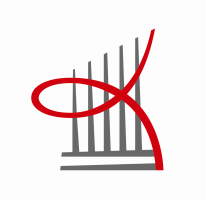


ETÄKÄÄNNÖSJÄRJESTELMÄN VIRTUALISOINTI

Tomi Yritys

Opinnäytetyö
Toukokuu 2011
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

YRITYS, TOMI: Etäkäännösjärjestelmän virtualisointi

Opinnäytetyö 24 s., liitteet 3 s.
Toukokuu 2011

Tässä työssä kerrotaan, kuinka etäkäännösjärjestelmän virtualisointi toteutettiin. Työssä esitellään vaiheet aina virtualisointiympäristön valinnasta sen ohjelmalliseen hallintaan.

Työn sisällön perusteella aiheeseen perehtymättömän lukijan toivotaan saavan peruskäsityksen Xen-virtualisointiympäristön toiminnasta sekä sen etähallinnasta ohjelmointirajapinnan kautta. Työssä pyritään havainnolistamaan lukijalle virtualisoimalla saavutettavia etuja sekä virtualisoinnin tuomia uusia mahdollisuuksia. Ymmärtääkseen ohjelmakoodeja lukijalla tulee olla perusymmärrys ohjelmoinnista sekä käsitys Python-kielen syntaksista.

Työ on toteutettu käyttäen Citrixin XenServerin versiota 5.5.0 ja ohjelmakoodeissa on käytetty Xen Management API:n versiota 1.0.0.

Avainsanat: Virtualisointi, Xen

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer Science
Option of Software Engineering

YRITYS, TOMI: Virtualization for Remote Compile Service

Bachelor's thesis 24 pages., appendices 3 pages.
May 2011

This thesis explains how remote compile service's virtualization was done. It covers steps from selecting virtualization platform to mastering it.

Readers without previous knowledge to the subject should get good introduction to virtualization, and specially how Xen works and how to master it remotely, using the application programming interface. This thesis tries to illuminate some positive aspects and new opportunities that virtualization provides. To fully understand all code samples the reader should know Python's syntax and have some programming experience.

This work was done by using Citrix's XenServer version 5.5.0 and applications are written using Xen Management API version 1.0.0.

Key words: Virtualization, Xen

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Sisällys

Termien ja lyhenteiden luettelo

API	Application Programming Interface. Ohjelmointirajapinta ohjelmien väliseen kommunikointiin.
CoW	Copy-on-Write. Tässä optimointistrategiassa kopioidusta datasta kirjoitetaan vain muuttunut tieto.
I/O	Input/output. I/O:ksi lasketaan kaikki prosessorin ja muistin välisen kommunikoinnin ulkopuolinen kommunikointi, esimerkiksi verkkoliikenne tai levylle kirjoitus.
Qt	Kehitysympäristö alustariippumattomaan ohjelmakehitykseen.
Qt Mobility API	Kokoelma ohjelmointirajapintoja helpottamaan Qt-ohjelmakehitystä mobiilialustoille.
Scratchbox	Maemon sovelluskehitykseen suunniteltu kehitysympäristö.
SDK	Software Development Kit. Kokoelma kehitystyökaluja ja rajapintoja ohjelmien kehitykseen jollekin tietylle alustalle.
SSH	Secure Shell. Salattuun tietoliikenteeseen käytetty protokolla.
Virtualisointi	Yksi fyysinen resurssi näkyy useina loogisina resursseina.
VNC	Virtual Network Computing. Graafisen työpöydän etähallintaan kehitetty teknologia.
VPN	Virtual private network. Tällä tekniikalla voidaan julkisen verkon kautta voidaan muodostaa turvattu yhteys lähiverkkoon.
Xen	Avoimen lähdekoodin virtalisointialusta.
XML	Extensible Markup Language. Yleisesti käytetty kuvauskieli järjestelmien väliseen tiedonsiirtoon.
XML-RPC	Extensible Markup Language-Remote Procedure Call. Protokolla jossa XML muotoiltua dataa siirretään HTTP-siirtoprotokollaa käyttäen.

1 Johdanto

Tässä työssä tutustutaan etäkäännösjärjestelmän tuotteistamisen yhteydessä toteutettuun virtualisointiratkaisuun, jolla pyrittiin saavuttamaan helpommin hallittava ja skaalautuva ympäristö aiemmin fyysisillä koneilla toteutettuun verrattuna. Kyseinen etäkäännösjärjestelmä on pilvipalvelu, johon käyttäjät voivat lähettää Qt-ohjelmakoodinsa käännettäväksi eri mobiilialustoille. Käännösympäristö koostuu koneista, jotka hoitavat käännöksen Maemolle sekä eri versioille Symbian käyttöjärjestelmästä. Tässä työssä ei käsitellä varsinaiseen käännösten toteutukseen liittyviä ohjelmakoodeja tai arkkitehtuurista toteutusta.

Palvelun tarkoituksena on tarjota käyttäjille mahdollisuus käyttöjärjestelmästä tai laitteesta riippumatta kehittää Qt-ohjelmiaan mobiilialustoille. Käyttäjän ei tarvitse asentaa omalle työasemalleen isoja kehitysympäristöjä eikä ylläpitää niitä.

Toisessa luvussa tutustutaan virtualisointiin ja sen tuomiin etuihin. Lisäksi tutustutaan työssä käytettyyn Xen-virtualisointiympäristöön ja sen toimintaan sekä esitellään ominaisuuksia, joiden johdosta Xen valittiin alustaksi virtualisoinnille.

Kolmannessa luvussa tehdään yleiskatsaus käännösympäristöön asennettujen koneiden luonnista. Luvussa käydään läpi käyttöjärjestelmien ja tarvittavien sovellusten asennus koneille, jotta koneet toteuttavat niille asetetut määritykset.

Työn neljännessä luvussa perehdytään käännösympäristön hallintaan. Luvussa demonstroidaan koodiesimerkkien avulla miten ympäristöön luodaan uusi käännöskone tai tuhotaan vanha.

2 Virtualisointi

Virtualisoinnilla tarkoitetaan tässä yhteydessä useiden käyttöjärjestelmien suorittamista samalla koneella. Virtualisointi mahdollistaa kulloisen tarpeen mukaan muuttuvan ja helposti hallittavan kokonaisuuden. Tämä mahdollistaa resurssien paremman hyödyntämisen, koska ne voidaan aina ohjata sinne, missä niitä tarvitaan.

Valittaessa virtualisointiympäristöä tärkeimpinä vaatimuksina olivat järjestelmän luotettavuus sekä laaja tuki virtualisoitaville käyttöjärjestelmille. Useista tarjolla olevista vaihtoehdoista suurin osa karsiutui pois suppean arkkitehtuuri- tai käyttöjärjestelmätuen johdosta. Hyvin nopeasti päädyttiin valitsemaan kahden suositun ympäristön, Xenin ja VMwaren, välillä. Järjestelmän avoimuus oli lopulta ominaisuus, jonka johdosta Xen valittiin toteutuksen alustaksi.

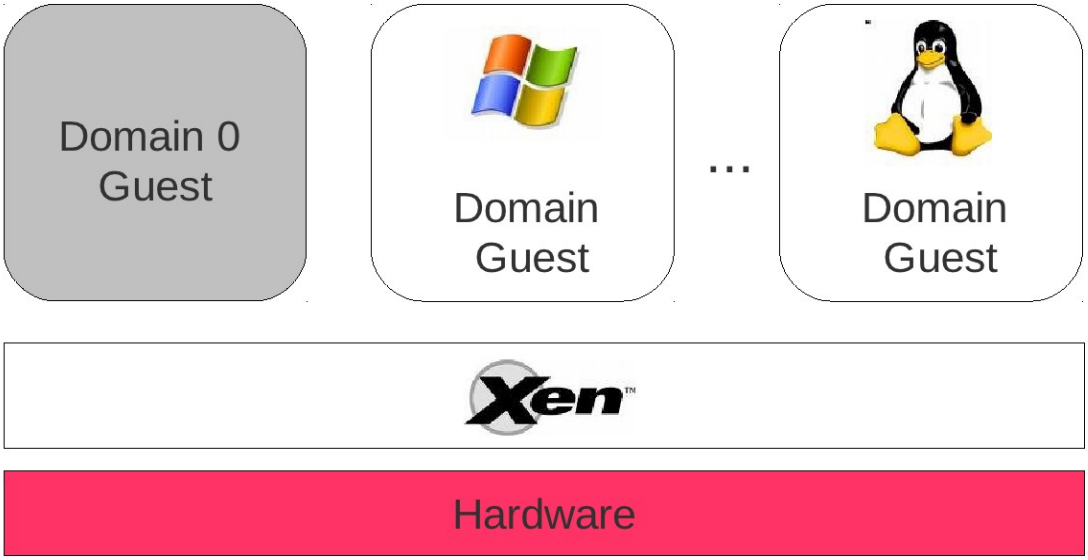
2.1 Xen

Xen on alun perin Cambridgen yliopiston kehittämä virtualisointialusta. Vuodesta 2010 sitä on kehitetty avoimen lähdekoodin GNUn (= General Public License) (GPLv2) lisenssillä (Wikipedia: Xen 2011). Se sopii hyvin alustaksi tälle työlle, kosta sen tukemien käyttöjärjestelmien kirjo on hyvin laaja. Sillä voidaan virtualisoida muun muassa Microsoft Windows, Linux ja Open Solaris -käyttöjärjestelmiä sekä eri versiota BSD:stä (Wikipedia: Xen 2011). Lisäksi Xen mahdollistaa palvelinraudan tehokkaan hyödyntämisen, sille se toimii hyvin ohuena kerroksena fyysisen laitetason ja käyttöjärjestelmän välissä (How Does Xen Work 2009, 3).

Xen on saanut nopeassa ajassa vakaan aseman virtualisointimarkkinoilla. Monet alan isot toimijat, mm. Amazon ja Cloud.com, käyttävät sitä pilvipalvelintensa virtualisoinnissa. Lisäksi siihen pohjautuvia kaupallisia ratkaisuja tarjoavat monet alalla tunnetut yritykset, muun muassa Cisco, Citrix, Fujitsu, Lenovo ja Oracle. (Spector & Xen.org yhteisö 2011, 5.)

Xen toimii hyvin ohuena kerroksena laitetason päällä. Se huolehtii ainoastaan prosessien aikataulutuksesta (eng. scheduling) sekä muistin jakamisesta virtuaalikoneille. Se ei ole tietoinen koneen verkosta, ulkoisista levyasemista tai mistään muistakaan I/O (input/output) -toiminnallisuuksista. Luonnollisesti järjestelmässä täytyy olla jokin taho, jolla on pääsy fyysisiin I/O -resursseihin. Xenissä tätä konetta kutsutaan domain 0 -ko-

neeksi. Kuviossa 1 on esitetty Xen ympäristön karkean tason rakenne. (How Does Xen Work 2009, 3–5.)



KUVIO 1: Xen toimii ohuena kerroksena käyttöjärjestelmien ja laitetason välissä. (How Does Xen Work 2009, 3)

2.2 Domain 0

Domain 0, tai dom0 (tästä eteenpäin käytetään termiä dom0) on yksittäinen virtuaalikone, jossa ajetaan tähän tarkoitukseen muokattua Linux kerneliä. Sen lisäksi, että sillä on pääsy fyysisiin I/O -resursseihin, se pystyy myös kommunikoimaan muiden virtuaalikoneiden kanssa. Xenissä näitä muita koneita kutsutaan domain U tai domU -koneiksi (tästä eteenpäin käytetään termiä domU). (How Does Xen Work 2009, 3–5.)

Dom0-koneeseen on asennettu ajurit verkon- ja levynhallintaa varten. Se kommunikoi suoraan laitteiston kanssa ja käsittelee domU-koneiden kutsut verkkoon sekä hoitaa levyiltä luvun ja niille kirjoituksen (How Does Xen Work 2009, 8). Tässä työssä ei keskitytä dom0-koneisiin, koska niiden asennus ja ylläpito kuuluu tässä järjestelmässä palvelinten ylläpitäjän tehtäviin.

2.3 Domain U

Suurin ero dom0- ja domU-koneiden välillä on se, että domU-koneella ei ole suoraa yhteyttä laitteistoon. Kaikki domU-koneen I/O-kutsut välitetään dom0-koneelle. Xen tarjoaa kaksi mahdollisuutta domU-koneiden virtualisointiin. Kone voidaan virtualisoida joko kokonaan tai osittain. (How Does Xen Work 2009, 3–5.)

Osittaisvirtualisoinnissa (PV, eng. Paravirtualization) virtuaalikone on tietoinen olevansa virtualisoitu, eikä täten ole oikeutettu kommunikoimaan suoraan laitteiston kanssa. Osittaisvirtualisoidun koneen käyttöjärjestelmään on sisällytetty ajurit Xenin verkkoa ja levynhallintaa varten. Ajurit kommunikoivat dom0-koneen ajurien kanssa. (How Does Xen Work 2009, 4–5.)

Kokonaan virtualisoidulla koneella (HVM, eng. hardware-assisted virtualization) ajetaan käyttöjärjestelmää ilman muutoksia. Jotta käyttöjärjestelmä voidaan käynnistää ilman muutoksia, simuloi Xen virtual firmware -ohjelma käyttöjärjestelmälle biosin toimintaa. Xen käyttää dom0-koneella QEMU-taustaprosessia (qemu-dm, QEMU device manager) hoitamaan kokonaan virtualisoitujen koneiden verkkoa ja levynhallintaa. (How Does Xen Work 2009, 5–7.)

Virtualosoimalla kone vain osittain saavutetaan parempi suorituskyky kuin kokonaan virtualisoimalla, mutta se vaatii muutoksia käyttöjärjestelmään. Linux kernel on tukenut osittaisvirtualisointia jo versiosta 2.6.23 asti. Windows-käyttöjärjestelmille ei ole osittaisvirtualisoinnin virallista tukea. (Wikipedia: Xen.)

Myöhemmässä vaiheessa puhuttaessa virtuaalikoneista, tarkoitetaan domU-koneita.

2.4 Nopea kopiointi

Xenissä on tuki nopealle domU-koneen kopioinnille (eng: Fast Clone). Tämä mekanismi käyttää hyödykseen levyjärjestelmän CoW (Copy on Write) ominaisuutta, jossa levylle kirjoitetaan ainoastaan muuttunut tieto. Tämän ansiosta koneen kopiointi on hyvin nopeaa. Toinen merkittävä hyöty saavutetaan siinä, että levytilaa tarvitaan huomattavasti vähemmän verrattuna todelliseen datamäärän kaksinkertaistamiseen. (XenServer Virtual Machine Installation Guide 2011.)

Tässä työssä käytettyjen käännöskoneiden keskimääräinen elinikä on hyvin lyhyt, ja muutoksia levylle kirjoitetaan pääasiassa vain käännösaikana. Koska kone levyineen tuhotaan heti käännöksen päätyttyä, ei koneiden määrän kasvattaminen juurikaan kasvata levytilan tarvetta. Käännöskoneen vaatima tila levyllä on keskimäärin joitain megatavuja.

Edellä mainitut ominaisuudet, jotka saavutetaan nopean kopioinnin avulla, olivat tämän työn kannalta kynnyskysymyksiä. Ilman nopeaa kopiointia usean gigatavun kokoisten

levykuvien kopiointi vaatisi huomattavan määrän tilaa ja saattaisi asettaa rajoja koneiden määrälle, puhumattakaan ajasta jota tähän kopiointiin kuluisi.

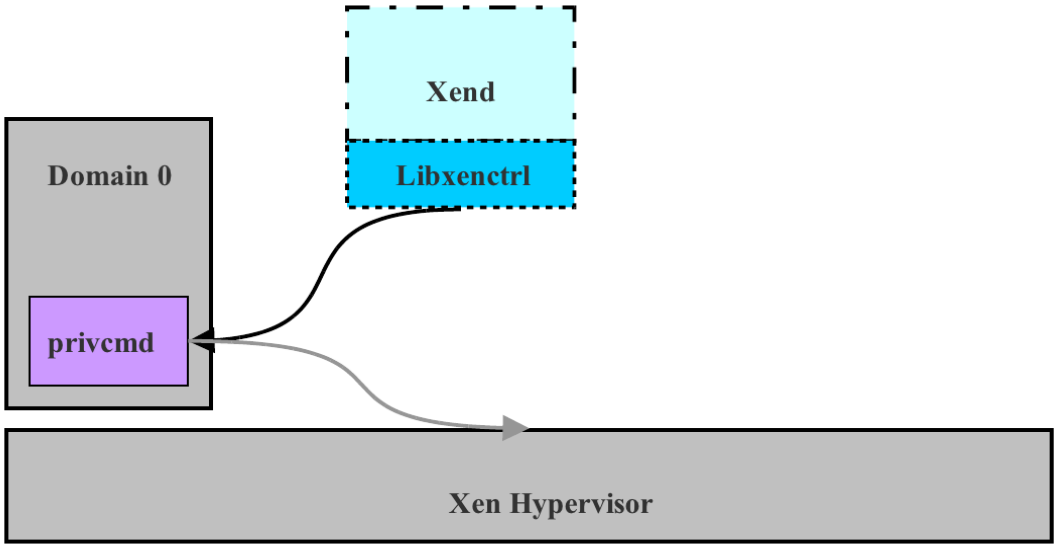
2.5 Xen-ohjelmointirajapinta

Xen Management API (Application Programming Interface) on Xen-virtuaalikoneiden etähallintaa varten toteutettu rajapinta. Rajapintakutsut perustuvat XML-RPC-protokollaan (Mellor, Sharp, Scott 2007, 4). XML-RPC-protokollassa XML-muotoillut kutsut siirretään HTTP -siirtotietä käyttäen (Wikipedia: XML-RPC 2011). Protokollaan tarjotaan sidokset Pythonille, C:lle ja Javalle (Xen Wiki 2011). Tässä työssä Xen hallinta on toteutettu käyttäen Python-ohjelmointikieltä ja Xenin API-kirjastoa.

Kutsut tehdään dom0-koneelle, joka välittää kutsut eteenpäin Xenille. Rajapinnan kautta voidaan toteuttaa kaikki tässä työssä tarvitut koneiden hallintaan liittyvät toiminnot, esimerkiksi koneiden käynnistys, kopiointi ja tuhoaminen.

Koska kutsut noudattavat standardia XML-RPC-protokollaa, hallinnan voi toteuttaa millä tahansa kielellä ilman, että kyseiselle kielelle olisi julkaistu omaa kirjastoa tätä varten. Käytettäessä virallista kirjastoa voidaan kuitenkin varmistaa rajapintaversiosta riippumaton toiminnallisuus.

Kuviossa 2 havainnollistetaan, kuinka dom0-kone käsittelee sille saapuneet kutsut. Xend on Pythonilla ohjelmoitu taustaprosessina ajettava ohjelma, joka ottaa vastaan XML-RPC-kutsut. Libxenctrl on C-kirjasto, jonka avulla kutsut välitetään privcmd-ajurin kautta Xenille. (How Does Xen Work 2009, 6.)



KUVIO 2: Xen API (How Does Xen Work 2009, 6)

3 Käännösympäristön valmistelu

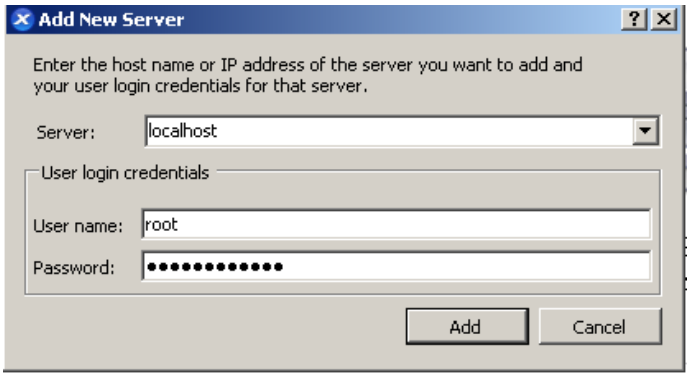
Käännösympäristöllä tarkoitetaan tässä kokonaisuutta, joka koostuu Xenissä virtualioiduista koneista. Nämä koneet hoitavat varsinaisen käännöstyön. Jokainen käännöskone koostuu käyttöjärjestelmästä, kehitystyökaluista sekä ohjelmakoodeista, jotka hoitavat käännöksen suorituksen. Tässä työssä käännösympäristöön luotiin käännöskoneet Maemo- ja Symbian-käyttöjärjestelmille. Kommunikointi käännöskoneiden ja käännöksiä hallinoivien palvelinten välillä on toteutettu http-rajapintaa käyttäen.

3.1 Koneiden luonti

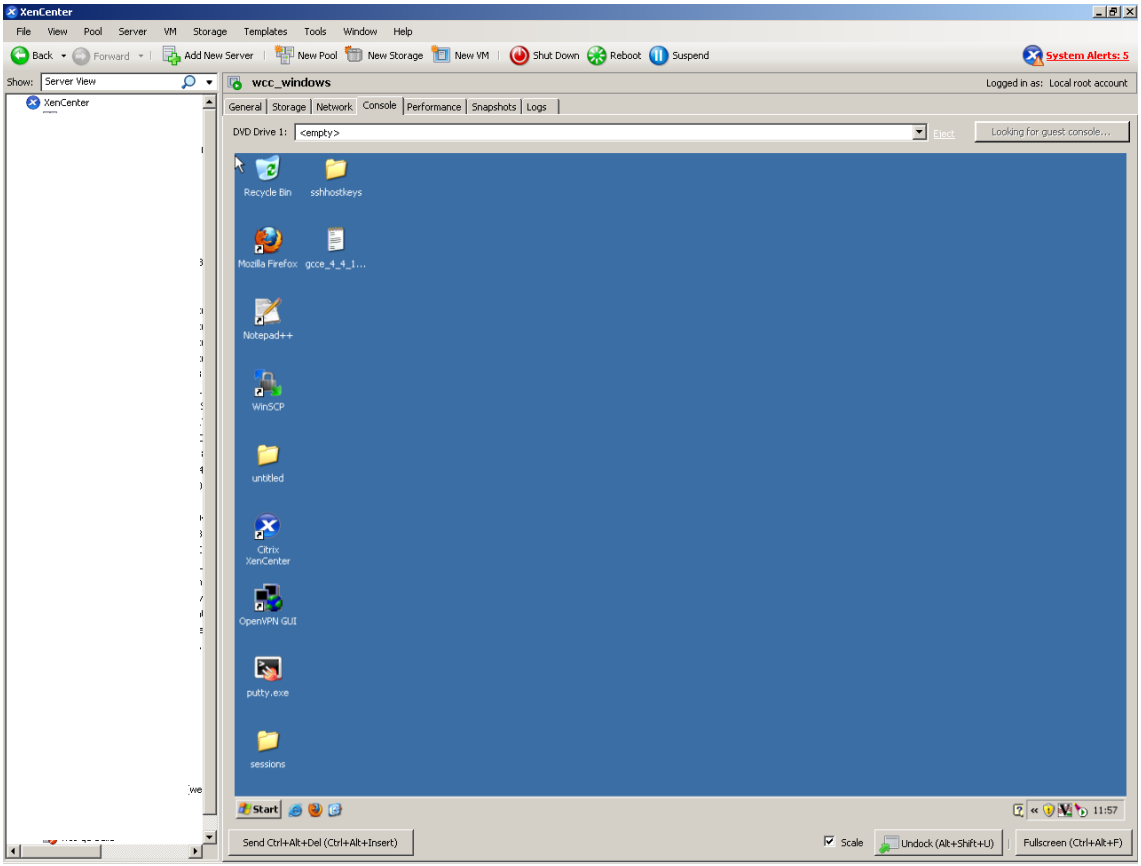
Citrixin XenCenter on Xenin hallintaan tarkoitettu graafinen työkalu Windows-käyttöjärjestelmille. Se mahdollistaa muun muassa koneiden luonnin, käyttöjärjestelmän asennuksen sekä virtuaalikoneiden resurssien hallinnan. Tässä työssä virtuaalikoneet luotiin käyttäen XenCenteriä.

Jotta XenCenterillä voidaan hallinnoida Xen-ympäristöä, täytyy sillä ensiksi luoda yhteys dom0-koneelle. Kuviossa 3 on näkymä uuden yhteyden avaukseen. Dom0-koneelle kirjaudutaan paikallisilla tunnuksilla, joilla on oikeus Xenin hallintaan. Koska tässä tapauksessa dom0-koneella ei ole tietoturvasyistä suoraa yhteyttä julkiseen verkkoon, koneiden luontia varten täytyi muodostaa yhteys lähiverkkoon. Tämä toteutettiin asennusvaiheessa VPN-yhteyden avulla. Myöhemmässä vaiheessa hallintaa varten luotiin SSH-porttiohjaus palvelimen kautta, jolla on yhteys sekä julkiseen verkkoon että dom0-ko-

neen verkkoon. Myös XenCenter käyttää Xenin hallintaan XML-RPC-kutsuja, joten kutsut menevät portin 443 kautta. Porttiohjauksen avulla ei kuitenkaan ole mahdollista saada kuvaa virtuaalikoneelta kuviossa 4 esitettyyn näkymään, jolloin käyttöjärjestelmän asennus on mahdotonta (Citrix Forums 2010).



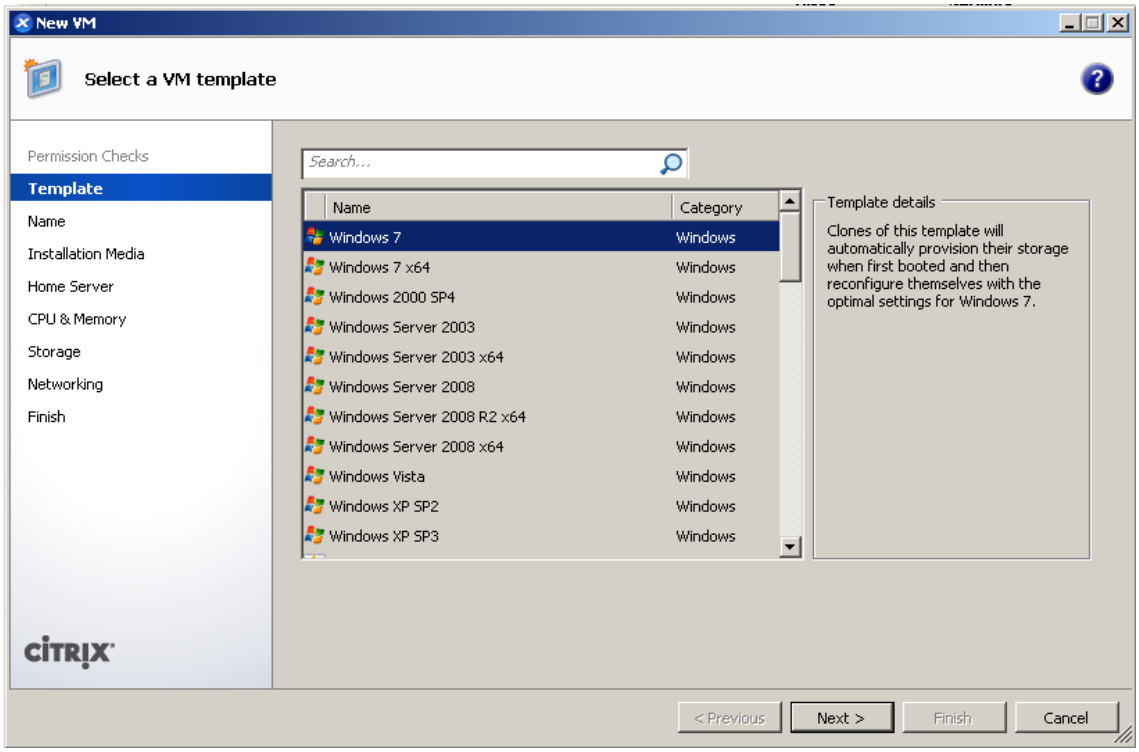
KUVIO 3: Yhteyden luonti XenCenterissä.



KUVIO 4: XenCenterin näkymä virtuaalikoneeseen.

Kuvio 5 on XenCenterin uuden koneen luonnista. Koneen luontia varten on tehty hyvin helppokäyttöinen velho. Ensiksi luodaan kone halutuilla resursseilla, minkä jälkeen siirrytään käyttöjärjestelmän asennukseen.

Kuten aiemmin on mainittu, kykenee Xen virtualisoimaan laajaa kirjoa käyttöjärjestelmiä. Tämän ansiosta pystyttiin käyttöjärjestelmä valitsemaan muiden ominaisuuksien perusteella. Symbian käännöksille käyttöjärjestelmäksi valittiin Windows XP ja Maemolle Debian GNU Linux. Kumpaankin käyttöjärjestelmään asennettiin ”guest agent” -työkalu, joka tarjoaa Xenille tietoa koneesta, esimerkiksi sen IP-osoitteen ja muistin käytön.



KUVIO 5: XenCenterin uuden koneen luonti-ikkuna.

3.2 Ohjelmiston asennus

Kumpaankin käyttöjärjestelmään asennettiin käännöksen toteutuksessa tarvittavia apuohjelmia. Käännöstöiden vastaanottoa varten asennettiin Apache http-palvelinohjelma PHP-laajennoksella. Lisäksi Windows-koneelle asennettiin etähallintaa helpottamaan TightVNC. Se on avoimen lähdekoodin ohjelma, joka mahdollistaa koneen etähallinnan laajennettua VNC-protokollaa käyttäen (Wikipedia: TightVNC 2011). Debianin etähallintaa varten koneelle asennettiin SSH-palvelin.

Windows koneelle asennettiin Symbian S60 SDK:n versiot:

- 3rd Edition, Feature Pack 1.
- 3rd Edition, Feature Pack 2.

- 5th Edition
- 5th Edition, N97 SDK

SDK versioiden asennuksen jälkeen koneesta ei puutu enää muuta kuin Qt-kirjastot. Tarkoitus on tarjota käyttäjille useita versioita Qt-kirjastoista yhdisteltynä eri versioihin Mobility API:sta. Koska järjestelmä haluttiin pitää mahdollisimman yksinkertaisena ja ylläpidettävänä, jokaiselle eri versiolle tehtiin oma koneensa. Tämä toteutettiin luomalla asennetusta koneesta uusi kopio, johon kirjastot asennettiin.

Debian koneelle asennettiin Scratchbox-pohjainen kehitysympäristö Maemo-käännöksiä varten. Sen kanssa noudatettiin samaa periaatetta kuin ennenkin, eli jokaisesta uudesta Qt-kirjastoversiosta luodaan uusi kone.

4 Käännösympäristön hallinta

Käännösympäristön aikaisemmassa versiossa siivottiin käännösaikana syntyneet tiedostot ennen seuraavaa käännöstä. Hyvin aikaisessa vaiheessa todettiin, että tällä menetelmällä ei voida taata käännösympäristön puhtautta ilman, että käyttäjän toimia käännösaikana rajoitetaan merkittävästi. Koska tavoitteena on tarjota käyttäjälle vapaat kädet ja turvattu ympäristö, päädyttiin siihen, että jokaisen käännöksen jälkeen siihen käytetty kone on tuhottava, ja tilalle on käynnistettävä uusi kone. Virtualisoinnin ansiosta tämä oli toteutettavissa.

Xenin hallinta on toteutettu Python-kirjastoa käyttäen erillisillä pienillä ohjelmilla, joista jokainen toteuttaa tietyn tehtävän. Käsittelemme tässä tarkemmin ohjelmat, joista toisella voidaan käynnistää uusi käännöskone, toisella tuhota. Ohjelmien lähdekoodit löytyvät kokonaisuudessaan liitteestä 1.

4.1 Yhteyden avaus dom0-koneelle

Kaikki ohjelmat alkavat yhteyden avauksella dom0-koneelle. Esimerkissä 1 on näytetty, kuinka istunnon luonti ja kirjautuminen hoidetaan.

ESIMERKKI 1: Yhteyden muodostaminen dom0-koneelle

```
session = XenAPI.Session(url)
session.xenapi.login_with_password(username, password)
```

Esimerkissä 2 on esitetty uuden koneen käynnistyksessä käytettyä pääohjelmaa. Kaikkien ohjelmien pääohjelmat ovat hyvin samankaltaisia. Niissä käsitellään ohjelmalle annetut parametrit ja luodaan istunto, minkä jälkeen kutsutaan varsinaisen toiminnallisuuden toteuttavaa funktiota.

ESIMERKKI 2: Startup.py pääohjelma

```
if __name__ == "__main__":
  if len(sys.argv) <> 6:
    print "Usage:"
  print sys.argv[0], "<VM_template_name> <new_vm_name> <url>
<username> <password>"
    sys.exit(1)
  vm_template_name = sys.argv[1]
  vm_new_name = sys.argv[2]
  url = sys.argv[3]
  username = sys.argv[4]
  password = sys.argv[5]
  trycount = 0
  while trycount < 5:
    try:
     session = XenAPI.Session(url)
     session.xenapi.login_with_password(username, password)
     sys.exit(main(session, vm_template_name, vm_new_name))
    except SystemExit:
     break
    except Exception:
     print "it was try n:o ", trycount
     trycount = trycount + 1
  raise
  sys.exit(1)
```

Suoritettaessa Xenille kuormitustestiä huomattiin, että kovassa rasituksessa yhteyden avaukset eivät aina onnistu. Tästä syystä koodissa yritetään yhteyden luontia viisi kertaa ennen poistumista. Tämän jälkeen ei ongelmia ole enää ilmennyt.

4.2 Koneen luonti ja käynnistys

Jokainen uusi kone on kopio halutusta käännöskoneesta. Esimerkissä 3 on esitetty funktio, jolle annetaan parametreina Xen-istunto, kopioitavan koneen nimi sekä nimi uudelle koneelle. Yksinkertaistettuna koodissa toimitaan seuraavasti:

1. Haetaan kaikki Xenissä olevat koneet.
2. Etsitään kopioitava kone.
3. Luodaan kopio.
4. Käynnistetään kopio.

ESIMERKKI 3: Startup.py main-funktio

```
def main(session, vm_template_name, vm_new_name):
  # Find a non-template VM object
  vms = session.xenapi.VM.get_all()
  for vm in vms:
    record = session.xenapi.VM.get_record(vm)
    #we want to copy a wm so..
  if not(record["is_a_template"]) and
not(record["is_control_domain"]):
      name = record["name_label"]
      if name == vm_template_name:
        print "Found VM uuid", record["uuid"], "called: ", name
        record = session.xenapi.VM.get_record(vm)
        # Make sure the VM has powered down
        #if templateVm is not halted it might be under construction
        #(shouldn't never happen)
        if record["power_state"] != "Halted":
          print "Template is under construction, try again laiter"
          session.xenapi.session.logout()
          return 1
        print "clonig"
        #clone
        cloneVm = session.xenapi.VM.clone(vm, vm_new_name)
        print " starting cloned vm"
        #start(vm, start_paused, force)
        session.xenapi.VM.start(cloneVm,False,True)
        session.xenapi.session.logout()
        return 0

  print "No VM called: ", vm_template_name
  session.xenapi.session.logout()
  return 1
```

Kopioitavan koneen tulee olla sammutettuna, jotta kloonaus voidaan toteuttaa (Mellor, Sharp & Scott 2007, 26). Käynnistyttyään kone ilmoittaa käännöksiä hallinoiville palvelimille DHCP-palvelimelta saamansa IP-osoitteen. Virtuaalikone ei ole tietoinen omasta identiteetistään Xenissä, joten käynnistynyttä konetta ei voida yhdistää mihinkään tiettyyn käynnistyskutsuun. Tästä syystä koneet tunnistetaan jatkossa tämän IP-osoitteen perusteella.

4.3 Koneen sammutus ja tuhoaminen

Käännöstyön valmistuttua siihen käytetty kone tuhotaan. Jotta oikea kone löydetään, täytyy virtuaalikoneen IP-osoite selvittää. Tätä varten toteutettu funktio on esitetty esimerkissä 4.

ESIMERKKI 4: Funktio koneen IP-osoitteen selvitykseen.

```
def read_ip_address(vm):
    vgm = session.xenapi.VM.get_guest_metrics(vm)
    try:
        os = session.xenapi.VM_guest_metrics.get_networks(vgm)
        if "0/ip" in os.keys():
            return os["0/ip"]
        return None
    except:
        return None
```

Kun oikea kone on löydetty, se sammutetaan ja tuhotaan. Xen muistaa myös sammutettujen koneiden IP-osoitteet. Ettei väärää konetta poistettaisi, tarkistetaan esimerkin 5 alussa, onko kone käynnissä. Virtuaalikoneen lisäksi pitää tuhota myös koneelta orvoiksi jääneet virtuaalilevyasemat (VDI, eng: Virtual Disk Image). Tämä on esitetty esimerkissä 6.

ESIMERKKI 5: Koneen sammutus ja poisto.

```
#we don't want to destroy our template so we check is the machine
#running
#..if is, continue searching
if vm_record["power_state"] != "Running":
    print "  ... founded one vm with ip: ", vm_ip, "(", name, ")
that ain't running (template)"
    continue
#..else destroy vm
print "  ... shutting down"
session.xenapi.VM.hard_shutdown(vm)
print "  ... destroying vm"
session.xenapi.VM.destroy(vm)
```

ESIMERKKI 6: Virtuaalikoneen levyasemien tuhoaminen.

```
all_vdis = session.xenapi.VDI.get_all_records()
all_vbds = session.xenapi.VBD.get_all_records()
for vbd in vm_record["VBDs"]:
  vbd_record = all_vbds[vbd]
  if vbd_record["type"] == "Disk":
    # destroy corresponding vdi
    vdi_record = all_vdis[vbd_record["VDI"]]
    vdi = session.xenapi.VDI.get_by_uuid(vdi_record["uuid"])
    print "Destroying VDI snapshot: ", vdi_record["name_label"],
vdi_record["uuid"]
    session.xenapi.VDI.destroy(vdi)
    break
```

5 Yhteenveto

Virtualisointi tarjoaa hyvät edellytykset tämän tapaisen toiminnallisuuden pyörittämiseen. Sen ansiosta järjestelmästä voitiin rakentaa helposti ylläpidettävä ja kulloisenkin tarpeen mukaan skaalautuva.

Virtualisointialustaksi valittu Xen on osoittautunut toimivaksi valinnaksi. Se on avoimen lähdekoodin virtualisointiympäristö, jota kehitetään aktiivisesti. Tämän työn puitteissa se on ehtinyt vakuuttaa vakaudellaan ja tehokkuudellaan. Puutteitakin ohjelmasta löytyi. Työssä käytetty versio 5.5.0 ei tukenut käynnissä olevan koneen kloonausta, joten uuden koneen käynnistyksessä hukataan resursseja käyttöjärjestelmän käynnistämiseen. Tässä onkin yksi selkeä kehityskohta tulevaisuutta varten, kunhan tämä ominaisuus Xenin seuraaviin versioihin saadaan.

Xenin hallinta sen kehitysrajapinnan avulla oli tehty helpoksi. Rajapinta on monipuolinen ja se on dokumentoitu kiitettävän hyvin. Sitä käyttäen on helppo toteuttaa ohjelmia, joilla voidaan automatisoida ympäristössä ajettavien koneiden toimintaa.

Tulevaisuuden optimointimahdollisuudet, aiemmin mainitun kloonauksen lisäksi, kohdistunevat pitkälti oikeanlaisen raudan valintaan. Koska käännösprosessissa on huomattavan paljon I/O-toimintaa, eritoten Symbianin tapauksessa, hakuajaltaan nopealla massamuistilla saattaisi olla positiivinen vaikutus käännösaikojen lyhenemiseen. Hallintaan käytettyjen ohjelmien optimoinnilla taasen ei kokonaiskäännösajan kannalta ole havaittavaa merkitystä.

LÄHTEET

Citrix Forums. 2010. XenCenter Console Tab not working. Luettu 27.4.2011. http://forums.citrix.com/message.jspa?messageID=1012169.

Citrix. 2009. How Does Xen Work. [Versio 1.0]. Tulostettu 14.3.2011. http://www.xen.org/files/Marketing/HowDoesXenWork.pdf.

Citrix. 2011. XenServer Virtual Machine Installation Guide. Cloning an existing VM. [Versio 5.0.0]. Tulostettu 12.4.2011. http://docs.vmd.citrix.com/XenServer/5.0.0/1.0/en_gb/guest.html#creatingVMs-clone.

Mellor, E., Sharp, R. & Scott, D. 2007. Xen Management API. [Versio 1.0.0]. Tulostettu 14.3.2011. http://wiki.xensource.com/xenwiki/XenApi?action=AttachFile&do=get&target=xenapi-1.0.0.pdf.

Spector, S. & Xen.org yhteisö. 2011. Why Xen. Tulostettu 14.3.2011. http://www.xen.org/files/Marketing/WhyXen.pdf.

Wikipedia. 2011. TightVNC. Luettu 27.4.2011. http://en.wikipedia.org/wiki/TightVNC.

Wikipedia. 2011. Xen. Luettu 14.3.2011. http://en.wikipedia.org/wiki/Xen.

Wikipedia. 2011. XML-RPC. Luettu 27.4.2011. http://en.wikipedia.org/wiki/XML-RPC.

Xen Wiki. 2011. XenApi. Luettu 11.5.2011. http://wiki.xensource.com/xenwiki/XenApi.

Liitteet

Liite 1: Virtuaalikoneen hallinnointiin käytettyjen ohjelmien lähdekoodit.

startup.py LIITE 1: 1 (3)

```
# -*- coding: utf-8 -*-
import sys
import XenAPI

def main(session, vm_template_name, vm_new_name):
  # Find a non-template VM object
  vms = session.xenapi.VM.get_all()
  for vm in vms:
    record = session.xenapi.VM.get_record(vm)
    #we want to copy a wm so..
    if not(record["is_a_template"]) and not(record["is_control_domain"]):
      name = record["name_label"]
      if name == vm_template_name:
        print "Found VM uuid", record["uuid"], "called: ", name
        record = session.xenapi.VM.get_record(vm)
        # Make sure the VM has powered down
        #if templateVm is not halted it might be under construction
        if record["power_state"] != "Halted":
          print " ... Template is under construction, try again laiter"
          return 1
        print " ... clonig"
        #clone
        cloneVm = session.xenapi.VM.clone(vm, vm_new_name)
        print " starting cloned vm"
        #start(vm, start_paused, force)
        session.xenapi.VM.start(cloneVm,False,True)
        session.xenapi.session.logout()
        return 0
  print "No VM called: ", vm_template_name
  session.xenapi.session.logout()
  return 1

if __name__ == "__main__":
  if len(sys.argv) <> 6:
    print "Usage:"
    print sys.argv[0], "<VM_template_name> <new_vm_name> <url> <username>
<password>"
    sys.exit(1)
  vm_template_name = sys.argv[1]
  vm_new_name = sys.argv[2]
  url = sys.argv[3]
  username = sys.argv[4]
  password = sys.argv[5]
  trycount = 0
  while trycount < 5:
    try:
      session = XenAPI.Session(url)
      session.xenapi.login_with_password(username, password)
      sys.exit(main(session, vm_template_name, vm_new_name))
    except SystemExit:
      break
    except Exception:
      print "it was try n:o ", trycount
      trycount = trycount + 1
  raise
  sys.exit(1)
```

(jatkuu)

shutdown.py LIITE 1: 2 (3)

```
# -*- coding: utf-8 -*-
import sys
import XenAPI

def read_ip_address(vm):
  vgm = session.xenapi.VM.get_guest_metrics(vm)
  try:
    os = session.xenapi.VM_guest_metrics.get_networks(vgm)
    if "0/ip" in os.keys():
      return os["0/ip"]
    return None
  except:
    return None

def main(session, vm_ip):
  # Get records
  all_vms = session.xenapi.VM.get_all_records()
  all_vdis = session.xenapi.VDI.get_all_records()
  all_vbds = session.xenapi.VBD.get_all_records()
  for vm in all_vms:
    ip = read_ip_address(vm)
    vm_record = session.xenapi.VM.get_record(vm)
    name = vm_record["name_label"]
    if ip == vm_ip:
      print "Found VM uuid", vm_record["uuid"], "called: ", name


      #we don't want to destroy our template so we check is the machine
running
      #..if is, continue searching
      if vm_record["power_state"] != "Running":
        print "  ... founded one vm with ip: ", vm_ip, "(", name, ") that
ain't running (template)"
        continue
      #..else destroy vm
      print "  ... shutting down"
      session.xenapi.VM.hard_shutdown(vm)
      print "  ... destroying vm"
      session.xenapi.VM.destroy(vm)

      #we also want to remove all vm's disks so..
      # loop through this vm's vbds (disks only)
      for vbd in vm_record["VBDs"]:
        vbd_record = all_vbds[vbd]
        if vbd_record["type"] == "Disk":
          # destroy corresponding vdi
          vdi_record = all_vdis[vbd_record["VDI"]]
          vdi = session.xenapi.VDI.get_by_uuid(vdi_record["uuid"])
          print "Destroying VDI snapshot: ", vdi_record["name_label"],
vdi_record["uuid"]
          session.xenapi.VDI.destroy(vdi)
          break
      session.xenapi.session.logout()
      return 0
  print "No VM with ip: ", vm_ip
  session.xenapi.session.logout()
  return 1
```

(jatkuu)

LIITE 1: 3 (3)

```
if __name__ == "__main__":
  if len(sys.argv) <> 5:
    print "Usage:"
    print sys.argv[0], "<VM-ip> <url> <username> <password>"
    sys.exit(1)
  vm_ip = sys.argv[1]
  url = sys.argv[2]
  username = sys.argv[3]
  password = sys.argv[4]
  trycount = 0
  while trycount < 5:
    try:
      session = XenAPI.Session(url)
      session.xenapi.login_with_password(username, password)
      sys.exit(main(session, vm_ip))
    except SystemExit:
      break
    except Exception:
      print "it was try n:o ", trycount
      trycount = trycount + 1
  raise
  sys.exit(1)
```