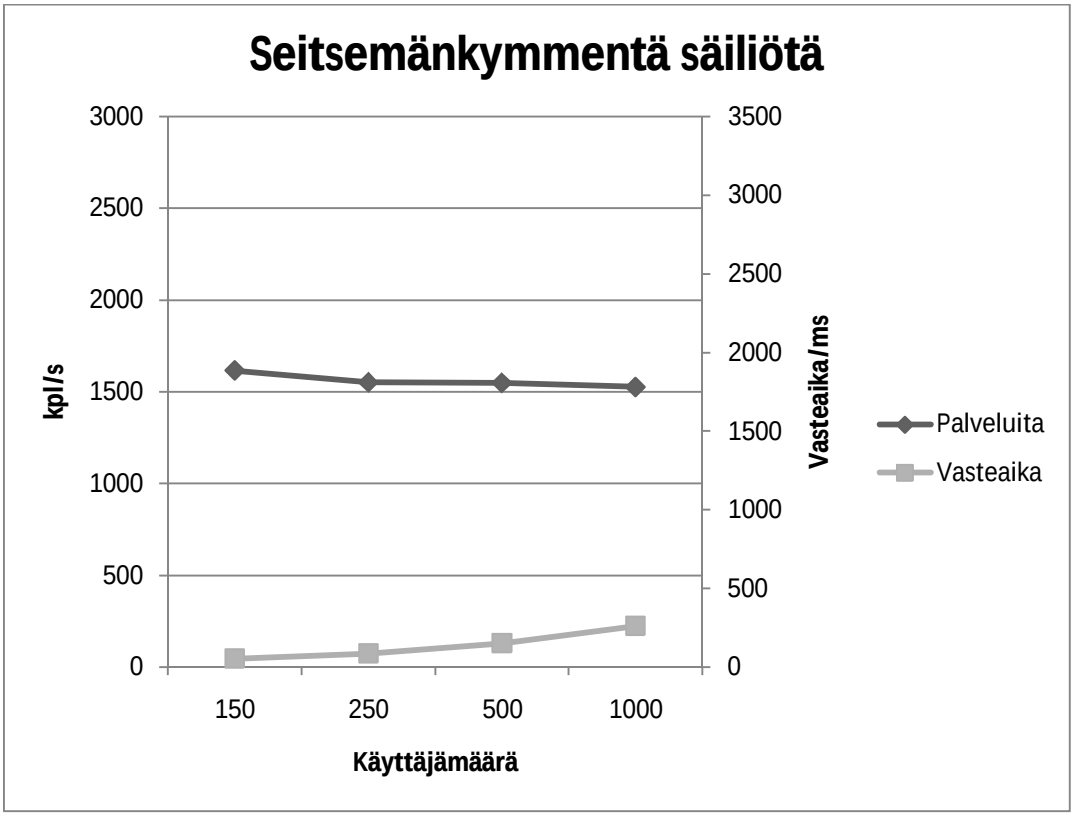
Kuva 11. Vasteaikojen ja palvelumäärän muutokset viidelläkymmenellä säiliöllä.

Vasteajoissa oli hyvin pieni ero omistautuneen palvelimen tapaukseen nähden. Vasteajat ovat kauttaaltaan noin 10 % isommat. Palvelun tavoitettavuus oli kuitenkin suurimmilla yhtäaikaisilla käyttäjämäärillä vain 98 %. Kyselyistä 2 % päättyi ilman vastausta. Omistautuneella palvelimella kaikki saivat palvelupyyntöönsä vastauksen.

Se, että vasteajat ovat parantuneet kolmenkymmenen säiliön tapauksesta viittaisi siihen, että säiliöiden resurssit olivat vieläkin riittämättömät. Tähän viittaa myös se, että kolmenkymmenen säiliön tapauksessa palvelimet pystyivät tarjoamaan yli tuhatta palvelua/sekunti vain 25 yhtäaikaisella käyttäjällä. Tällainen skenaario ei ole kuitenkaan

mitenkään todennäköinen. Käytännössä se tarkoittaisi sitä, että yhtäaikaiset käyttäjät kykenisivät tekemään jokainen yli 40 palvelupyyntöä sekunnissa.

Oletetun virtuaalimuistin käytön todentamiseksi lopuksi otettiin käyttöön 70 säiliötä yhtäaikaisesti. Virtuaalipalvelimet toimivat odotettua paremmin vähäisemmällä, alle 250 yhtäaikaisen käyttäjän kuormalla, mutta suuremmilla kuormilla jouduttiin osa tiedosta siirtämään virtuaalimuistiin. Näin suurella palvelinmäärällä testaaminen aloitettiin 150 yhtäaikaisella käyttäjällä. Jako tarkoittaa, että aluksi jokaista palvelinta kuormitetaan keskimäärin kahdella palvelupyynnöllä sekunnissa ja suurimmalla käyttäjämäärällä palvelupyyntöjä oli keskimäärin 14 sekunnissa. Tilanne on esitetty kuvassa 12.



Kuva 12. Vasteaikojen ja palvelumäärän muutokset seitsemälläkymmenellä säiliöllä.

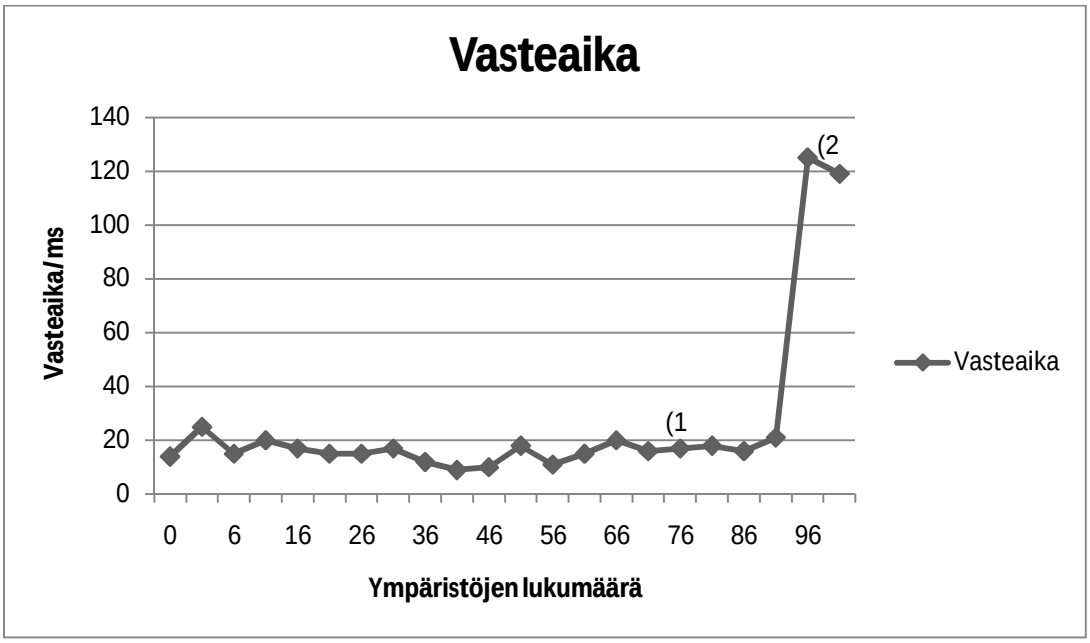
Tulosten mielenkiintoa lisää se, että vasteajat olivat parempia kuin omistautuneella palvelimella. Selityksenä täytyy olla Apache. Koska yksittäisen palvelimen kuormitus oli näin suurella säiliömäärällä kohtalaisen vähäinen ja vasteajat suurimmallakin

kuormalla pysyivät alle 300 ms:n, on tulosten ainoa järkevä selitys palvelinsovelluksen rajoitteet. Verrattaessa omistautuneen palvelimen ja 70:tä säiliötä suorittavan palvelimen 150 yhtäaikaisen käyttäjän vasteaikaa on ero säiliöitetyn palvelimen eduksi noin 30 ms.

http_loadilla tehtyjen testien tuloksista ei suoraan voi sanoa, aiheuttaako OpenVZ virtualisoinnillaan rasitetta. Käyristä voidaan kuitenkin tulkita, että palvelimia voidaan suorittaa suuria määriä ennen sen vaikutusta suorituskykyyn. Rasitteesta voidaan kuitenkin turvallisesti sanoa, että tavanomaisella paravirtualisoinnilla ei palvelimella kyettäisi suorittamaan seitsemääkymmentä palvelinta ilman selkeästi huomattavissa olevaa hitautta [13, s. 5.]

4.2.2. Siege

Virtualisoinnista aiheutuvan rasitteen selvittämiseksi Siege-testi suoritettiin kasvattamalla jokaisen suorituskerran jälkeen säiliöiden määrää viidellä. Testi suoritettiin aina samoilla parametreilla. Yhtäaikaisten käyttäjien määrä pidettiin vakiona 100. Hakukertojen määrä asetettiin arvoon 20, jolloin yhteensä palvelin vastasi 2000 palvelupyyntöön. Säiliöiden maksimimäärä oli 101. Testi suoritettiin jokaisella säiliömäärällä kymmenen kertaa ja palvelin uudelleenkäynnistettiin seuraavaa säiliömäärää varten. Testaus aloitettiin yhdellä säiliöllä. Järjestelmissä ei käytetty muistin- eikä prosessoriajan rajoituksia. Säiliökohtaisia rajoituksia ei otettu käyttöön edes silloin, kun säiliöltä selkeästi loppuivat resurssit. Kuva 13 näyttää vasteaikojen vertailun eri säiliömäärillä. Kuvassa on merkitty kohta 1, jossa resurssit alkoivat loppua ja kuormituksen aikana jouduttiin turvautumaan virtuaalimuistiin. Kuvaan on myös merkitty kohta 2, jossa ensimmäisen testiajon vasteaika kasvoi merkittäväksi virtuaalimuistin huomattavan käytön takia. Ensimmäisenä kuvaajassa on säiliöimättömällä palvelimella saatu vasteaika 14 ms.



Kuva 13. Vasteaikoja Siegellä mitattuna

Yhden rajoittamattoman säiliön vasteaika oli 25 ms. Ero säiliöimättömään palvelimeen oli lähes kaksinkertainen. Tämä tarkoittaisi, että OpenVZ:n virtualisointirasite olisi 78 %. Siege ilmoittaa tuloksensa kuitenkin vain sekunneissa ja kahden desimaalin tarkkuudella, minkä vuoksi pyöristyksetkin aiheuttavat virhettä.

Tuloksen tarkistamiseksi sekä säiliöimätöntä palvelinta että yhtä virtuaaliympäristöä testattiin ping-käskyllä. Ping lähettää ICMP ”Echo Request”-paketin kohteeseen ja odottaa ”Echo Response”-pakettia takaisin. Tälle prosessille lasketaan edestakaisinkulkuun kulunut aika. Suoritettu komento oli ’ping -c 50 <ip>’. Komennolla lähetettiin viisikymmentä ICMP-pakettia ja niille laskettiin keskimääräinen kiertoaika. Virtuaaliympäristölle saatiin keskimääräiseksi ajaksi 27 µs. Säiliöimättömän palvelimen vastaava aika oli 23 µs. Tällä perusteella OSI-taso 3:lla toimivalla protokollalla virtualisointirasite olisi 17 %.

4.3 Tulosten analysointi

Työssä rakennettua OpenVZ-järjestelmää voidaan testien perusteella käyttää kohtalaisella toimivuudella jopa tuhannen yhtäaikaisen asiakkaan palvelemiseen

seitsemälläkymmenellä säiliöllä. Järjestelmän suorituskyky on kuitenkin selkeästi kiinni siitä, mitä palveluja säiliössä tarjotaan. Apache-prosessin palauttama pieni staattinen sivu tarvitsee hyvin vähän mitään resursseja. Jos palautetusta sivusta tehtäisiin dynaaminen ja vaikkapa vain yksi tieto haettaisiin jostain tietokannasta, muuttuisivat vasteajat huonommiksi. Tuloksia ei siis voida käyttää minkä tahansa säiliön suorituskyvyn takeena. Tuloksia vertailtaessa on myös huomioitava, että mitä suurempi määrä säiliöitä on käytössä, sitä pienempi on yksittäiseen palvelimeen kohdistuva kuormitus. Kymmenellä säiliöllä tuhannen yhtäaikaisen käyttäjän aiheuttama kuormitus on 100 käyttäjää säiliötä kohden, kun taas seitsemänkymmenen aiheuttama kuormitus on vain 14. Testejä ei kuitenkaan voitu suorittaa suuremmilla käyttäjämäärillä käyttöjärjestelmän ja ohjelmiston rajoitusten vuoksi. http_load ohjelma ei suoriudu yli tuhannen käyttäjän yhtäaikaisesta hyökkäyksestä muistin loppumisen vuoksi. Siegellä muisti loppuu reilulla 250 yhtäaikaisella käyttäjällä. Säiliön muistintarvetta suunniteltaessa merkittävin tekijä on kuormitus. Seuraavaksi merkittävin tekijä on palvelun tyyppi. Yksittäinen säiliö selviää monimutkaisestakin palvelupyynnöstä, jos niitä ei ole suurta määrää yhtä aikaa. Koska työssä jokainen virtuaaliympäristö oli täysin identtinen muiden virtuaaliympäristöjen kanssa ja kuormituksella on suurin merkitys muistinmäärän määrittelyssä, ei yksittäisen ympäristön muistintarvetta ole syytä tarkemmin laskea.

Suhteettoman suuret vasteajat vähäisillä säiliömäärillä johtuvat ainoastaan säiliön sisäisen rajoitetun muistin loppumisesta. Kun Siege-testissä rajoitukset poistettiin, olivat vasteajatkin järkeviä. Muistin loppumisen huomaa myös siitä, että palveluiden määrä sekunnissa putoaa aina, kun muistin loppumisen takia joudutaan odottamaan vastausta.

Testauksien suorittaminen paikalliselta palvelimelta vähentää verkkoviivettä, mutta kuluttaa myös palvelimen resursseja. Tuloksia tarkasteltaessa resurssien kulutus on otettava huomioon. Vaikka prosessorissa on tarpeeksi tehoa palvelemaan yksinkertaista palvelupyyntöä, aiheuttaa valtavan sokettimäärän avaaminen viivästyksen, joka on havaittavissa kummassakin testiohjelmassa jokaisen suorituskerran maksimivasteajassa. Samoin tietojen lataaminen välimuistiin aiheutti useimmiten testiohjelman ensimmäiseen suorituskertaan vasteajan piikin, jonka jälkeen vaste tasaantui. Mikäli

tulokset olisi otettu yhden testikerran perusteella, olisi vasteaikojen kuvaaja noussut nopeammin kuin nyt. Useassa tapauksessa ensimmäisen testiajon vasteaika oli kaksi kertaa suurempi kuin seuraavien ajojen. Jos säiliöiltä pyydettäisiin eri sivuja jokaisella latauskerralla, olisi ensimmäinen vasteaika keskiarvon sijaan totuudenmukaisempi. Edellä mainittu skenaario vastaa enemmän reaalimaailman palvelinkäyttöä.

Virtualisoinnista aiheutuvaa rasitetta mitattaessa saatiin epätyydyttäviä tuloksia. Ohjelmiston valmistajan ilmoittama muutaman prosentin rasite on testauksen perusteella paikkaansapitämätön tällaisella kokoonpanolla. Parallels ei kuitenkaan sivuillaan kerro, miten rasite on mitattu, joten testausmenetelmien erilaisuus todennäköisesti vaikuttaa asiaan. Jos virtualisoinnista aiheutuva rasite kasvattaisi palvelimen kuormaa tosiasiallisesti 78 %, olisi suoritettavien säiliöiden maksimimäärä pienehkö. Testattavan säiliön vaatiessa hyvin vähän resursseja, kuten staattisen sivun palauttavan Apachen tapauksessa, ei rasite paljastu erityisen hyvin.

Testien tuloksien perusteella voidaan työn peruskysymyksiin kuitenkin vastata. Yhdellä fyysisellä palvelimella voidaan suorittaa jopa satoja OpenVZ-säiliöitä. Säiliöiden tarjoamien palvelujen tulee kuitenkin olla varsin yksinkertaisia. Muisti on säiliön tärkein resurssi, ja se loppuu todennäköisesti ensin nykyaikaisesta palvelimesta. Varsin tavanomainen 3 GHz:n tuplaydinprosessori ei ollut edes puoliksi kuormitettu palvelinprosessien takia. Tosin staattisen sivun palauttaminen ei ole mitenkään vaativaa prosessorin suhteen.

5 Soveltaminen Metropolian ympäristöön

Metropolian nykyinen ympäristö selvitettiin sähköpostihaastattelulla. Kysely lähetettiin syyskuussa tietohallintojohtajalle. Vastauksia saatiin sekä tietohallintojohtajalta että järjestelmäpäälliköltä. Alkuperäiset kysymykset vastauksineen ovat liittessä 1.

Nykyään Metropoliassa on noin 80 fyysistä ja 100 virtuaalista palvelinta. Virtualisointialustana toimii VMware Infrastructure 3.5. Kun Metropoliaan valittiin

virtualisointialustaa, oli vaatimuksena Windows-, Linux- ja NetWare-käyttöjärjestelmien tuki. Nykyisistä virtuaalipalvelimista noin puolessa on käyttöjärjestelmänä Linux, neljässä kymmenestä Windows ja yhdessä kymmenestä Netware. Fyysisissä palvelimissa jakauma on samankaltainen. Virtuaalipalvelimilla toteutetaan erinäisiä opetus- ja perusinfrastruktuurin palveluita, esimerkiksi webmailit, Moodle ja kotisivut.

Osa Metropolian palveluista toteutetaan palvelinryppäillä, tällöin monta erillistä fyysistä palvelinta tuottaa samankaltaista palvelua. Tällaiseen ratkaisuun on päädytty palveluiden eristämisen takia. Kuorman takia useita palvelimia yhdelle palvelulle on toteutettu vain webmailia varten. Nykyisellään palvelimien käyttöasteet vaihtelevat 0−70 %:in välillä.

Metropolian tarpeiden vuoksi virtuaaliympäristöä ei ole mahdollista vaihtaa täysin OpenVZ-pohjaiseksi. Lievää kustannussäästöä voitaisiin käyttöönotossa saavuttaa mikäli Linux-pohjaiset järjestelmät muutettaisiin ilmaiselle alustalle. Tällöin kuitenkin tulisi käyttöön kaksi virtualisointiympäristöä, jotka eivät ole keskenään yhteensopivia ja hallinnasta tulisi monimutkaista. Myös ylläpito aiheuttaisi kohonneita kustannuksia niin kauan, kuin henkilöstöä jouduttaisiin kouluttamaan järjestelmän käyttöön. Todennäköisesti VMwaren ratkaisu on kokonaishinnoiteltu. Tällöin mikäli osa paketista karsittaisiin pois, nousisivat jäljelle jäävien palveluiden yksittäishinnat.

OpenVZ olisi kuitenkin mahdollista ottaa käyttöön erilaisissa oppimisympäristöissä ilman suuria muutoksia. Virtualisointi on merkittävä osa-alue palvelimissa. Sen ymmärtäminen on hyödyllistä.

Tietotekniikka-alan insinööriopinnoissa käytettiin esimerkiksi lelux shellpalvelinta. Palvelin oli tarkoitettu Linux-käyttöjärjestelmän opiskeluun ja sitä käyttivät kaikki opiskelijat yhtä aikaa. Mikäli joku opiskelija suoritti ohjelman, joka käytti kaikki palvelimen resurssit, päättyi kaikkien opiskelijoiden työskentely. Mikäli tällainen ympäristö olisi toteutettu niin, että jokainen opiskelija toimisi omassa säiliössään ja

säiliöiden resurssit olisi rajoitettu, olisi epäonnisen opiskelijan toimimaton ohjelma jumiuttanut vain hänen oman säiliönsä.

Samoin joillain opintojaksoilla käytettiin Linux-palvelimia VMware-ympäristössä. GUI:n käyttö ei ollut välttämätöntä. Tällaisen opintojakson resurssivaatimukset olisivat huomattavasti pienemmät, mikäli virtualisointiympäristö olisi toteutettu OpenVZ:llä.

Kokonaisvaltaisesti OpenVZ:aan siirtyminen ei ole mahdollista, mutta sen käyttäminen VMwaren rinnalla joissain triviaaleissa järjestelmissä voisi tuoda arvokasta kokemusta sen toimimisesta Metropolian ympäristössä.

6 Yhteenveto

OpenVZ:n käyttämä virtualisointitapa tarjoaa mielenkiintoisen vaihtoehdon nykyiselle VMwaren hallitsemalle virtualisointialalle. Sen rajoittuneisuus vain Linux-käyttöjärjestelmiin on otettava huomioon ennen järjestelmän hankkimista. Etuna oleva alhaisempi resurssivaatimus on kuitenkin merkittävä.

Järjestelmän käyttöönotto on helppoa kattavan ohjesivuston avulla. Kaikki tarvittavat toimenpiteet on selitetty yksityiskohtaisesti http://wiki.openvz.org-sivustolla. Ilmaiseen versioon on tarjolla myös foorumi käyttötukea varten. Maksullisessa versiossa tukea saa henkilökohtaisesti.

Järjestelmän testaus oli työssä hankalampaa kuin aluksi luulin. Testien tulokset olivat aluksi merkillisiä. Järjestelmän resursseja seuratessa syy tuloksiin kuitenkin selveni. OpenVZ:n työkalupaketissa ei kuitenkaan tule mitään työkalua, jolla voisi seurata reaaliajassa säiliöiden käyttämiä resursseja. Tällainen työkalu olisi helpottanut huomattavasti ongelmien selvittämistä ja syitä, miksi jokin testi antaa epäuskottavia tuloksia.

Säiliöiden hallinnointiin vähäisessä määrin aluksi käytetty hypervm oli ominaisuuksiltaan paljon ilmaista vtonfia parempi. Ohjelman valitettavan maksullisuuden takia, ja koska säiliöt eivät vaatineet erilaisia konfiguraatioita hallinnointiin käytettiin OpenVZ:n mukana tulleita komentorivipohjaisia työkaluja. Mikäli säiliöt olisivat vaatineet erilaisia konfiguraatioita ja niitä olisi tosiasiallisesti pitänyt hallinnoida, olisivat komentorivipohjaiset työkalut olleet riittämättömät. Graafisen käyttöliittymän avulla on kokonaiskuva paremmin esillä. Hallinnointiin siis on kehitetty varsin hyviä työkaluja.

Työssä saavutettiin asetetut tavoitteet, vaikkakaan ne eivät päde mihinkään muuhun kuin kyseisenlaiseen laitteisto- ja ohjelmistokonfiguraation sisältävään laitteistoon. Tuloksista voidaan olettaa, että pelkällä muistin lisäyksellä voitaisiin säiliöitä suorittaa enemmän. Tuloksista ei kuitenkaan voida päätellä, miten vasteajat muuttuisivat, jos palautettava sivu olisi monimutkaisempi.

Työssä olisi voitu selvittää muistin määrän muutoksen todellinen vaikutus. Metropolialla ei kuitenkaan ollut ylimääräisiä muistikampoja työtä varten, joten tätä ei voitu toteuttaa. Tutkimusta voisi jatkaa muuttamalla palautettava sivu vaikkapa php-sivuksi ja muokkaamalla jokainen säiliö palauttamaan eri sivu. Koska jokainen reaalimaailman tapaus on erilainen, ei minkäänlaisella testillä pystytä kiistatta kertomaan suorituskykyä oikeassa tilanteessa. Suuntaa antavat tulokset ovat vain suuntaa antavia, mutta parasta mitä tällaisilla testeillä voidaan saada aikaan.

Metropolian erilaisten palveluvaatimusten takia järjestelmällä ei kyettäisi täysin korvaamaan nykyistä virtualisointiympäristöä. Osittainen korvaaminen voisi tulla kysymykseen, mutta sen mukanaan tuomat ongelmat tulisi ottaa huomioon, mikäli selvitystä tehtäisiin. Järjestelmän käyttöönoton helppouden ja vähäisten resurssivaatimusten takia kokeilu olisi helppo suorittaa jollain vähemmän kriittisellä palvelulla.

Lähteet

1 Platform Virtualization. (WWW-dokumentti.) Wikipedia. <http://en.wikipedia.org/wiki/Platform_virtualization>. Luettu 20.6.2009.

2 Golden, Bernard. Virtualization for Dummies. Hoboken: Wiley Publishing Inc., 2008

3 Golden, Bernard ja Scheffy, Clark. Virtualization for Dummies, Sun and AMD Special Edition. Hoboken: Wiley Publishing Inc., 2008

4 Warren, Steven. Why Virtualize? (WWW-dokumentti.) WebMediaBrands. <http://www.serverwatch.com/tutorials/article.php/3629431>. 30.8.2006. Luettu 24.6.2009.

5 Server Virtualization, Storage Virtualization & Network Virtualization. (WWW-dokumentti.) Virtualtechnologies. <http://www.virtualtechnologies.com/virtualization/whatisvirtualization.html>. Luettu 24.6.2009.

6 Full Virtualization. (WWW-dokumentti.) Wikipedia. <http://en.wikipedia.org/wiki/Full_virtualization>. Luettu 26.6.2009.

7 Liguori, Anthony. Tales of a Code Monkey: The Myth of Type I and Type II Hypervisors. (WWW-dokumentti.) Google. <http://blog.codemonkey.ws/2007/10/myth-of-type-i-and-type-ii-hypervisors.html>. 8.10.2007. Luettu 26.6.2009.

8 Kolyshkin, Kirill. Virtualization in Linux. (WWW-dokumentti) < http://download.openvz.org/doc/openvz-intro.pdf>.OpenVZ. 1.9.2006. Luettu 27.6.2009.

9 OpenVZ Users Guide. (WWW-dokumentti.) OpenVZ. <http://download.openvz.org/doc/OpenVZ-Users-Guide.pdf>. Luettu 27.6.2009.

10 Main Page – OpenVZ Wiki. (WWW-dokumentti.) OpenVZ. <http://wiki.openvz.org/Main_Page>. Luettu 1.7.2009.

11 Quick Installation – OpenVZ Wiki. (WWW-dokumentti.) OpenVZ. <http://wiki.openvz.org/Quick_installation>. Luettu 1.7.2009.

12 OpenVZ. (WWW-dokumentti) Wikipedia. <http://en.wikipedia.org/wiki/OpenVZ>. Luettu 1.7.2009.

13 Padala, Zhu, Wang, Singhal ja Shin. Performance Evaluation of Virtualization Technologies for Server Consolidation. (WWW-dokumentti.) HP.

<http://www.hpl.hp.com/techreports/2007/HPL-2007-59R1.pdf>. 30.8.2008. Luettu 1.7.2009.

14 VMware has 82% share of market of virtualized servers, Microsoft has 13%. (WWW-dokumentti.) IT Facts. <http://www.itfacts.biz/vmware-has-82-share-of-market-of-virtualized-servers-microsoft-has-13/11075>. Luettu 20.7.2009.

15 Virtuozzo. (WWW-dokumentti.) Parallels. <http://www.parallels.com/eu/products/virtuozzo/>. Luettu 20.7.2009.

16 Roadmap. (WWW-dokumentti.) OpenVZ. <http://openvz.org/development/roadmap>. Luettu 20.7.2009.

17 Morgan, Timothy Prickett. Parallels: Bare-metal hypervisor in the works. (WWW-dokumentti.) The Register. <http://www.theregister.co.uk/2009/03/26/parallels_bare_metal/> 26.3.2009. Luettu 20.7.2009.

18 Difference between Redhat and fedora. (WWW-dokumentti.) LinuxQuestions. <http://www.linuxquestions.org/questions/linux-distributions-5/difference-between-rethat-and-fedora-742444/>. Luettu 14.8.2009.

19 Bottleneck in httpd. (WWW-dokumentti.) Kerneltrap. <http://kerneltrap.org/mailarchive/openbsd-misc/2007/5/8/149296>. 8.5.2007. Luettu 15.8.2009.

Liite 1: Sähköpostihaastattelun kysymykset ja vastaukset

Kysymykset

1. Tietohallinnon sivujen mukaan metropoliassa on n.150 fyysistä ja 120 virtuaalipalvelinta. Minkä valmistajien ratkaisuja virtuaalipalvelimissa on käytetty ja miksi on päädytty juuri tähän/näihin tuotteeseen?
2. Mitä käyttöjärjestelmiä palvelimissa on? Minkälainen jakauma niillä on?
3. Mitä palveluja virtuaalipalvelimilla toteutetaan? Mitkä niistä toimivat linuxin päällä?
4. Toteutetaanko useilla fyysisillä palvelimilla samaa palvelua eri tarpeisiin. Onko esimerkiksi useita fyysisiä palvelimia eriytettyinä webservereinä?
5. Minkälaisia käyttöasteita palvelimilla on?

Vastaukset

1) Nykyisin palvelimia palvelinhuoneissa: fyysisiä noin 80kpl ja virtuaalisia 100kpl. Virtualisointijärjestelmä on Vmwaren Infrastructure 3.5 (esx). Aikoinaan kun valitsimme toimittajaa niin vaatimuksena oli, Tuki Windows, Linux, Netware virtuaalikoneille. Lisäksi haluttiin toimiva virtuaalikoneiden online siirto. Tällöin ainoa vaihtoehto oli Vmware. Palvelinten määrä on laskenut, koska entisiä EVTEK:in ja Stadian järjstelmiä on saatu ajettua alas.

2) Virtuaalikoneissa jakauma: Windows 40%, Linux 50% , Novell Netware 10%

3) Sekalainen joukko erilaisia palveluita opetus ja perus infran tarpeisiin
Esimerkiksi, webmailit, moodle,edunix, käyttäjien kotisivut., erilaisia webbi -palvelimia(LAMP)

4) kyllä. Palveluiden eristäminen eri käyttäjäkunnille. (pääsy sallittu palomuurista esim. vain talonsisältä/henkilökunnan koneilta/ määrätyiltä henkilökunnan koneilta). Aika harvoin on tarvetta useammalle palvelimelle kuorman takia. Suoraan mieleen tulee ainoastaan webmail.metropolia.fi, jonka tehtäviä hoitaa 2kpl virtuaalipalvelimia.

5) 0 - > 70% riippuen palvelusta