Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Trnjanin

Integracija vmesnika Libvirt v okolje virtualnega laboratorija LRK VCL

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00143/2011
Datum: 02.09.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NEJC TRNJANIN**

Naslov: **INTEGRACIJA VMESNIKA LIBVIRT V OKOLJE VIRTUALNEGA LABORATORIJA LRK VCL**

INTEGRATING LIBVIRT INTERFACE INTO LRK VCL VIRTUAL LABORATORY ENVIRONMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Opišite virtualni laboratorij LRK VCL in njegovo navezanost na Vmware. Komentirajte prednosti neodvisnosti platforme VCL od tehnologije hipervizorja in poenotenja upravljanja virtualnih strojev. Opišite idejo infrastrukture in storitev oblaka in VCL umestite v ustrezno storitveno kategorijo. Podrobneje analizirajte v VCL uporabljene tehnologije in arhitekturni model. Preučite izvedljivost uporabe virtualizacijskega vmesnika Libvirt v okviru VCL in opredelite funkcionalnost, ki bi jo bilo potrebno za ta namen implementirati. Nazadnje izvedite programski modul, ki bo omogočal transparentno uporabo različnih hipervizorjev. Implementacijo testirajte v realnem okolju.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Dekan:
Z
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Trnjanin, z vpisno številko **63070339**, sem avtor diplomskega dela z naslovom:

Integracija vmesnika Libvirt v okolje virtualnega laboratorija LRK VCL

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 14. septembra 2011 Podpis avtorja:

Najlepše se zahvaljujem mentorici doc. dr. Mojci Ciglarič za mentorstvo in nasvete pri izdelavi diplomskega dela. Hvala tudi družini in prijateljem, ki so mi stali ob strani ter pretrpeli mojo odsotnost.

Kazalo

Seznam uporabljenih kratic in simbolov

VCL (angl. *Virtual Computing Laboratory*); Virtualni laboratorij.

IT Informacijske tehnologije.

OOP (angl. *Object-oriented programming*); Objektno Orientirano Programiranje.

XML (angl. *Extensible Markup Language*); Lahko berljiv strukturiran označevalni jezik.

PHP (angl. *Hypertext Preprocessor*); Skriptni programski jezik.

API (angl. *Application Programming Interface*); Programski vmesnik

DOM (angl. *Document Object Model*); Objektni model podatkov v zvezi z XML zapisom.

Perl Skriptni programski jezik.

D&C (angl. *divide and conquer*); Algoritem za reševanje problemov.

VMM (angl. *Virtual Machine Manager*); Nadzornik virtualnih storjev.

QEMU Emulator za procesor.

QCOW2 (angl. *QEMU copy-on-write*); Oblika zapisa podatkov.

VM (angl. *virtual machines*); Virtualni stroj.

HA (angl. *high availability*); Visoka razpoložljivost.

LDAP (angl. *Lightweight Directory Access Protocol*); Lahki protokol za dostop do imenikov.

IP (angl. *Internet Protocol)*; Protokol omrežnega sloja.

xCAT (angl. *Extreme Cloud Administration Toolkit*); Orodje za upravljanje.

NCSU (angl. *North Carolina State University*); Državna univerza Severne Karoline.

KVM (angl. *Kernel-based Virtual Machine*); Odprtokodni nadzornik virtualnih strojev.

SSL (angl. *Secure Sockets Layer*); Protokol za šifriranje.

OpenVZ Nadzornik virtualnih strojev na osnovi vsebnikov.

GB (angl. *gigabyte*); Milijarda bajtov.

SASL (angl. *Simple Authentication and Security Layer*); Programski sklad za preverjanje istovetnosti.

VMDK (angl. *Virtual Machine Disk*); Format diskov virtualnih strojev.

AJAX (angl. *asynchronous JavaScript and XML*); Skupina tehnologij za spletni razvoj.

SQL (angl. *Structured Query Language*); Strukturiran jezik za poizvedbe.

CoW (angl. *Copy on Write*); Strategija za optimizacijo.

zlib Knjižnica za kompresijo.

CVS (angl. *Concurrent Versioning Systems*); Sistem za kontrolo izvorne kode.

KAZALO

CSV (angl. *Comma-separated values*): Datotečni format.

PDF (angl. *Portable Document Format*): Datotečni format.

LaTeX Orodje za izdelavo dokumentov.

TLS (angl. *Transport Layer Security*); Protokol za šifriranje.

SSH (angl. *Secure Shell*); Varna lupina.

MAC (angl. *Media Access Control*); Fizični naslov omrežne naprave.

x.509 Standardni format za varnost in certifikate.

TCP (angl. *Transmission Control Protocol*); Protokol za nadzor prenosa.

HTTPS (angl. *Hypertext Transfer Protocol Secure*); Protokol za varni prenos informacij po spletu.

XEN Nadzornik virtualnih srojev.

UML (angl. *User-mode Linux*); Nadzornik virtualnih strojev.

DHCP (angl. *Dynamic Host Configuration Protocol*); Sistem za dinamično dodeljevanje naslovov.

SAN (angl. *storage area network*); Omrežje diskovnih naprav.

NAS (angl. *network attached storage*); Omrežno diskovno polje.

CI (angl. *continuous integration*)); Način testiranja za izboljšavo kvalitete.

Povzetek

Cilj diplomskega dela je integracija vmesnika Libvirt (*Libvirt API*) v okolje virtualnega laboratorija (VCL, angl. *Virtual Computing Laboratory*), kateremu omogočimo učinkovitejšo uporabo in poenoten način upravljanja različnih nadzornikov virtualnih strojev (VMM, angl. *Virtual Machine Managers*). S tem se zagotovi boljša izraba upravljalnih funkcij, ki so potrebne pri delu z virtualnim okoljem. Virtualni stroji (VM, angl. *virtual machines*) predstavljajo delovna okolja, ki so ustvarjena s tehniko virtualizacije, v katerih uporabniki opravljajo enake naloge kot v fizičnih sistemih. V nalogi je opisan razvoj programskega modula, ki prevzame dodeljevalno vlogo virtualnih strojev QEMU, z uporabo odprtokodnih tehnologij kot so: Perl, QEMU in QCOW2. Za poenoteno komunikacijo je uporabljen format XML, ki prenaša ključna sporočila iz upravljalne enote virtualnega laboratorija k izbranemu nadzorniku virtualnih strojev. Ustvarjanje klonov virtualnih strojev je realizirano z uporabo algoritma hitrega kloniranja QCOW2, ki poveča učinkovitost diskovnega sistema z zmanjševanjem zasedenosti prostora na disku. Izdelana rešitev izpolnjuje zahteve modularnosti okolja in omogoča pravilno in učinkovito delovanje rezervacij virtualnih strojev. Modul je testiran v testnem okolju virtualnega laboratorija, ki je primerljiv s produkcijskim okoljem, za katerega je tudi izdelan.

Ključne besede:

programski vmesnik Libvirt, virtualni laboratorij, virtualizacija, QEMU

Abstract

The aim of this thesis is the integration of the Libvirt API into the virtual computing laboratory (VCL), enabling more efficient use and uniform way to manage virtual machines of different virtual machine managers. This is to ensure better use of management functions that are necessary when working with virtual environments. Virtual machines present working environments that are created with the technique of virtualization, in which users perform the same functions as in physical systems. The thesis describes the development of a software module, which takes over the role of provisioning for QEMU virtual machines, using open source technologies such as Perl, QEMU and QCOW2. The XML standard is used for standardized communication and message transfer from a management unit to the selected virtual machine manager. The cloning of virtual machines is implemented via QCOW2 fast cloning algorithm, which increases disk system performance by reducing disk space occupancy. Designed solution meets the requirements of modularity of the environment and allows for a proper and efficient operation of provisioning virtual machines. The module is tested in a virtual laboratory test environment that is comparable with the production environment, for which this module was built.

Key words:

Libvirt API, virtual computing lab, virtualization, QEMU

Poglavje 1

Uvod

V današnjem svetu imajo informacijski sistemi, podprti z računalniško tehnologijo, vse večji pomen. To je razvidno iz eksponentne rasti vsakodnevnega povpraševanja po informacijah in storitvah, ki se nahajajo na različnih sistemih in v raznovrstnih okoljih. Zaradi vse večjega pritiska na ponudnike **storitev** informacijske industrije, so le-ti zaradi obstoja na trgu prisiljeni v širitev in nadgradnjo informacijskega okolja. V izogib prevelikemu naraščanju stroškov širjenja informacijske infrastrukture in vzdrževanja programske opreme se odločijo za način združevanja pomembnejših sistemov v prostorsko in cenovno ugodnejša okolja, ki zagotavljajo enako kvalitete ponujenih storitev na trgu. To združeno okolje predstavijo kot virtualno okolje, saj na enemu fizičnemu sistemu hkrati in vzporedno poteka več virtualnih sistemov.

Virtualizacija je sicer nov pojem na tem področju, koncept pa je že več desetletij star. Zaradi nenehnega razvoja strojne in programske opreme je tudi ta tehnologija postala bolj dostopna in primerna za uporabo bodisi v privatnem bodisi v industrijskem okolju. Njen glavni namen je abstrakcija in ustrezno upravljanje virov sistema. Posledica tega je učinkovitejša izraba strojne opreme za namen nudenja različnih storitev, aplikacij, operacijskih sistemov ter računalniških okolij, ki so vedno na voljo, ne glede na čas in lokacijo. Koncept računalništva v **oblaku** je pravi preporod virtualizacijske

tehnologije, ki ima bistveno vlogo pri pretvarjanju sistemov in programske opreme v obliko storitev. Zahteva za določeno storitev se izvrši kar preko spletnega vmesnika (angl. *web interface*), ki uporabniku zagotovi vse ustrezne komponente in ga posledično razbremeni dodatnega dela in tehničnega znanja, ki je drugače potrebno za upravljanje fizične strojne opreme. Tako postane programska oprema še bolj zanimiva in privlačna do končnega uporabnika in do razvijalcev programske opreme. Tem ni potrebno opraviti nakupa številne strojne opreme za izvajanje naprednih faz stresnega in obremenilnega testiranja. Namesto tega lahko brez poznavanja kompleksnih okolij zakupijo storitev želenih sistemov za določeno obdobje. Pri tem ne potrebujejo znanja o vzdrževanju sistemov in drugih posebnih izkušenj za njihovo normalno delovanje. Do najetih sistemov dostopajo preko oddaljenih povezav v katerem koli obdobju in s poljubne lokacije.

Hitra postavitev in pripravljenost rezerviranih virtualnih strojev je v okolju, kot je **izobraževalna** ustanova, ključnega pomena. Vsaka moderna ustanova, ki nudi z oblakom podprte izobraževalne storitve, lahko v razmeroma kratkem času ponudi večje število vnaprej nastavljenih virtualnih delovnih okolij. Za to poskrbi sistem, ki je v neposredni povezavi z nadzornikom virtualnih strojev. Le-ta izvaja dodatne procese nadzora, upravljanja, razporejanja in dodeljevanja računalniških virov. Največja pridobitev je enostavna dostopnost vsem uporabnikom, ki nimajo predhodnega računalniškega znanja z upravljanjem in konfiguracijo različnih sistemov. Za ponudnike teh storitev pa se posledično zmanjša potreba po površini in vzdrževanju informacijskega centra.

1.1 Opis problemskega stanja

Za uspešno in brezhibno delovanje storitvenih sistemov je potrebna pravilna izbira infrastrukture, katera je glavni nosilec virtualizacije. Pri tem je potrebno poznati želene sisteme in aplikacije, ki bodo na razpolago v obliki storitev. Nadzorniki virtualnih strojev so ključnega pomena, saj neposredno zagotavljajo okolje, ki je potrebno virtualiziranim strojem. Omejeni so na nabor delovnih ukazov, način virtualizacije in vmesnik za upravljanje. Vsak načeloma uporablja edinstven način ustvarjanja virtualnih strojev, kar za ponudnika storitev predstavlja nujni vložek v znanje, ki je potrebno za upravljanje in podporo sistemov. Poleg tega je potrebno še opozoriti na raznolikost ukazov in grafičnih vmesnikov pri uporabi različnih tehnologij virtualizacije.

Podoben problem se je prikazal po vzpostavitvi okolja virtualnega laboratorija na Fakulteti za računalništvo in informatiko v Ljubljani. Zaželena je bila uporaba konkurenčne tehnologije za podporo storitev virtualnega laboratorija. Ta je že v osnovni postavitvi temeljila na tehnologiji podjetja VMware, ki velja za enega izmed največjih ponudnikov virtualizacijskih rešitev. S pomočjo raziskave in analize trga se je postopoma oblikovala konkurenčna rešitev, ki je bila funkcionalno enakovredna in finančno bolj ugodna obstoječi tehnologiji, saj ne zahteva obveznega licenciranja nadzornega orodja in upravljalnih funkcij pri delu z virtualnim okoljem.

1.2 Pomen rešitve problema

Rešitev problema virtualnega laboratorija je v skupni uporabi knjižnice Libvirt in nadzornika virtualnih strojev KVM (angl. *Kernel-based Virtual Machine*) [14], nadgrajenega z emulacijo QEMU. Vmesnik Libvirt, katerega glavna značilnost je standardizacija ukazov za nadzor različnih virtualizacijskih tehnologij, se integrira v obstoječi sistem in **poenoti** upravljanje okolja. Ta se izvede preko skupnega medija, ki temelji na formatu XML in tako omogoči enostavno in strukturirano komunikacijo v virtualnem okolju.

Za integracijo je bila uporabljena modularna zgradba programske opreme virtualnega laboratorija, v katero smo implementirali samostojni modul Libvirt. Kot konkurenčni in cenovno najbolj ugoden ter obetaven sistem smo izbrali KVM z dodatkom emulacije tipa QEMU [10].

KVM je odprtokodna rešitev za virtualizacijo sistemov, ki ustreza vsem zahtevam virtualnega laboratorija. Za uporabo ne zahteva posebnih licenc, saj je brezplačen in javen. Ponuja preprost grafični vmesnik, ki vzpostavi povezavo z različnimi virtualizacijskimi tehnologijami kot so: VMware, XEN, VirtualBox, OpenVZ in UML. Pri tem pa podpira že uporabljeno obliko zapisa VMDK (angl. *Virtual Machine Disk*) in QCOW2. Slednja je bila uporabljena za izdelavo klonov virtualnih strojev, pri katerih je dosegla odlične rezultate v porabi diskovnega prostora. Nalogo posnemanja (angl. *emulation*) strojne opreme je izvajal emulator QEMU. Le-ta je poskrbel za učinkovito in uporabnikom neopazno širitev vseh storitev virtualnega laboratorija iz obstoječega na novo okolje KVM. Vpeljava vmesnika Libvirt je **vplivala** na:

- učinkovitejšo izrabo funkcij okolja VCL,
- standardizacijo virtualnih tehnologij,
- uporabo poenotene komunikacijo sistemov,
- manjšo porabo prostora in omogočila enakovredno gostovanje storitev na konkurenčni in finančno bolj ugodni tehnologiji.

Poglavje 2

Računalništvo v oblaku

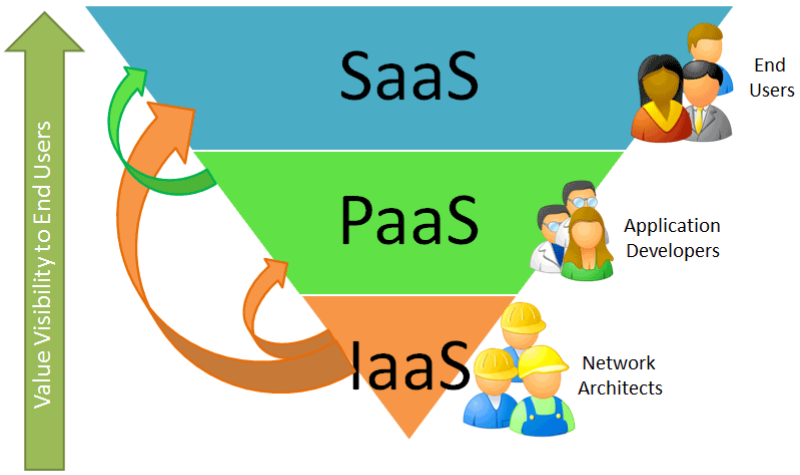
Računalništvo v oblaku (angl. *cloud computing*) predstavlja nov koncept ponujanja storitev uporabnikom preko spletnega portala (angl. *web portal*). Uporaba storitev je zelo preprosta, saj uporabniku ni potrebno postavljati lastne infrastrukture, temveč jo le-ta najame za določen čas. Ponujene storitve so lahko aplikacije, razni sistemi, okolja, podatkovne baze in strojna oprema. Pogosto pa jih omenjamo v povezavi s storitveno programsko opremo (SaaS, angl. *Software as a Service*), saj je ta trenutno najbolj razširjen koncept ponujanja **storitev**. V ponudbi storitev lahko sodeluje celotna množica ponudnikov, ki so lahko hkrati tudi končni uporabniki (Slika 2.1). Tehnološko gledano je računalništvo v oblaku le nadgradnja uslužnostnega računalništva (angl. *utility computing*) v povezavi z mrežnim računalništvom (angl. *grid computing*) [1]. Oblak (angl. *cloud*) predstavlja vse ponujene storitve, ki se nahajajo v podatkovnem centru in so na razpolago končnim uporabnikom.

Javni oblaki (angl. *public clouds*) kot so Amazon Web Services, Google AppEngine in Microsoft Azure nudijo plačljive in prosto dostopne storitve za javno uporabo. Zasebni oblaki (angl. *private clouds*) pa te storitve ponujajo znotraj podatkovnega centra podjetij.

Odprtokodni projekti VCL, Eucalyptus, OpenQRM, oVirt ter OpenNebula so primeri zasebnih oblakov. Postavitev privatnega oblaka po navadi preseže stroške najema storitev javnega oblaka, a je v nekaterih primerih nujna. Za to se odločajo podjetja, ki zahtevajo lasten razvoj storitev. Z vidika strojne opreme se v računalniškem oblaku pojavijo trije aspekti:

- iluzija neskončnih računalniških virov, ki uporabnika razbremenijo načrtovanja širitve sistema,
- postopoma povečevanje zakupljenih sistemskih virov, ki privabi podjetja z manjšimi zahtevami po informacisjkih rešitvah,
- začasni najem računalniških virov za specifične operacije kot so: restavriranje podatkov, testiranje in izvajanje izračunljivosti na množici zakupljenih procesorjev.

Ponujene storitve omogočajo številne nove priložnosti za obstoječe in bodoče programske rešitve. Aplikacijam, ki za svoje delovanje potrebujejo večjo količino podatkov, se ponudi možnost uporabe večjih podatkovnih centrov, vzporednega izvajanja sistemov, visoke zanesljivosti, paketne obdelave in poslovne analize. Ključna ideja računalništva v oblaku je redefinicija in **osvežitev** trga storitev, ki je na voljo vsem uporabnikom v relativno ugodnih cenah.



Slika 2.1: Vertikalni model ponudbe storitev oblaka.

Ključne značilnosti računalniškega oblaka so:

- učinkovitejša izraba računalniških virov,
- dinamično dodeljevanje in rezervacija okolij (angl. *provisioning*),
- upravljanje storitev v oblaku z uporabo vmesnika (API, angl. *application programming interface*),
- zanesljivost z uporabo redundance (angl. *redundancy*) in visoke razpoložljivosti sistemov (HA, angl. *high availability*),
- overjen dostop pri delu s storitvami (angl. *authentication*),
- razširljivost strežniških sistemov glede na potrebe uporabnikov (angl. *scalability*),
- uporaba istih storitev za potrebe različnih strank (angl. *multitenancy*).

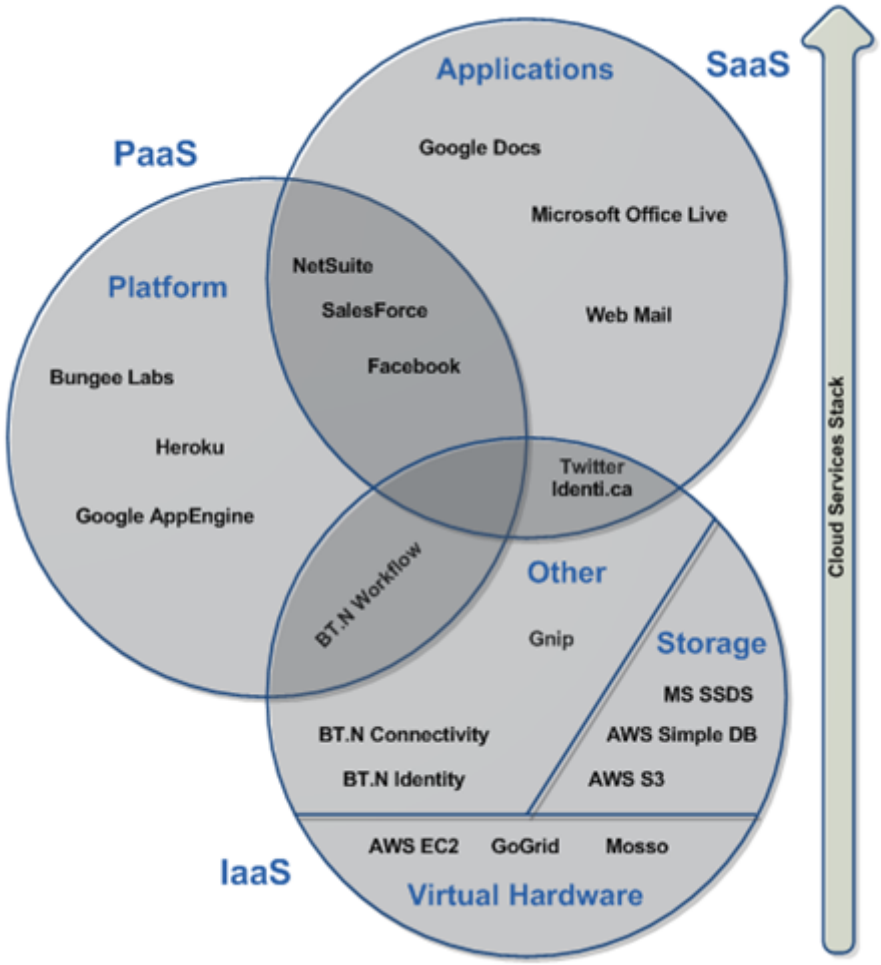
2.1 Storitveni model

Storitveni modeli predstavljajo različne nivoje storitev, ki so ponujene s strani računalniškega oblaka. Razlikujejo se glede na področje uporabe, kot so:

- **Infrastruktura** kot storitev (angl. *Infrastructure as a Service*) - predstavlja glavni strošek vseh strežnikov, omrežnih naprav ter naprav namenjenim za hrambo in varnostno kopiranje podatkov. Primerna je za začetna podjetja, ki še niso pripravljena investirati v vrhunsko strojno opremo. Primer storitve je Amazon Web Services (Slika 2.2), ki ponuja vrsto različnih sistemov na oddaljenih lokacijah, za katere ni potrebna namestitev lastne infrastrukture.

- **Programski sklad** kot storitev (angl. *Platform as a Service*) - omogoča enostaven razvoj in postavitev aplikacij v oblaku. V okviru storitev ponuja operacijske sisteme s potrebno vmesno programsko opremo (angl. *middleware*). Samodejno izvaja preglede, optimizacijo, posodobitve sistema in poskrbi za izdelavo varnostnih kopij. Primer gostiteljske platforme je Google App Engine (Slika 2.2), ki podpira Java in Python programska jezika, s katerima izvajamo razne ukaze vmesnika. Programska sklada Heroku in Engine Yard ponujata poleg programskega jezika Ruby on Rails še podatkovno bazo MySQL. Podjetje eyeOS prav tako ponuja odprtokodni namizni sistem, ki je namenjen za komunikacijo med uporabniki in delu z dokumenti.

- **Programska oprema** kot storitev (angl. *Software as a Service*) – ne potrebuje nobenega dodatnega nameščanja ali poganjanja komponent v lokalnem okolju. Velja za najbolj ponujeno storitev do katere dostopamo na preprost in učinkovit način, in sicer z uporabo spletnega brskalnika (angl. *web browser*). Primer storitev je Google Docs (Slika 2.2), ki nam omogoča oddaljeno upravljanje poljubnih dokumentov, podobno kot v programskem paketu Microsoft Office 365.



Slika 2.2: Storitveni sklad oz. model računalniški oblakov v realnem času.

2.2 Okolje VCL

Virtualni laboratorij je večkratno nagrajena **odprtokodna** programska oprema, ki omogoča vrsto rešitev za varno in učinkovito delovanje z virtualiziranimi in podatkovnimi sistemi [5]. Njeni glavni značilnosti, storitvena usmerjenost in delovanje na zahtevo končnega uporabnika, jo uvrstita v sodobne računalniške oblake.

Namen virtualnega laboratorija je ponujanje virtualnih sredstev v obliki sistemov za potrebe **izobraževalnih** ustanov. Prva implementacija sistema virtualnega laboratorija je bila izvedena na Državni univerzi Severne Karoline (NCSU), kot odgovor na vse večje povpraševanje po izobraževalni programski opremi [2]. Virtualni laboratorij je osnovan kot varen, razširljiv, obvladljiv in storitveno usmerjen sistem z raznovrstno ponudbo virtualnih okolij.

Zahtevek za storitev rezervacije virtualnega okolja se izvrši preko spletnega vmesnika, nato se določi rezervacija virtualnega stroja v poljubni virtualni infrastrukturi, ki na podlagi dobljenih podatkov pripravi novo okolje. Postopek je zelo preprost, varen in **učinkovit**. Nad uporabnikom, ki je zahteval storitev, se pred vpisom v okolje VCL izvede preverjane pristnosti z uporabo protokola LDAP (angl. *Lightweight Directory Access Protocol*). Zanimiva je tudi implementacija požarnega zidu (angl. *firewall*), ki se v času rezervacije okolja nastavi tako, da omogoči dostop le uporabniku rezervacije.

Arhitektura

Virtualni laboratorij je podprt s tehnologijo storitveno usmerjene arhitekture [4], ki omogoča komunikacijo med okolji kot so: fizične delovne postaje, raznovrstni strežniki (spletni strežniki, podatkovne baze) in virtualni stroji (Slika 2.3).

Slike oz. okolja (angl. *images*)

Slika je zbirka programske opreme, ki je nameščena na operacijskem sistemu. Njen namen je distribucija programskega paketa oz. storitve z možnostjo posodabljanja in nadgrajevanja. Potrebno jih je hraniti na diskovnih sistemih (angl. *storage systems*), do katerih lahko dostopata upravljalna enota in gostitelj virtualnega okolja (angl. *virtualization host*). Prav tako je možno uporabiti več različnih podvrst iste slike (angl. *subimage*). Slike se distribuira na fizična in virtualna okolja. Pri namestitvi na fizično okolje se uporabi orodje xCAT (angl. *Extreme Cloud Administration Toolkit*). Za virtualno okolje pa je potrebno uporabiti virtualizacijo podjetja VMware.

Virtualni stroji (angl. *nodes*)

Slike se namestijo na virtualizirane stroje, kateri morajo biti registrirani in pravilno nastavljeni v podatkovni bazi (angl. *database*) virtualnega laboratorija. Virtualni stroji, ki imajo naloženo zahtevano sliko, tvorijo rezervirano storitev oz. okolje.

Nadzorne enote (angl. *management nodes*)

Nadzorne enote upravljajo procese, ki se izvajajo na strani strežnika. Naloga zalednih (angl. *backend*) procesov je poskrbeti za pravilno namestitev slik na virtualne stroje z uporabo podatkovne baze MySQL. Ta se lahko nahaja na poljubni lokaciji.

Urniki (angl. *schedules*)

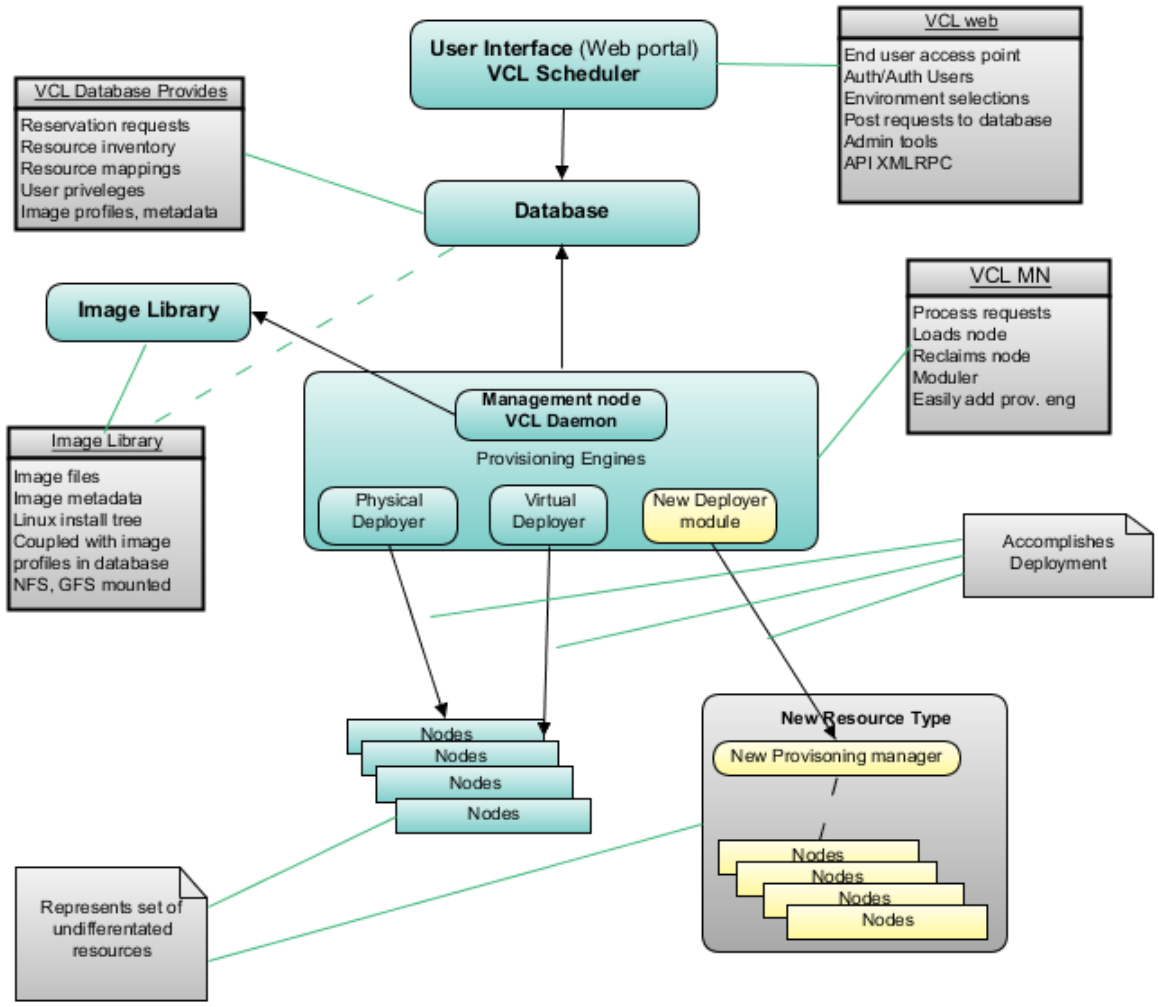
Vsak registriran virtualni stroj pripada določenemu urniku, s katerim si uporabnik pomaga pri izbiri termina za rezervacijo slike. Prav tako lahko spremlja trajanje aktivnih rezervacij za določeno obdobje.

Spletni vmesnik

Vlogo čelnega sistema (angl. *frontend*) ima spletni vmesnik, ki skrbi za interakcijo z uporabniki. Narejen je v tehnologijah PHP in AJAX (angl. *asynchronous JavaScript and XML*). Ponuja vrsto uporabnih funkcij kot so: preverjanje uporabnikov, rezervacije slik, analiza preteklih rezervacij, upravljanje z uporabniškimi skupinami, dodeljevanje pravic uporabnikom in konfiguracija upravljalne enote.

Podatkovna baza

Podatkovna baza MySQL hrani podatke, ki so potrebni za delovanje okolja. Pri tem beleži zahtevke za rezervacije, inventar opreme, nastavitve slik, podatke o uporabniških profilih in pravicah dostopa.



Slika 2.3: Arhitekturni model virtualnega laboratorija [6].

2.2.1 Virtualizacija

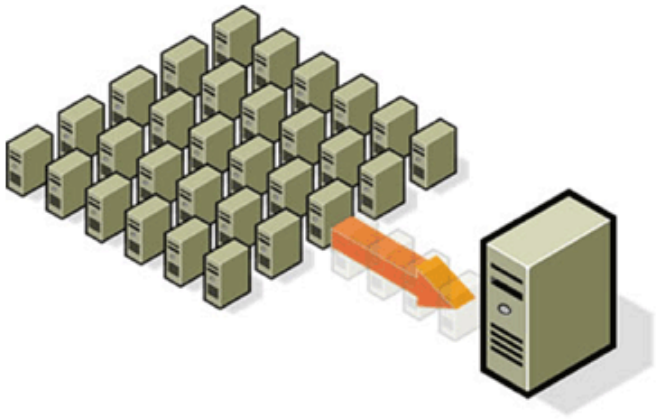
Definicija in namen

Na splošno je virtualizacija proces združevanja in abstrakcije sistemskih virov, ki jo izvedemo z namenom prikrivanja fizičnih lastnosti določene naprave, računalniških virov ter njihovih omejitev. Omogoča navidezno razdelitev strojne opreme med več različnih operacijskih sistemov. Najprej se izvede **združitev** (Slika 2.4) večjega števila fizičnih naprav v sistem, ki mu pravimo gostitelj. Na njem se potem izvede virtualizacija vseh združenih sistemov. Tako lahko naprava, ki je sposobna virtualizacije, hkrati izvaja več med seboj ločenih in neodvisnih virtualnih strojev, ki predstavljajo virtualizirane operacijske sisteme. Za pravilno razporeditev sistemskih virov med virtualnimi stroji poskrbi nadzornik virtualnih strojev.

Virtualizacija načeloma odpira tri glavna področja izboljšav:

1. znižanje celotnih stroškov za okolje informacijske tehnologije (IT, angl. *information technology*),
2. povečanje elastičnosti informacijske infrastrukture,
3. oblikovanje kakovostnih IT storitev z obvladljivo razpoložljivostjo.

Virtualizacija poenostavi **nadzor** in namestitev sistemov z uporabo enotnega orodja, ki omogoča vzporedno nameščanje in kloniranje različnih sistemov na različne lokacije. Prav tako poveča izkoriščenost strojne opreme, zniža operativne stroške in zmanjša porabo energije. Pomembna lastnost je tudi takojšnja ponovljivost in obnovitev sistema v primeru katastrofe in nerazpoložljivosti informacijske infrastrukture (angl. *disaster recovery*).



Slika 2.4: Združevanje računalniških sistemov [7].

Vrste virtualizacije

Vrsta virtualizacije se določi glede na način posnemanja strojne opreme [16]. **Popolna** virtualizacija (angl. *full virtualization*) je celotno posnemanje delovanja sistemske opreme na programski ravni. Velja za nastarejši način virtualizacije, ki z abstrakcijo strojne opreme ustvari popolno virtualno okolje. Znotraj tega okolja se izvaja gostujoč operacijski sistem, natanko tako, kot bi se izvajal na fizični sistemski opremi. Za popolno virtualizirane sisteme je značilno, da so neodvisni od strojne opreme in jih lahko brez težav selimo med gostitelji iste arhitekture. Finančno in razvojno bolj ugoden način virtualizacije je **paravirtualizacija** (angl. *paravirtualization*). Ta način zahteva spremenjen operacijski sistem na katerem potekajo glavni procesi virtualizacije. Posnemana strojna oprema je podobna fizični opremi. Vsi procesi, ki bi se sicer težje izvajali v popolnem virtualnem okolju, so preneseni na gostujoči operacijski sistem. To omogoči hitrejše delovanje gostujočih sistemov in iz-

boljšano izvajanje določenih procesov. Virtualizirani stroji tako postanejo odvisni od sistemske opreme fizičnega računalnika.

Najnovejši način virtualizacije je **strojno podprta** virtualizacija (angl. *hardware-assisted virtualization*), ki predstavlja največji dosežek v okolju virtualnih tehnologij. Deluje tako, da izključi potrebo po binarnem prevajanju ukazov in tako pohitri delovanje nespremenjenega operacijskega sistema. Ustvarjena je bila z dodajanjem procesorskih ukazov na strojnem nivoju.

Nadzorniki virtualnih strojev

Nadzornik virtualnih strojev se lahko izvaja na različnih nivojih. Pri popolni virtualizaciji se izvaja kot programski proces znotraj operacijskega sistema, medtem ko se le-ta pri paravirtualizaciji izvaja neposredno na strojni opremi. Najbolj uporabljeni nadzorniki virtualnih strojev so VMware ESX, Microsoft HyperV ter odprtokodna XEN in KVM. Vsi so med seboj **primerljivi**, vendar ima podjetje VMware še vedno občutno prednost, saj se je na trgu pojavilo kot prvi ponudnik komercializiranih virtualizacijskih storitev.
Poleg tega ima najbolj dodelan uporabniški vmesnik za zahtevnejše upravljanje z virtualnim okoljem. Zelo obetaven je tudi KVM, saj je skupaj z emulatorjem QEMU zmožen opravljanja vseh potrebnih zahtev virtualizacije. Upravljanje izvajamo preko ukazne vrstice. Delovanje virtualizacije je torej odvisno od nadzornika virtualnih strojev katerega **glavne naloge** so:

- simulacija delovanja virtualne strojne opreme,
- abstrakcija fizične strojne opreme,
- pravilno razporejanje sistemskih virov med virtualne stroje,
- upravljanje virtualnih strojev.

Vloga virtualizacije v okolju VCL

Virtualizacija ima veliko vlogo v arhitekturi virtualnega laboratorija, saj omogoča abstrakcijo računalniških virov za potrebe virtualnih strojev. Rezultat tega je skupek virtuliziranega operacijskega sistema s pripadajočo programsko opremo, ki predstavlja **sliko**. Na strani računalniške infrastrukture se uporabi fizični strežnik, ki ustvari virtualni stroj, na ketermu se omogoči oddaljen dostop do rezerviranega okolja. To okolje je dostopno ne glede na napravo, ki jo uporabljamo. Tako lahko s tehniko virtualizacije omogočimo ponujanje programske opreme v obliki **storitev**. Okolje virtualnega laboratorija je v osnovi pripravljeno na virtualizacijo VMware, z možnostjo razširitve na ostale tehnologije, kot so XEN, KVM in VirtualBox.

2.2.2 Funkcionalnost

Spletni vmesnik okolja VCL ponuja večjo izbiro funkcij, ki zadoščajo navadnim in bolj naprednim uporabnikom storitev. Med **osnovne funkcije** uvrščamo rezervacije slik, pregled prostega termina za rezervacijo ter funkcijo za izhod iz spletnega portala.

Napredne funkcije portala VCL

Ustvarjanje bločne rezervacije okolja (angl. *block allocation*)

Funkcija omogoča večkratno izvedbo večjega števila rezervacij za določeno obdobje (Slika 2.5). Uporablja se lahko za namene virtualne učilnice. S predhodno pripravo ustreznih sistemov se zmanjša čakalna doba oddanih zahtev za novo okolje. Bločna rezervacija se lahko uporabi za točno določeno skupino uporabnikov, ki so ob prijavi obravnavani z večjo prioriteto. Po izteku rezervacije sledi izklop in izbris sistemov.

Upravljanje z uporabniki in skupinami

Funkcija omogoča upravljanje skupin uporabnikov (angl. *user groups*) in sistemskih virov (angl. *resource groups*). Omogoča dodajanje lokalnih in obstoječih uporabnikov, ki obstajajo na drugih sistemih. To

se izvede s protokolom LDAP, ki poveže okolje VCL z obstoječimi imeniškimi strukturami, kot je aktivni imenik (AD, angl. *Active Directory*). V skupino sistemskih virov lahko dodajamo fizične in virtualne sisteme, okolja, ki so namenjena rezervacijam, in urnike. Ko dodeljujemo virtualne stroje skupinam sistemov, uporabimo metodo preslikovanja skupin (angl. *image mapping*) in tako poenostavimo problematiko dodeljevanja večjega števila okolij (Slika 2.6).

Izdelava lastnih urnikov

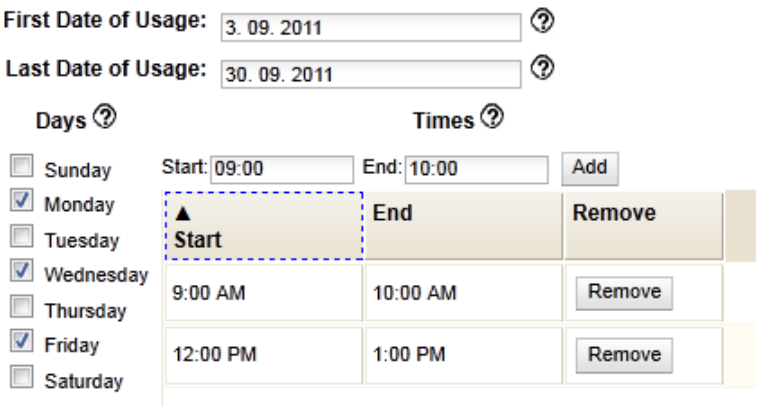
Funkcija omogoča bolj učinkovito izrabo okolij in sistemskih virov, saj jih uporabi le za določeno obdobje. Tako izkoristimo uporabo prostih in neobremenjenih sistemov in jim dodelimo druga opravila.

Upravljanje slik (angl. *image management*)

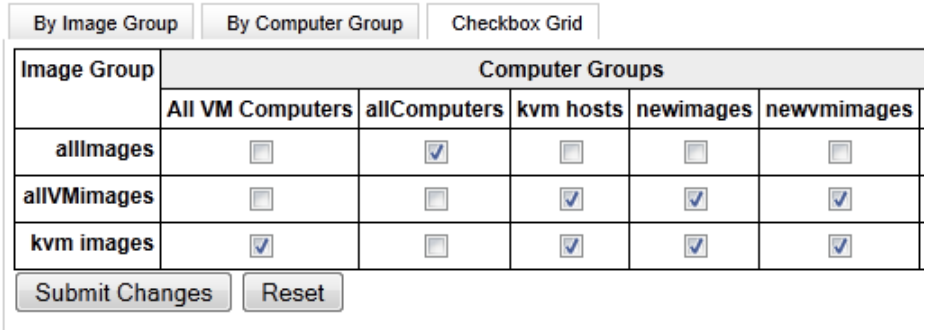
Funkcija ustvarjanja in posodabljanja slik je zelo podobna funkciji za izdelavo rezervacije okolja. Uporabnik ima možnost izbiranja različnih vzorcev okolja, ki lahko vsebujejo določene aplikacije. Prav tako je možno urejati imena slik, lastnike okolij, strojne zahteve in nastavitve varnostne politike. Zaradi preprostejše uporabe je potrebno slike postaviti v določene skupine in jih nato povezati z računalniškimi viri (angl. *resource mapping*). Podobno urejamo tudi fizična in virtualna okolja, ki jih razporedimo v okolja gostiteljev. Te je potrebno nastaviti v virtualizacijskih profilih (angl. *VM host profiles*), v katere vnesemo pot ali lokacijo osnovnih okolij in njihove strojne zmožnosti, kot so: frekvenca procesorja, velikost pomnilnika, pasovna širina omrežja in tip sistema.

Upravljanje s pravicami

Administrator lahko dodeli razne pravice uporabnikom in uporabniškim skupinam. Novim uporabnikom se pravice, ki so potrebne za upravljanje osnovnih funkcij, samodejno dodelijo. Uporabniku se lahko odobri določen nabor računalniških virov, uporaba urnikov, upravljanje s skupinami, ustvarjanje novih okolij in uporaba rezervacij slik.



Slika 2.5: Primer bločne rezervacije v dveh intervalih na dan.



Slika 2.6: Primer preslikovanja skupin slik in računalniških virov.

2.2.3 Modularna arhitektura

Najbolj pomembni del virtualnega laboratorija je zagotovo modularna zgradba arhitekture na strani strežnika, ki loči določene funkcije od jedra okolja VCL. Ta vrsta zgradbe omogoča možnost **razširitve** in lažje integracije ostalih zunanjih tehnologij, kot so sistemi za rezervacije oz. dodeljevanje virov (angl. *provisioning engines*), operacijski sistemi in nadzorna orodja. Integracija poteka v okviru lastnega razvoja. Način integracije pa je odvisen od okolja s katerim virtualni laboratorij operira. Slika 2.7 prikazuje razredni diagram zalednega dela okolja VCL.

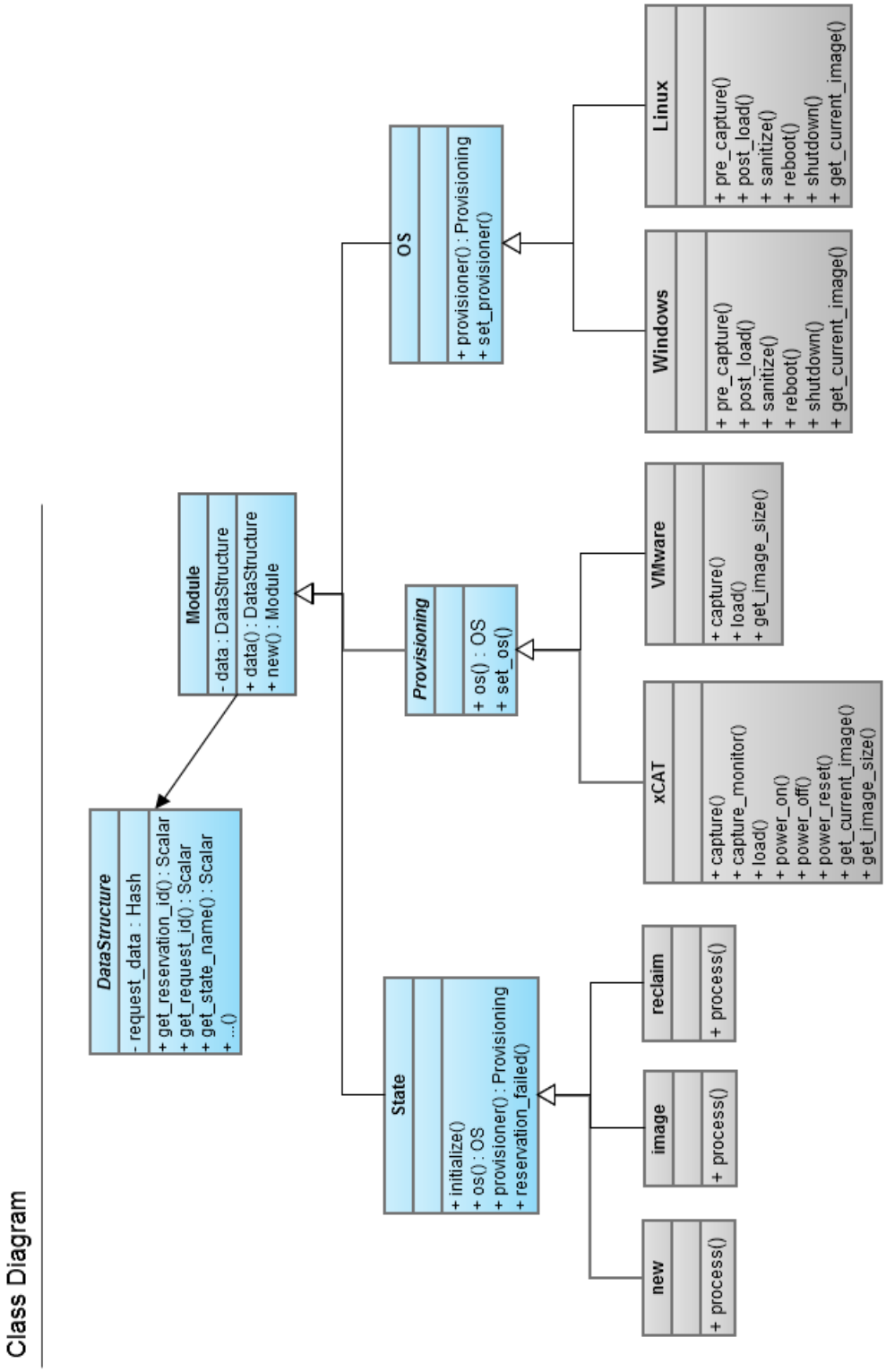
Objektna usmerjenost in dedovanje

Proceduralno programiranje pogostokrat zahteva veliko discipline in upoštevanje navodil "lepega" programiranja. Tako se lahko kompleksnost kode poveča skladno s kompleksnostjo projekta, kar privede do neurejene programske opreme.

Objektna usmerjenost (OPP, angl. *object-oriented programming*) omogoča enostavno abstrakcijo problema, enkapsulacijo, polimorfizem in dedovanje. Z uporabe tehnike dedovanja (angl., *inheritance*) omogoča dostop do funkcij višjih podedovanih razredov in zmanjša podvajanja programske kode.

Prednosti uporabe:

- razvijalcem je olajšano delo pri implementaciji novih tehnologij,
- lažje razumevanje problema - koncept "deli in vladaj" (D&C, angl. *divide and conquer*),
- jedra okolja virtualnega laboratorija ni potrebno spreminjati, za to uporabimo module,
- okolje je bolj prilagodljivo na različne nastavitve,
- lažje vzdrževanje programske kode.



Slika 2.7: Modularna zgradba zalednega dela VCL.

Poglavje 3

Tehnologije in orodja

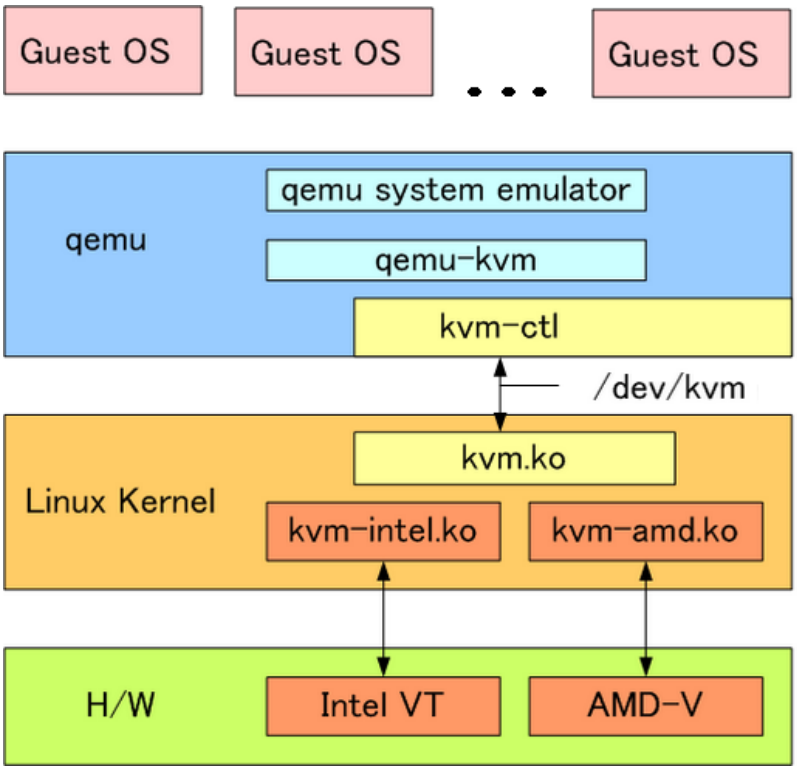
Za razvoj rešitve smo izbrali širše uporabljena **odprtokodna** orodja in tehnologije na področju virtualizacije, zapisovanja podatkov, programskih jezikov in vmesnikov. Vsa omenjena področja odražajo preprostost, enostavno uporabo in učinkovitost.

3.1 Uporabljene tehnologije

3.1.1 Nadzornik virtualnih strojev KVM

Sistem KVM (angl. *Kernel-based Virtual Machine*) uvrščamo med nadzornike virtualnih strojev, ki ponujajo polno virtualizacijo na nivoju operacijskega sistema in strojne opreme. Ustvarjen je v obliki modula za jedro operacijskega sistema Linux, na katerega ne vpliva, saj deluje v ločenem okolju. Za komunikacijo s procesorji podjetja Intel in AMD uporablja posebna modula **kvm-intel.ko** ter **kvm-amd.ko**, ki omogočita pohitritev delovanja gostujočih virtualnih strojev (Slika 3.1). Večina ukazov se izvede neposredno na nivoju centralno procesne enote (CPU, angl.*central processing unit*). Popolno abstrakcijo strojno opreme pa doseže s pomočjo emulatorja QEMU, ki skrbi za posnemanje in vzporedno izvajanje strojne opreme.

QEMU je generična in odprtokodna programska rešitev, ki omogoča **emulacijo** strojne opreme in virtualizacijo operacijskih sistemov. Z uporabo tehnike binarnega dinamičnega prevajanja (angl. *Dynamic binary translation*) je zmožen poganjati iste sisteme v okoljih različnih arhitektur. V povezavi z nadzornikoma virtualnih strojev KVM in XEN izvaja ukaze neposredno v procesorju gostitelja. Omogoča virtualizacijo x86, PowerPC in S390 sistemov. Prav tako ga lahko uporabimo kot samostojnega nadzornika virtualnih strojev. Njegove značilnosti so izkoriščene v različnih tehnologijah virtualizacije KVM, XEN in Virtualbox, saj omogoča istočasno in vzporedno izvajanje virtualnih procesorjev.



Slika 3.1: Arhitektura virtualizacije KVM z emulatorjem QEMU.

3.1.2 Oblika zapisa QCOW2

Zapis QCOW2 velja za enega izmed množice podprtih formatov emulatorja QEMU. V osnovi se predstavi kot datoteka z fiksnim bločnim zapisom, ki uporablja **optimiziran** algoritem za dodeljevanje podatkovnih blokov. Algoritem CoW (angl. *Copy on Write*) velja za optimizirano strategijo v računalništvu, ki v končnem pogledu zmanjša zasedenost diskovnih sistemov. Deluje na osnovi zapisovanja podatkovnih razlik v ločeno strukturo.

QCOW2 podpira večkratno shranjevanje stanja virtualnih strojev v podatkovih gručah. V primeru, ko se pojavi zahteva po spremembi pomnilniške lokacije, katera je v lasti različnih procesov, jedro sistema prestreže klic za prepis lokacije in jo kopira v posebno strukturo tistega procesa, ki je poslal zahtevo po spremembi lokacije (angl. *write*). Postopek se izvede transparentno in je neopazen ostalim procesom.

Pridobitve zapisa QCOW2:

- manjša zasedenost diskovnega prostora,
- optimizacija,
- večkratna možnost zajemanja stanja slik (angl. *snapshotting*),
- podpora za kompresijo zlib,
- podpora šifriranju AES (angl. *Advanced Encryption Standard*).

3.1.3 Programski jezik Perl

Za razvoj rešitve je bilo potrebno znanje programskega jezika Perl [12], v katerem je tudi napisana vsa modularna zgradba virtualnega laboratorija. Perl velja za visoko-nivojski programski jezik namenjen izdelavi skript. Povzame veliko funkcionalnosti iz ostalih programskih jezikov ko so C, AWK, sed in shell script. Uporablja se za razvoj mrežnih aplikacij, pri sistemski administraciji, v bioinformatiki ter pri razčlenjevanju teksta (angl. *text parsing*).

3.2 Uporabljena orodja

3.2.1 Programski vmesnik Libvirt

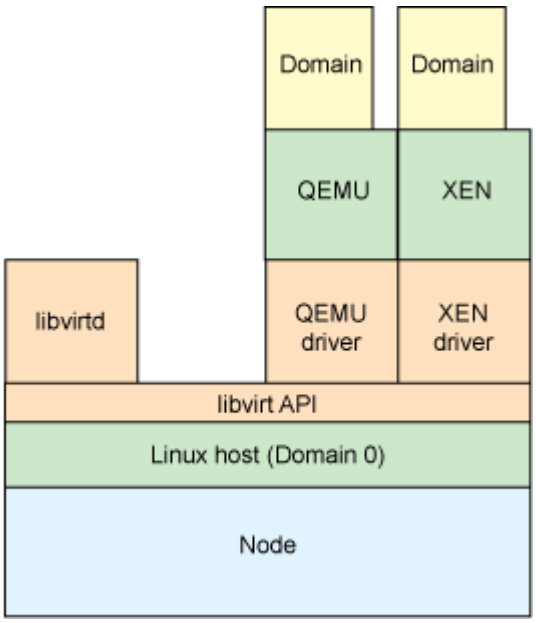
Libvirt [9] je trenutno najbolj obetajoč programski vmesnik na področju upravljanja nadzornikov virtualnih strojev, ki so narejeni na osnovi operacijskega sistema Linux. Za njegovo uporabo ni potrebno zakupiti licenc. Napisan je v programskem jeziku C in tako velja za dolgoročno stabilen programski vmesnik (angl. *Stable C API*). Glavna ideja vmesnika je oddaljeno **upravljanje** virtualnih okolji s katero lahko dodeljujemo, rezerviramo, ustvarjamo, nadziramo in selimo virtualne stroje. Lahko se poveže tudi z drugimi programskimi jeziki kot sta Perl in Pyton. Primer povezave med vmesnikom Libvirt in programskim jezikom Perl je modul **Sys-Virt** [11], ki uporablja vse potrebne funkcije za upravljanje virtualnih okolij. Vmesnik omogoča delo z virtualnimi sistemi KVM, QUEMU, LXC, OpenVZ, UML, VirtualBox in VMware (Slika 3.2).

Storitve, ki jih **ponuja** vmesnik Libvirt:

- uporaba šifriranja TLS (angl. *Transport Layer Security*) in certifikatov x.509 za oddaljeno upravljanje,
- overjanje oddaljenega upravljanja s protokoloma Kerberos in SASL (angl. *Simple Authentication and Security Layer*),
- upravljanje s pravicami lokalnega dostopa z uporabo sistema PolicyKit,
- samodejna uporaba odkrivanja Avahi multicast-DNS,
- upravljanje virtualnih strojev, omrežij in sistemov za shranjevanje podatkov,
- uporaba agenta za okolja operacijskih sistemov Linux, Solaris in Windows.

Storitve, ki jih vmesnik Libvirt **ne ponuja**:

- pretvarjanje formatov virtualnih diskov,
- upravljanje nadzornika virtualnih strojev Microsoft HyperV,
- upravljanje gruč gostiteljev,
- tehnologije visoke zanesljivosti.



Slika 3.2: Uporaba vmesnika libvirt za virtualizacije QEMU in XEN [8].

3.2.2 Razvojno okolje Eclipse - EPIC

Eclipse je odprtokodno programsko okolje in napredni urejevalnik besedila, ki ponuja pregleden in preprost uporabniški vmesnik z možnostjo prilagajanja za vsakega uporabnika. Velja za široko uporabljeno orodje z možnostjo razširitev. V osnovi je bil ustvarjen v okviru projekta JDT (angl. *Java development tools*) in kasneje razširjen za podporo programskih jezikov C, PHP, Python, Ruby, C++ in Perl. EPIC (angl. *Eclipse Perl Integration*) je vtičnik okolja Eclipse, ki nudi podporo programskemu jeziku Perl [15]. Njegove značilnosti so obarvanje kode, pregled nad napisanimi metodami, uporaba pregledovalnika napak in integracija s sistemi za kontrolo izvorne kode (CVS, angl. *Concurrent Versioning Systems*).

3.2.3 Podatkovna baza MySQL

Virtualni laboratorij uporablja podatke, ki so shranjeni v relacijskih tabelah baze MySQL. Trenutno jo razvija in vzdržuje podjetje Oracle, ki je prav tako ponudnik storitev na področju podatkovnih baz. MySQL omogoča standardiziran dostop do podatkov preko poizvedb SQL (angl. *Structured Query Language*). Zaradi dodanega nivoja abstrakcije vmesnika je omogočena prenosljivost napisanih poizvedb med različnimi sistemi.

3.2.4 Spletni strežnik Apache

Spletni strežnik Apache omogoča javni dostop (zunanje omrežje) do storitev okolja virtualnega laboratorija. Prednost strežnika Apache je v prenosljivosti in dostopnosti aplikacij. Uporabniki lahko do sistema dostopajo iz poljubnih lokacij pod pogojem, da imajo omogočeno internetno povezavo.

3.2.5 PhpMyAdmin

PhyMyAdmin je odprtokodna programska rešitev, ki se uporablja za upravljanje podatkovne baze MySQL. Vsebuje bogat grafični vmesnik, zgrajen na tehnologijah PHP in JavaScript, ki omogočita strukturiran prikaz podatkov v spletnem vmesniku.

Orodje PhpMyAdmin ponuja:

- uporabniku prijazen in oblikovan izpis podatkov,
- neposredno izvajanje poizvedb SQL,
- ažuriranje, brisanjem in ustvarjanje tabel,
- vzdrževanje strežnika podatkovne baze MySQL,
- izvoz podatkov v različne formate kot so: CSV (angl. *Comma-separated values*), SQL, XML, PDF (angl. *Portable Document Format*) ter LaTeX.

Poglavje 4

Uporaba vmesnika Libvirt

Poglavje je namenjeno predstavitvi uporabe vmesnika Libvirt pri upravljanju virtualnega okolja. Predstavljeni primeri uporabe so izvedeni v ukazni lupini (angl. *shell*) operacijskega sistema Centos 6.0. Torej gre za Linux okolje na osnovi distribucije Red Hat. Ukaze vmesnika Libvirt izvajamo z uporabo programa **virsh**, ki omogoča: ustvarjanje, začasno ustavljanje in izklapljanje virtualnih strojev. Prav tako nam ponudi vrsto funkcij za upravljanje virtualnih omrežij (angl. *virtual networks*) in diskovnih sistemov.

4.1 Connection URI

Connection URI (angl. *Uniform Resource Identifier*) predstavlja unikatni niz podatkov, ki je potreben za določitev ustreznega gonilnika (angl. *connection driver*), s katerim se izvede povezava do nadzornika virtualnih strojev (Slika 4.1). Obvezne **komponente** povezave:

- tip povezave glede na virtualno tehnologijo (XEN, QEMU, ESX),
- protokol povezave kot so: TCP (angl. *Transmission Control Protocol*), TLS in SSH,
- naslov ali ime gostitelja,
- nivo dostopa do gostitelja (session, system).

```
type://[username@]hostname[:port]/[datacenter[/cluster]/server]
```

Slika 4.1: Struktura privzete povezave *Connection URI*.

VMware ESX URI

Povezava z nadzornikom virtualnih strojev ESX se izvrši preko storitvenega vmesnika VMware vSphere API, ki je nameščen na strani gostitelja. Komunikacija poteka preko varnega protokola HTTPS (angl. *Hypertext Transfer Protocol Secure*), ki zahteva dodatno konfiguracijo certifikatov SSL (angl. *Secure Sockets Layer*). Slika 4.2 prikazuje primer povezave do gostitelja *example-esx.com*, ki je del gruče *cluster1* v podatkovnem centru *dc1*, katerega nadzoruje strežnik vCenter *example-vcenter.com*.

```
vpx://example-vcenter.com/dc1/cluster1/example-esx.com
```

Slika 4.2: Povezava *Connection URI* do gostitelja ESX.

4.2 Upravljanje navideznega stroja

Za upravljanje virtualnih strojev je trenutno na voljo približno 50 ukazov. Najbolj pogosto se uporabljajo ukazi za ustvarjanje, ponovni zagon in izklop virtualnega stroja. Eden bolj pomembnih pa je ukaz za spreminjanje

nastavitev (angl. *edit*) virtualnega stroja. Ob izvedbi ukaza se izpiše trenutna konfiguracija stroja v obliki XML, katero lahko urejamo, shranimo in izvršimo z nadzornikom virtualnih sredstev.

Izpis virtualnih strojev

Izvedba ukaza v osnovi ne zahteva dodatnih argumentov, saj izpiše le aktivne virtualne stroje. Za izpis neaktivnih strojev pa je potrebno podati argument "–inactive". V primeru izpisa vseh virtualnih strojev, ne glede na njihovo operativno stanje, pa uporabimo argument "–all".

Primer izpisa vseh virtualnih strojev z ukazno vrstico *virsh* (Libvirt).

```
1 [root@ljkvm01 datastore]# virsh list --all
2  Id Name                 State
3 ----------------------------------
4   1 vcl                  running
5   3 slot5-winxp01212-v0-Fri-Sep--2-13:17:45-CEST-2011 running
6   7 slot6-winxp01212-v0-Sun-Sep--4-05:26:21-CEST-2011 running
```

4.3 Pregled diskovnega okolja

Najprej je potrebna uspešna povezava do gostitelja virtualnih strojev, nato se uporabi ukaz "pool-list", ki v osnovi prikaže le aktivno uporabljene lokacije na diskovnem okolju. Če želimo izpisati podrobnejše informacije kot so: kapaciteta, zasedenost ter količina prostora, ki je na voljo, uporabimo ukaz *pool-info*. Trenutno je na voljo približno 40 ukazov za manipulacijo z diskovnim okoljem.

Primer izpisa informacij lokacije za shranjevanje klonov slik.

```
[root@ljkvm01 datastore]# virsh pool-info clones
Name:           clones
UUID:           4e17d0fe-bdd5-11e0-82b0-001a4b53923a
State:          running
Persistent:     yes
Autostart:      yes
Capacity:       192.25 GB
Allocation:     14.01 GB
Available:      178.24 GB
```

4.4 Upravljanje omrežij

Za upravljanje omrežij lahko uporabimo približno 15 ukazov. Način izvedbe ukazov je zelo podoben kot pri delu z ostalimi komponentami vmesnika Libvirt (virtualni stroji, diskovna okolja).

Najpomembnejši ukazi so:

- izpiši aktivna omrežja,
- informacija o izbranem omrežju,
- spreminjanje nastavitev omrežja,
- registriraj in usvari omrežje,
- odstrani izbrano omrežje

Primer izpisa registriranih omrežij.

```
[root@ljkvm01 datastore]# virsh net-list --all
Name                 State      Autostart
-----------------------------------------
private              active     no
public               active     no
```

Poglavje 5

Razvoj modula Libvirt

Za uspešno izdelavo programske rešitve je bilo potrebno uporabiti optimalen, predvidljiv in ponovljiv postopek razvoja, s katerim bi dosegli večjo produktivnost in **kvaliteto** programske opreme. Slabo poznavanje problemske domene in kompleksnost rešitve je bilo potrebno minimalizirati s pomočjo metodologije ekstremnega programiranja (XP, angl. *Extreme Programming*), ki je vključevala naslednje **faze**:

1. študija izvedljivosti rešitve,
2. zajem zahtev okolja VCL,
3. načrtovanje programske rešitve,
4. kodiranje rešitve,
5. implementacija rešitve,
6. testiranje v testnem okolju,
7. prenos v končno produkcijsko okolje.

Zaradi vzporednega poteka zajema zahtev, načrtovanja in testiranja rešitve je bila izbira metodologije ekstremnega programiranja najbolj primerna, saj prav tako priporoča sprotne priprave in implementacijo rešitve v testno okolje.

5.1 Analiza in načrtovanje

Fazi analize in načrtovanja sta del razvojnega cikla vseh kompleksnejših informacijskih sistemov. Z uporabo analize je možno izdelati razumljiv opis področja na katerega se nanaša razvoj aplikativnega sistema. Tako lahko ugotovimo katere funkcije in delovne procese mora **podpirati** informacijski sistem, kaj se izvaja v osnovnih funkcijah, katera pravila morajo funkcije upoštevati in katere podatke potrebujejo za svoje delovanje. Osnovna naloga opravil je izdelava konceptualnih modelov kot so podatkovni, funkcionalni in procesni model.

Naloga načrtovanja pa je izdelati načrt informacijskega sistema na podlagi modelov in ostalih specifikacij, ki so bili pridobljeni v fazi analize. S poznavanjem kapacitet in karakteristike strojne opreme, na kateri bo delovala aplikacija, lahko ustvarimo:

- načrt aplikativnega sistema, ki ustreza specifikacijam in upošteva tehnološke omejitve sistema,
- zasnovo strategije prehoda iz obstoječe na novo aplikacijo.

5.1.1 Pregled zahtev za izdelavo modula

Izdelani modul mora zajemati vse funkcionalnosti okolja VCL, od komunikacije z nadzornikom virtualnih strojev, do rezervacije slik. Med izdelavo pa je potrebno upoštevati vso dokumentacijo, dostopno na [5].

Modul za dodeljevanje virov

Vloga modula je zagotavljanje razpoložljivosti in priprava sistemskih slik za uporabnike virtualnega laboratorija. Ti med uporabo podajo zahtevo po rezervaciji okolja, ki se zabeleži v podatkovno bazo. Proces vcld pregleda, ali se v bazi nahajajo nove zahteve po rezervaciji okolij. Če prebrana zahteva nima ustrezne rezervacije, jo proces definira kot nov objekt tipa State, kate-

remu se nato dodeli ustrezen modul za dodeljevanje virov. Naloga izbranega modula je pripraviti virtualni stroj, ki odgovarja zahtevam rezervacije.

Pri izdelavi modula pa je potrebno paziti, da je le-ta zaradi varnosti dostopen znotraj modulov State.pm in OS.pm.

Priporočene funkcije za implementacijo so:

node_status()

Funkcija preveri trenutno stanje enote v katero se naloži rezervirana slika in na podlagi tega, vrne referenco tipa slovar (angl. *hash*). Ta vsebuje informacije o statusu rezervacije, stanju odzivnosti (angl. *ping*), možnosti povezave preko varne lupine (SSH, angl. *Secure Shell*) in nazivu trenutne slike (angl. *image name*). Najpomembnejša je status, ki sistemu pove ali je slika v celoti že pripravljena ali pa je potrebna ponovnega nalaganja.

capture()

Glavna naloga funkcije je zajetje trenutnega stanja slike. V principu se izvede kopiranje diska virtualnega stroja na izbrano lokacijo. Med izvajanjem se lahko izvrši klic *pre_capture()*, ki pripravi operacijski sistem na postopek zajemanja slike. Funkcija se uporabi v primeru dodajanja nove slike v okolje virtualnega laboratorija.

power_off()

Funkcija poskrbi za izklop virtualnega stroja. Pri tem lahko uporabi klice operacijskega sistema in ukaze nadzornika virtualnih strojev.

power_on()

Funkcija poskrbi za vklop virtualnega stroja. Kot rezultat izvedbe vrne numerično vrednost 0 ali 1.

power_reset()

Glavna naloga funkcije je ponovni zagon virtualnega stroja, katerega lahko izvedemo preko nadzornika virtualnih strojev in operacijskega sistema.

get_current_image()

Namen funkcije je ugotoviti, katero sliko vsebuje izbrana delujoča enota. Le-ta vsebuje zapis slike v datoteki *currentimage.txt*, ki se nahaja v imeniku korenskega uporabnika sistema.

get_image_size()

Naloga funkcije je ugotoviti velikost porabljenega prostora slike. Kot rezultat vrne numerično vrednost 0 ali 1.

load()

Funkcijo je potrebno obvezno implementirati, saj predstavlja jedro modula. Njene naloge so: izvedba procesa dodelitve slik, komunikacija z nadzornikom virtualnih strojev, kloniranje osnovne slike (angl. *base image*), izdelava konfiguracijskih datotek in obveščanje o napakah.

5.1.2 Pregled obstoječih modulov

Pri izdelavi modula smo si zaradi boljšega razumevanja problema pomagali z obstoječimi rešitvami, ki jih ponuja virtualni laboratorij. S tem načinom dela smo se izognili potencialnim in nepredvidljivim težavam, ki so vedno prisotne pri razvoju programske opreme, in ugotovili kateri deli so potrebni dopolnitve ali nadgradnje. Podrobneje smo analizirali **obstoječi** modul VMware Provisioning Module, ki skrbi za dodeljevanje virov na osnovi virtualizacije VMware.

VMware Provisioning module

Modul je namenjen upravljanju nadzornikov virtualnih strojev VMware Server 2.0, VMware ESX/ESXi 3.5 in VMware ESX/ESXi 4.0. Za izvrševanje ukazov uporablja vmesnik (vSphere API), razvojno knjižnico (vSphere SDK) in vmesnik ukazne vrstice (vim-cmd CLI), ki se nahajajo v modulu VMware.pm. Vsi ostali uporabni ukazi, ki so potrebni za oddaljeno interakcijo z nadzornikom virtualnih strojev, so locirani v podmapah modula VMware.

Razvojna knjižnica **vSphere SDK** ponuja:

- oddaljen dostop do virtualnih strojev in njihovih nadzornikov,
- dostop do podatkovnega sistema gostiteljev,
- upravljanje virtualnih strojev,
- komunikacijo za katero ne potrebujemo varne lupine.

Vmesnik ukazne vrstice (**vim-cmd CLI**) ponuja:

- oddaljen dostop do gostitelja,
- dostop do podatkovnega sistema z uporabo varne lupine,
- upravljanje virtualnega okolja v okviru omejene licence.

Ustvarjanje virtualnega stroja

Pri ustvarjanju virtualnega stroja se uporabi dodatna funkcija *prepare_vmx()*, ki na podlagi rezervacije pripravi konfiguracijsko datoteko. Pri tem se uporabi spremenljivka tipa razpršilne tabele (angl. *hash table*), ki vsebuje informacije o rezervaciji. Na podatkovnem sistemu gostitelja se ustvari mapa, v katero se shrani nastavitvena datoteka, ki je pripravljena na izvršitev. Nadzornik virtualnih strojev nato datoteko preveri in izvrši tako, da ustvari nov virtualni stroj, ki je pripravljen na dodelitev k zahtevani rezervaciji.

5.1.3 Zasnova modula Libvirt

Po temeljitem pregledu obstoječih modulov smo definirali funkcije, ki bodo poskrbele za komunikacijo z nadzornikom virtualnih strojev KVM. Pri postopku dodeljevanja virtualnih diskov in hitrega kloniranja smo se odločili za pisanje novih metod, saj se te že v osnovi razlikujejo od obstoječih v modulu VMware (Slika 5.1). Za delo s konfiguracijskimi datotekami smo uporabili format XML, ki je vodilo osnovne **standardizirane** komunikacije z vmesnikom Libvirt. Zaradi omejenosti virov je bila sprejeta realizacija le ključnih funkcij, ki so bile potrebne za izvedbo rezervacij virtualnih strojev. Vlogo ostalih funkcij pa je bilo potrebno nadomestiti s pripravo slik in ročnim ustvarjanjem osnovnih konfiguracijskih datotek. Realizirane **funkcije** modula Libvirt so:

initialize() - poskrbi za začetno inicializacijo modula,

poweron() – ustvari in hkrati tudi zažene virtualni stroj,

get_image_repository_path() – vsebuje informacijo o lokaciji slik,

get_clones_repository_path() – vsebuje informacijo o lokaciji klonov,

get_domain_base_xml_path() - vsebuje lokacijo osnovnih datotek,

get_clones_xml_path() - vsebuje lokacijo sestavnih datotek klonov,

check_datastore_disk_space() – preveri zasedenost podatkovnega prostora,

get_connection_uri() – ustvari pot za povezavo do gostitelja virtualizacije,

command_to_host() – izvaja ukaze na gostitelju virtualnih strojev,

create_xml() – ustvari konfiguracijske datoteke klonov,

load() – izvaja glavni del rezervacije virtualnih strojev,

capture() – zajame trenutno stanje slike in jo shrani na določeno lokacijo,

node_status() – preveri stanje virtualnega stroja,

does_image_exist() – preveri obstoj določene slike,

get_image_size() – poda velikost izbrane slike.

```
1  # generic script code of Libvirt.pm module
2  # code placed here is outside any entry point function
3
4  sub initialize {
5      # ... Insert your initialize code here
6  }
7
8  sub poweron {
9      # ... Insert your poweron code here.
10 }
11
12 sub get_connection_uri{
13     # ... Insert your get_connection_uri code here.
14 }
15 sub create_xml {
16     # ... Insert your create_xml code here.
17 }
18
19 sub load {
20     # ... Insert your load code here.
21 }
22 ...
```

Slika 5.1: Generični skriptni vir modula Libvirt.

5.2 Priprava gostitelja

Strojna oprema

Za izvajanje virtualizacije v testnem okolju je bilo potrebno izbrati ustrezno strojno opremo. Po dogovoru smo izbrali napredno delovno postajo HP XW 8600 s procesorjem Intel Xeon 5160, ki vsebuje vse dodatne ukaze za podporo **virtualizacije**. Zaradi predvidene možnosti večjega števila rezervacij je bilo potrebno uporabiti zadostno količino dinamičnega spomina (7 GB) in primerno velikost diska (250 GB). K izbrani konfiguraciji smo dodali še omrežni kartici, ki sta predstavljali javno oziroma zasebno omrežje virtualnega laboratorija (angl. *public/private network*).

Programska oprema

Vlogo operacijskega sistema smo dodelili okolju Centos 6.0, na katerega smo namestili modula KVM in QEMU ter knjižnico Libvirt. Pred tem smo v sistemu BIOS poskrbeli za prisotnost procesorskih ukazov, ki podpirajo virtualizacijo. Nameščeni modul KVM ne potrebuje dodatne konfiguracije, saj je le-ta samodejno izvedena.

Pri konfiguraciji **omrežja** smo ustvarili virtualna mostova (angl. *virtual bridges*), ki sta predstavljala javno oz. zasebno omrežje. Slednje smo ločili s strežnikom DHCP (angl. *Dynamic Host Configuration Protocol*), v katerega smo vnesli statične naslove namenjene virtualnim okoljem. Z uporabo tehnike usmerjanja paketov (angl. *routing*) smo povezali javno in testno omrežje ter tako zagotovili zunanji dostop virtualnim strojem.

Za uporabo **virtualnega prostora** smo uporabili lokalne diske gostitelja (DAS, angl. *Direct-attached storage*), na katerih smo ustvarili podatkovne mape za hrambo virtualnih strojev, njihovih klonov (Tabela 5.1) in konfiguracijskih datotek (Tabela 5.2).

Mapa	*Opis mape*
/datastore	korenska mapa
/datastore/clones	virtualni kloni slik *QCOW2*
/datastore/clone-xmls	konfiguracijske datoteke klonov slik

Tabela 5.1: Struktura map na strani gostitelja.

Mapa	*Opis mape*
/datastore	korenska mapa
/datastore/base	konfiguracijske datoteke osnovnih slik
/datastore/pools	konfiguracijske datoteke diskovnega prostora
/datastore/net	konfiguracijske datoteke omrežij

Tabela 5.2: Struktura map na strani upravljalne enote.

5.2.1 Namestitev programske opreme

Po uspešni namestitvi in konfiguraciji gostitelja virtualnih strojev je bilo potrebno ustvariti okolje virtualnega laboratorija. Najprej smo ustvarili virtualni stroj, na katerega smo namestili operacijsko okolje Centos 6.0. Pri tem smo si pomagali z vgrajenim grafičnim vmesnikom virt-manager, ki nam je olajšal delo pri začetni vzpostavitvi virtualnega okolja. Za tem je sledila namestitev naslednjih komponent:

- spletni strežnik Apache,
- šifrirni modul mod_ssl,
- knjižnica algoritmov libmcrypt,
- podatkovna baza MySQL Server,

- spletno orodje phpMyAdmin,
- skriptna jezika PHP in Perl z dodatnimi moduli,
- čelni in zaledni del virtualnega laboratorija,
- modul Sys-Virt,
- knjižnica Libvirt.

5.2.2 Konfiguracija okolja VCL

Podatkovna baza

Najprej je bilo potrebno na strežniku podatkovne baze MySQL ustvariti novo instanco z nazivom VCL, ki je hranila vse relacijske tabele potrebne za okolje virtualnega laboratorija. Za to smo uporabili ukazno vrstico mysql. Sledila je kreacija uporabnika, kateremu smo dodelili pravice administratorja. Za končno izgradnjo relacijskih tabel smo uporabili skripto *vcl.sql*, ki je samodejno ustvarila vse potrebne tabele in določevalne ključe.

PhpMyAdmin

Za konfiguracijo orodja PhpMyAdmin, smo uporabili navodila, ki so dostopna na proizvajalčevi domači strani [13]. Uporabniku smo omogočili spletni dostop do vmesnika, ki predstavlja komunikacijo s podatkovno bazo. V primeru zavrnjene povezave na orodje PhpMyAdmin je potrebno spremeniti varnostni kontekst spletnega strežnika.

Čelni del

Pri spletnem delu je bilo potrebno ustvariti uporabniški profil za komunikacijo s podatkovno bazo. Glede na to, da so bili vsi gradniki okolja VCL postavljeni v eno virtualno testno okolje, smo uporabili lokalni način dostopanja do baze virtualnega laboratorija.

Zaledni del

Pred zagonom strežniškega dela okolja je bilo potrebno konfigurirati ustrezne parametre v datoteki *vcld.conf*, kjer smo navedli:

- spletni naslov strežnika virtualnega laboratorija (FQDN, angl. *fully qualified domain name*),
- omrežni naslov strežnika podatkovne baze,
- uporabniški račun, ki ima pravice pisanja v podatkovno bazo,
- geslo podanega uporabniškega računa.

Varnost in oddaljen dostop

Osnovna komunikacija med sistemi okolja VCL poteka preko varne lupine, ki uporablja šifriranje z javnim ključem (angl. *public-key cryptography*). Za generiranje ključev pa se uporablja algoritem RSA (angl. *Rivest, Shamir and Adleman*).

DHCP

Nastavitev dinamičnega dodeljevanja omrežnih naslov smo izvedli z uporabo vmesnika Libvirt, preko katerega smo dodali statične zapise omrežnih naprav v konfiguracijsko datoteko privatnega omrežja. Za vsak rezerviran omrežni naslov smo izbrali unikatni naslov naprave ter naziv okolja z izbrano omrežno napravo. Pri tem je bilo potrebno paziti na podvajanje naslovov IP in MAC, saj bi v primeru enake dodelitve povzročili nepravilno delovanje okolja.

Prikaz omrežnih kartic z vlogo privatnega in javnega omrežja.

```
1 [root@ljkvm01 ~]# virsh iface-list
2 Name                 State      MAC Address
3 ---------------------------------------------------
4 lo                   active     00:00:00:00:00:00
5 private              active     fe:16:3e:3e:a8:4a
6 public               active     00:1a:4b:53:92:3a
```

Primer podrobnega izpisa privatnega omrežja s statičnimi zapisi.

```
1  [root@ljkvm01 ~]# virsh net-dumpxml private
2  <network>
3    <name>private</name>
4    <uuid>04ae72ce-0baf-ea4d-6d23-57a0b0404b60</uuid>
5    <bridge name='private' stp='on' delay='0' />
6    <ip address='192.168.122.10' netmask='255.255.255.0'>
7      <dhcp>
8        <range start='192.168.122.151' end='192.168.122.180' />
9        <host mac='00:16:3e:3e:a8:3a' name='slot4' ip='
             192.168.122.122' />
10       <host mac='00:16:3e:3e:a8:4a' name='slot5' ip='
             192.168.122.123' />
11       <host mac='00:16:3e:3e:a8:4c' name='slot6' ip='
             192.168.122.124' />
12       <host mac='00:16:3e:3e:a8:4e' name='slot7' ip='
             192.168.122.125' />
13       <host mac='00:16:3e:3e:a8:50' name='slot8' ip='
             192.168.122.126' />
14   </ip>
15 </network>
```

5.3 Programiranje rešitve

Med ustvarjanjem modula Libvirt smo se držali načela "lepega programiranja" in poskrbeli za smiselno in urejeno celoto modula. V vsakem primeru pa dopuščamo možnosti za bodoče izboljšave, nadgradnje in optimizacije kode in njenega delovanja. Komentarji, podani v razvitem modulu, so po dogovoru napisani v angleškem jeziku. Vsa predstavljena programska koda pa ima podano obrazložitev v slovenskem jeziku.

Do sedaj smo imeli postavljeno testno okolje, ki pa še ni bilo v celoti pripravljeno na virtualizacijo z nadzornikom virtualnih sredstev KVM. Potrebovali smo še realizacijo osnovanega modula za dodeljevanje virov, ki bo podpiral rezervacijo virtualnih strojev. Najprej je bilo potrebno izpolniti pogoje modularnosti, in sicer ponovna uporaba funkcije *load()* ter funkcije *initialize()*. Slednji smo zaradi lažjega pregleda nad testiranjem dodali le obvestili za sistem obveščanja (angl. *notification system*) in jo nato zaključili.

Kodni opis funkcije *initialize()*.

```
1 sub initialize {
2   notify($ERRORS{'DEBUG'}, 0, "Initializing libvirt module!");
3   notify($ERRORS{'DEBUG'}, 0, "Libvirt module initialized");
4   return 1;
5 }
```

Ker sistem VCL najprej preveri stanje virtualnega stroja, je bilo potrebno dodati funkcijo *node_status()*, ki preko podatkovnega objekta ugotovi naziv trenutne slike, odzivnost virtualnega stroja in možnost dostopa preko varne lupine.

Pridobitev podatkov iz podatkovnega objekta *data*

.

```
1   my $vmhost_hostname     = $self->data->get_vmhost_hostname;
2   my $vmhost_imagename    = $self->data->get_vmhost_image_name;
```

Funkcija nato medsebojno primerja parametre in v primeru ugotovitve neodzivnosti stroja ali napačne uporabe naložene slike, zahteva ponovni zagon virtualnega stroja. V nasprotnem primeru pa sporoči, da je zahtevana slika že pripravljena na uporabo.

V nadaljevanju je bilo potrebno implementirati obvezno funkcijo *load()*, katera naj bi na začetku pridobila vse potrebne podatke za ustrezno obdelavo. V ta namen smo uporabili referenčni sklic na podatkovni objekt rezervacije, s katerim smo pridobili dodatne podatke za bodočo komunikacijo z vmesnikom Libvirt. V spremenljivko *connection_uri* smo z uporabo funckije *get_connection_uri()* zapisali ustrezni gonilnik (angl. *driver*), ki skrbi za način dostopa do želenega nadzornika virtualnih strojev. Za dejansko povezavo pa smo uporabili modul **Sys-Virt** in pred tem pridobljeni gonilk ter preverili njeno uspešnost. V primeru neuspeha skripta obvesti sistem in prekine izvajanje rezervacije.

Kreiranje povezave *connection URI* z modulom *Sys::Virt*.

```
1  my $conn = Sys::Virt->new(uri=>$connection_uri);
```

Po uspešni povezavi je sledila izvedba zanke, ki se sprehodi skozi aktivne virtualne stroje, ter jih primerja z bodočo rezervirano enoto. Če ugotovi, da rezervirana enota že obstaja, jo izklopi, izbriše in obvesti sistem. S tem se izloči možnost prekrivanja poteklih rezervacij, ki bi zaradi napak v komunikaciji ostale v aktivnem stanju.
Za izgradnjo virtualnega stroja je bilo potrebno ustvariti konfiguracijsko datoteko z uporabo funkcije *create_xml()*, ki poskrbi za ustrezno nastavitev:

- univerzalnega unikatnega identifikatorja (UUID, angl. *universally unique identifier*),
- naziva virtualnega stroja,
- dinamičnega pomnilnika (RAM, angl. *Random-access memory*),
- rezervirane slike,

- ustreznega emulatorja,
- arhitekture in števila procesorjev,
- ustreznega klona slike.

Funkcija nato posodobi že obstoječo datoteko osnovne konfiguracije, ki vsebuje nastavitve za zvok, sliko, oddaljen dostop in omrežje. Z uporabo razčlenjevalnika besedil DOM (angl. *Document Object Model*) se ustvari končna konfiguracijska datoteka, ki izpolnjuje vse pogoje rezervacije. Zaradi boljše preglednosti in lažje identifikacije smo za naziv rezervacije stroja uporabili združeno ime (angl. *concatenated name*) virtualnega stroja in slike ter čas nastanka rezervacije.

Nastavitev imena slike s konkatenacijo spremenljivk *computer_name*, *image_name* ter *date*.

```
1 $doc->getElementsByTagName("name")->item(0)->getFirstChild()->
      setNodeValue("$computer_name-$image_name-$date")
```

Sledila je implementacija kloniranja slike, pred katero je bilo potrebno preveriti, ali je na lokaciji, ki je namenjena shranjevanju klonov, zadostna količina prostora. Pri tem se izvrši funkcija *check_datastore_disk_space()*, ki z uporabo funkcije *get_image_size()* izračuna velikost osnovne slike, katero nato primerja z velikostjo prostora, ki je na voljo. Če ugotovi, da je na lokaciji možna hramba klona, se izvajanje skripte nadaljuje, v nasprotnem primeru pa prekine. Po uspešni izvedbi kloniranja slike, se izvrši funkcija *poweron()*, ki ustvari in zažene virtualni stroj z zahtevano sliko.

Ustvarjanje virtualnega stroja iz konfiguracijske datoteke.

```
1 $connection->create_domain(\$dom_xml);
2 notify(\$ERRORS{'OK'}, 0, "VM started successfully!");
```

Ob uspešni dodelitvi rezerviranega okolja se uporabniku, ki je podal zahtevo, na spletnem vmesniku dodeli pravilni internetni naslov, preko katerega dostopa do rezerviranega okolja.

5.3.1 Knjižnica Libvirt

Knjižnica je dostopna preko klicev modula Sys-Virt. Ta vsebuje vse potrebne ukaze, ki so primerni za delo v okolju virtualnega laboratorija. Za skriptno uporabo knjižnice je potrebno ustvariti objekt (angl. *object*), ki predstavlja vmesnik Libvirt, in nato izvrševati klice ustreznih funkcij.

5.3.2 Izbira nadzornika virtualnih strojev

Izbira pravega nadzornika virtualnih strojev je predpogoj za učinkovito delo v virtualnem okolju, saj ta predstavlja glavno komponento povezave - *connection URI*, ki jo uporablja vmesnik Libvirt. Za izgradnjo povezave se izvede funkcija *get_connection_uri()*, ki preko poizvedbe SQL pridobi informacijo o virtualizacijski tehnologiji določenega gostitelja.

5.3.3 Kloniranje diska

Hitro kloniranje diska (angl. *fast clonnig*) je širše uporabljena metoda na področju računalništva v oblaku, ki zagotavlja hitre navidezne kopije. V primeru virtualnega laboratorija smo izbrali in uspešno implementirali način hitrega kloniranja vrste QEMU. Realiziran je v funkciji *load()*, v kateri ima vlogo predpogoja za kreiranje virtualnega stroja. Uspešno kloniranje se izvede takrat, ko so izpolnjeni naslednji pogoji:

- obstoj bazne slike v formatu QCOW2,
- definirana pot do bazne slike,
- zadostna količina prostega prostora na disku.

Klon ustvarimo z uporabo objekta *storage_pool*, ki predstavlja skupino definiranega prostora na disku. V primeru virtualnega laboratorija se izbere skupina z nazivom "*clones*", ki je namenjena za hranjenje virtualnih klonov. Funkciji *create_volume()* nato kot parameter podamo konfiguracijsko datoteko in tako ustvarimo klon bazne slike.

5.3.4 Ustvarjanje virtualnega stroja

Konfiguracijska datoteka ima ključno vlogo pri izdelavi virtualnega stroja, saj vsebuje poleg klasičnih sistemskih virov tudi podatke o zvočnih, omrežnih in drugih napravah, ki so prav tako pomembne za nadzornika virtualnih strojev. Ta mora omogočiti ustrezno emulacijo zahtevanih naprav. V primeru virtualnega laboratorija je to emulator QEMU, ki služi kot nadgradnja virtualizacije KVM. Za kreiranje virtualnega stroja je potrebna pravilno strukturirana konfiguracijska datoteka, ki jo podamo kot parameter funkciji *create_domain()* vmesnika Libvirt.

5.3.5 Komunikacija preko varne lupine

Sistemi okolja VCL uporabljajo varno lupino - SSH, za medsebojno komunikacijo, ki deluje na podlagi šifriranja z javnim ključem. V okviru razvitega modula je uporaba varne lupine izvedena znotraj ovojne funkcije (angl. *wrapper function*) *command_to_host()*. Ta jo nadgradi z možnostjo predvidevanja napak na omrežju kot sta slaba povezava in izguba omrežja. Nadgradnja je izvedena z zanko, ki izvaja določen ukaz varne lupine v časovnem intervalu, dokler se ta ne izvede uspešno oziroma zaključi. V tem primeru se predpostavi na prisotnost napak v omrežju.

5.4 Integracija

Po predpostavki, da se bo za integracijo v produkcijsko okolje uporabilo novo strojno opremo, je potrebno zagotoviti ustreznost naslednjih komponent:

strojna oprema – strežnik mora podpirati virtualizacijo in vsebovati zadostno količino dinamičnega pomnilnika (odvisno od uporabe števila rezervacij),

diskovni sistem – po načelu najboljše prakse je najbolj primerna uporaba omrežnih diskovnih sistemov SAN in NAS (angl. *storage area network*

in *network attached storage*) za hrambo baznih slik virtualnih strojev, medtem ko se lokalne diske gostitelja nameni virutalnim klonom slik,

omrežna oprema – vsi fizični in virtualni sistemi morajo vsebovati vsaj dve omrežni kartici, ki sta dane v javno oziroma zasebno omrežje (Slika 5.2).

V primeru integracije v obstoječe okolje virtualnega laboratorija pa je potrebno zagotoviti:

prisotnost poznavalca sistema

Ta mora ustrezno voditi postopek integracije in poznati zmožnosti, ki jih sistem ponuja. Kljub podpori odprtokodnih nadzornikov virtualnih strojev, modulu primanjkuje upravljanje večjih skupin gostiteljev oz. gruč (angl. *clusters*). Večina današnjih podjetij uporablja virtualizacijo ponudnika Microsoft, ki sicer velja za vrhunsko in učinkovito, vendar nima podanega preprostega vmesnika za odprtokodne tehnologije kot so: Perl, C in Python.

skladnost verzij modula in okolja VCL

Hiter in konstanten razvoj programske opreme prinaša novosti, spremembe in nadgradnje obstoječih sistemov. Virtualni laboratorij ni nobena izjema. Ta se pod okriljem fundacije Apache nenehno razvija in nadgrajuje, kar lahko privede do neskladnosti verzij oz. nepravilnega delovanja sistema.

skladnost modula Sys-Virt in knjižnice Libvirt

Za ustrezno uporabo vmesnika Libvirt je potrebno uskladiti verzije modula Sys-Virt, ki ustvarja objekte v modulu za dodeljevanje virov, in knjižnice Libvirt, ki izvaja funkcije modula na gostitelju. Uporabljena verzija modula Sys-Virt 0.2.4 je skladna s knjižnico Libvirt od verzije 0.8.1 dalje.

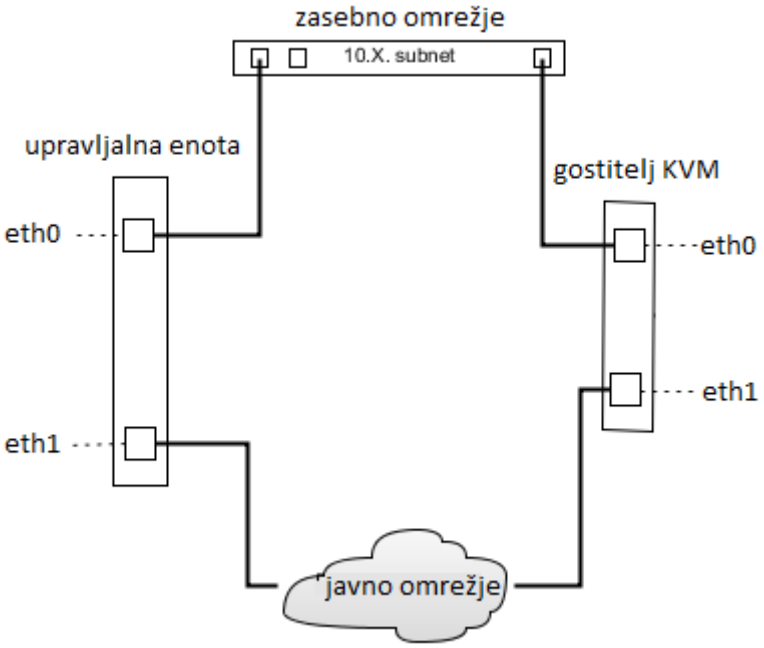
pretvorba formata obstoječih baznih slik

Pri uporabi hitrega kloniranja smo določili bazno sliko v formatu QCOW2.

Ta sicer ni obvezen, a vendar priporočljiv za izdelavo klonov baznih slik. Prav tako je možna uporaba neobdelanega formata (angl. *Raw image format*).

uporaba statičnega naslavljanja

Različica testnega okolja VCL (2.2.1) uporablja način statičnega dodeljevanja IP naslovov privatnega omrežja, ki je obvezen za konfiguracijo virtualnih strojev. V primeru, ko imamo storitev DHCP vezano na virtualni most, je potrebno v konfiguracijsko datoteko privatnega omrežja ročno navesti preslikave naslovov IP k ustreznim identifikatorjem omrežnih naprav (MAC, angl. *Media Access Control*). V ostalih primerih, ko je strežnik DHCP v ločenem okolju, lahko z uporabo spletnega vmesnika virtualnega laboratorija ustvarimo konfiguracijsko datoteko *dhcp.conf*, ki že vsebuje ustrezne preslikave naslovov.



Slika 5.2: Shema omrežja virtualnega laboratorija.

Postopek integracije modula v obstoječe okolje

1. Ustvarimo zahtevano imeniško strukturo na strani upravljalne enote, kot prikazuje tabela 5.2 v poglavju 5.2.

2. Ustvarimo imeniško strukturo na strani gostitelja, kot prikazuje tabela 5.1 v poglavju 5.2.

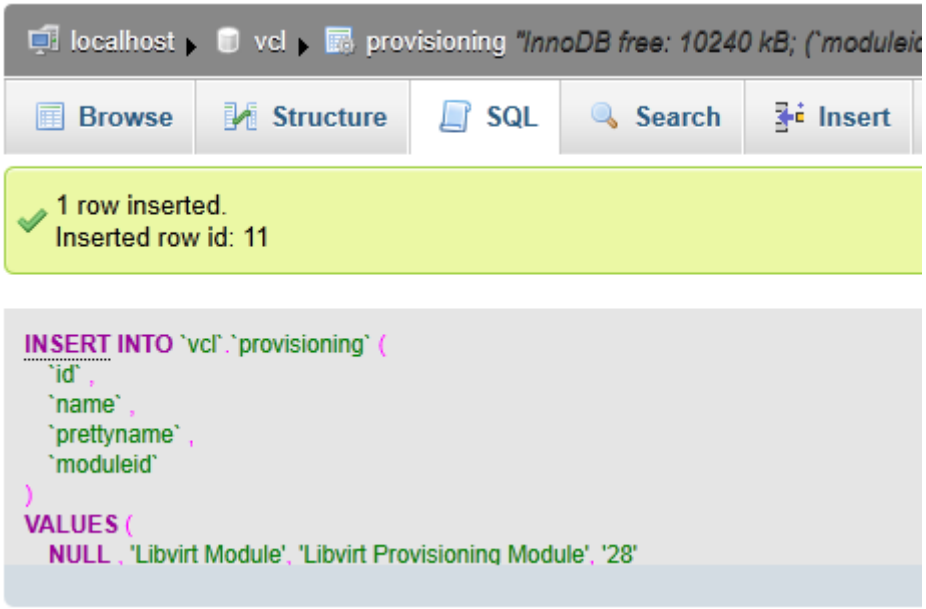
3. Ustvarjeni modul *Libvirt.pm* premaknemo v imeniško strukturo zalednega dela virtualnega laboratorija, in sicer na lokacijo: */usr/local/vcl/lib/VCL/Module/Provisioning/*.

4. V tabelo *module* dodamo modul Libvirt tako, da izpolnimo zahtevana polja preko vmesnika PhpMyAdmin (Slika 5.3).

Slika 5.3: Vstavljanje modula Libvirt v tabelo *module*

.

5. V tabelo *provisioning* dodamo zapis o modulu Libvirt in ga povežemo z *moduleid* (Slika 5.4).



Slika 5.4: Vstavljanje zapisa modula Libvirt.

6. Ustvarimo gostitelja KVM z modulom *Libvirt Provisioning Module*, katerega smo dodali v tabelo *provisioning*.

5.5 Uporaba spletnega vmesnika

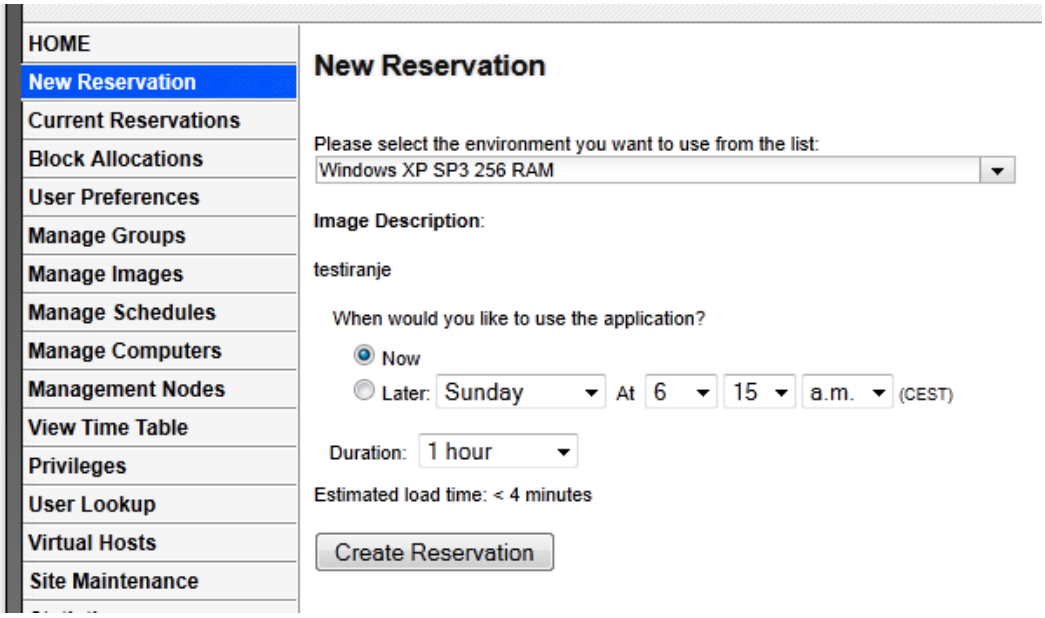
Spletni vmesnik oz. portal predstavlja vstopno točko za uporabnike virtualnega laboratorija, s katerim lahko izvajamo vsa potrebna opravila za upravljanje rezervacij virtualnih strojev, kot tudi administratorsko vzdrževanje.

Ustvarjanje rezervacije

Namen rezervacije je dodeliti ustrezno okolje, ki ga je uporabnik izbral, za določeno obdobje oz. časovni interval (Slika 5.5).

Za ustvarjanje rezervacije je potrebna izvedba sledečega postopka.

1. V levem navigacijskem meniju izberemo gumb *New Reservation*, ki določi novo rezervacijo.
2. Iz padajočega menija izberemo želeno okolje.
3. V primeru takojšnje rezervacije izberemo gumb *Now*, v nasprotnem primeru pa *Later* in določimo čas bodoče rezervacije.
4. V polju *Duration* določimo čas trajanja rezervacije.
5. Z uporabo potrditvenega gumba *Create Reservation* ustvarimo zahtevo po rezervaciji in postopek zaključimo.

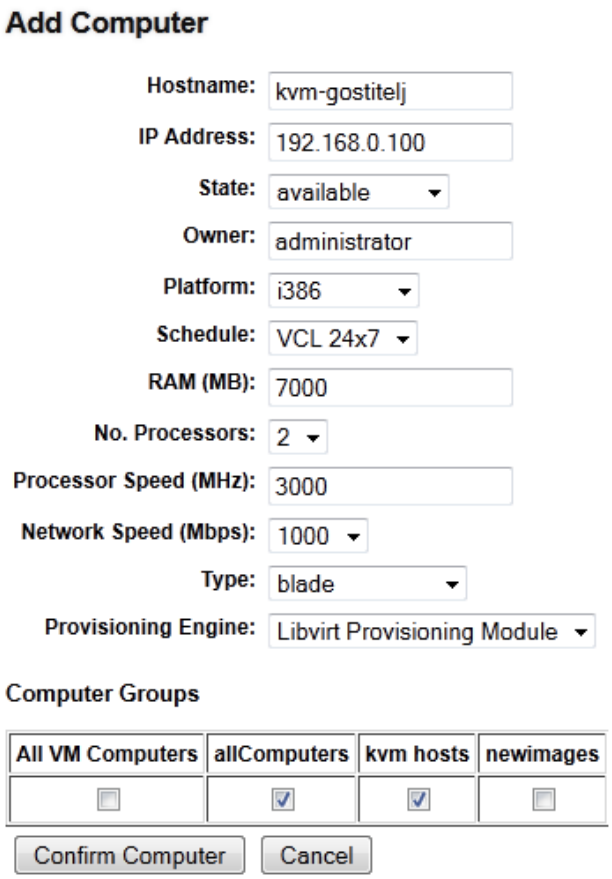


Slika 5.5: Rezervacija okolja *Windows XP SP3 256 RAM* za obdobje ene ure.

Dodajanje gostitelja KVM

Pred uporabo nadzornika virtualnih strojev je potrebno registrirati gostitelja virtualizacije v okolje VCL (Slika 5.6). Za uspešno dodelitev gostitelja KVM v okolje virtualnega laboratorija je potrebna izvedba sledečega postopka.

1. V levem navigacijskem meniju izberemo gumb *Manage Computers*.
2. Pod odprtem razdelku označimo možnost *Edit Computer Information* in potrdimo izbiro s pritiskom na gumb *Submit*.
3. V prikazani tabeli s pritiskom na gumb *Add* odpremo formo za vnos računalnika.
4. Izpolnimo zahtevana polja in izbiro potrdimo s pritiskom na gumb *Confirm Computer*.

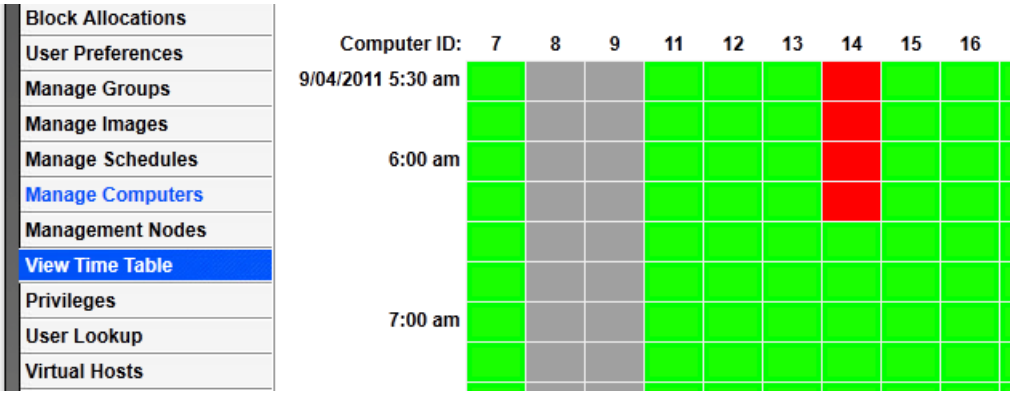


Slika 5.6: Dodajanje gostitelja KVM s sistemom za dodeljevanje virtualnih strojev Libvirt.

Pregled rezervacij

Preden rezerviramo virtualno okolje si lahko ogledamo proste termine, ki so na voljo (Slika 5.7). Za izpis urnika rezervacij je potrebna sledeča izvedba.

1. V levem navigacijskem meniju izberemo gumb *View Time Table*.
2. V seznamu ponujenih urnikov izberemo urnik celotnega dne *VCL 24x7*.
3. Izbiro potrdimo s pritiskom na gumb *Submit*.



Slika 5.7: Pregled prostih (zelena barva), zasedenih (rdeča) in neaktivnih (siva) rezervacij v določenem časovnem intervalu.

Poglavje 6

Testiranje delovanja sistema v testnem okolju

Tradicionalni pristop razvoja informacijskega sistema postavlja testiranje v eno izmed samostojnih in pomembnih faz razvojnega procesa. V navadi jo začnemo izvajati po integraciji in izvedbi sistema, a novejši pristopi opozarjajo na študije, ki uvrščajo testiranje kot aktivnost, katero je potrebno izvajati ves življenjski cikel, znotraj in na koncu vsake faze.

Postopek testiranja se izvaja z namenom, da se zagotovi pravilnost razvoja informacijskega sistema in skladnost z zahtevami, ki so bile podane na začetku realizacije projekta. Uvedba postopkov testiranja v vse faze razvoja zagotovo prinaša veliko koristi. V začetnih fazah neprestano preverjamo, ali rešitev gradimo pravilno oziroma v skladu s podanimi zahtevam. V zadnjih fazah pa poizkušamo zagotoviti zmanjšanje možnosti prekinitev in nedelovanja sistema v produkcijskem okolju. Končni rezultat pa je izgradnja kakovostnega produkta, ki najbolj izpolni zahteve okolja.

Testiranje programa lahko pokaže na prisotnost napak, ne more pa dokazati, da napak v programu ni! [3].

Namen testiranja je dokazati prisotnost ene ali več napak, ne pa tudi njihovo lokacijo. Bolj podrobno pa je potrebno preveriti in zagotoviti, ali ustvarjena rešitev ustreza zastavljenim zahtevam sistema.

6.1 Načrt testiranja

Pogosto se zgodi, da se k testiranju sistema uporabi nenačrtovan in slabo organiziran pristop, ki zmanjša kakovost rešitve oz. produkta. To izboljšamo tako, da pripravimo:

- natančen načrt testiranja,
- izberemo prave tehnike in postopke testiranja,
- uporabimo pravi postopek dokumentiranja in sporočanja napak,
- določimo prave izvajalce sistema.

Največja težava, ki se pojavi je postavitev ustreznih pravil, ki narekujejo kako testiranje izvesti, saj se informacijski sistemi med seboj v veliki meri razlikujejo.

V testnem okolju virtualnega laboratorija smo za izgradnjo načrta testiranja najprej določili:

predmet testiranja
: Kot testni subjekt smo določili tipične primere uporabe testnega okolja virtualnega laboratorija.

izvajalca testiranja
: Vlogo izvajalcev testiranja smo dodelili uporabniku in razvijalcu virtualnega laboratorija.

čas izvedbe testiranja
: Testiranje je bilo izvedeno po zaključeni izdelavi prototipne verzije virtualnega laboratorija.

Za pripravo testnih primerov smo se odločil **simulirati** najbolj pogoste dogodke, ki se zgodijo ob uporabi virtualnega laboratorija, in sicer:

- rezervacija virtualnega okolja brez dodatnih zahtev,
- podaljšanje obstoječe rezervacije,
- predčasna prekinitev obstoječe rezervacije.

Pri organizaciji in pripravi testiranja lahko pride do določenih odstopanj oziroma nevarnosti, ki onemogočijo izvedbo plana v skladu s pričakovanji. V ta namen smo se odločili, da se v fazo testiranja vključi osebi, ki imata zadostno količino znanja pri delu z virtualnem laboratorijem in dejanskim testiranjem programske opreme. Obema je bil razložen načrt in proces izvedbe testiranja. Za izvedbo testiranja smo uporabili testno okolje, ki je podrobneje razloženo v poglavju št. 5.2.

6.2 Izvedba testiranja

Testiranje je bilo razdeljeno na dve **fazi**. V prvi fazi sta oba osebi uporabnik oziroma razvijalec istočasno opravila niz testnih primerov v enakem zaporedju. Pred začetkom izvedbe testiranja je bil vsakemu dodeljen uporabniški račun uporabnik oz. razvijalec s pravicami, ki so omogočale delo z rezervacijami.

Uporabnik in razvijalec sta najprej preko spletnega vmesnika istočasno podala zahtevo po rezervaciji virtualnega stroja in počakala na njeno dokončno izvršitev. Ko je sistem obvestil oba uporabnika o uspešni dodelitvi virov, sta jo oba podaljšala za vrednost ene ure in potrdila novo zahtevo. Po tem sta ponovno čakala dve minuti, da je sistem obdelal sprejeto zahtevo. Med čakanjem pa nista opravila nobene interakcije s spletnim vmesnikom. Po izteku dveh minut sta za zaključek hkrati oddala novi zahtevek za prekinitev prej odobrene rezervacije in ob izvršitvi zaključila postopek testiranja zahtevanih funkcionalnosti virtualnega laboratorija.

V drugi fazi se je v enakem zaporedju izvedel podoben postopek kot v prvi fazi le, da je razvijalec pričel s procesom testiranja dve minuti pozneje kot uporabnik. Med izvajanjem testiranja smo skrbno **beležili** rezultate in opazke, ki smo jih dobili bodisi med izvedenim testiranjem bodisi preko testnega sistema virtualnega laboratorija.

6.3 Rezultati testiranja

Delo testne skupine je potekalo nemoteno in po predpisanem načrtu. Preizkuševalca sta se ustrezno držala procesnega načrta izvedbe testiranja, tako da med testiranjem in časovnim izvajanjem ni bilo opaženih napak oziroma napačnega delovanja sistema. V tem primeru se opazi, da je bila faza testiranja prisotna že v času razvoja programske rešitve in s tem posledično zagotovila odpravo večine napak.

Udeleženca sta testirala zahtevane funkcionalnosti na svojih delovnih postajah, kjer sta se preko spletnega vmesnika povezala v okolje virtualnega laboratorija in uspešno izvedla proces testiranja. Končna **ocena** testiranja sistema v testnem okolju je zadovoljiva.

Za bodoča testiranja bi lahko poleg večjega števila preizkuševalcev uporabili tudi računalniško podprto obremenitveno testiranje (angl. *stress testing*) s katerim bi lahko, na primeru masovnih rezervacij ter velikega števila izdelanih klonov, preverili stabilnost komponent sistema, kot so: podatkovna baza, procesor, dinamični spomin in diskovni sistem.

Poglavje 7

Sklepne ugotovitve

Pri izdelavi diplomske naloge smo z razvojem modula Libvirt izvedli integracijo vmesnika Libvirt v okolje virtualnega laboratorija. Razviti modul omogoča uporabo standardiziranih ukazov za upravljanje nadzornikov virtualnih strojev KVM in učinkovito izdelavo klonov virtualnih strojev. Integracijo smo v celoti izvedli z uporabo odprtokodnih tehnologij in orodij.

Glavni **prispevki** so v dodani podpori za virtualizacijsko tehnologijo KVM in QEMU, ki sta sicer osnovane na odprti kodi, a ravno tako enakovredne obstoječim in plačljivim tehnologijam virtualnega laboratorija (VMware). Uporaba emulatorja QEMU omogoča pohitreno kloniranje virtualnih strojev, pri tem pa porabi bistveno manj podatkovnega prostora in posledično diskovni sistem. Z vpeljavo vmesnika smo močno povečali izrabo funkcij virtualnega laboratorija in omogočili enakovredno gostovanje storitev v računalniškem oblaku.

Kljub temu, da smo v ospredje programske rešitve uspešno postavili **učinkovit** nadzornik virtualnih strojev KVM, se je pomembno držati načela standardizacije in tako v prihodnje **nadgraditi** oz. izpopolniti sistem za ostale virtualizacijske tehnologije kot so: XEN, OpenVZ ter VIrtualBox. Prav tako bi bilo zanimivo izvesti njihovo medsebojno primerjavo s pomočjo različnih testiranj.

Ne glede na uspešno delovanje modula, je bilo pri razvoju videti več **težav**. Ena od funkcionalnosti modula je pridobivanje zasebnega naslova IP, ki se ob ponovnem zagonu sistema ne izvede pravilno. V tem primeru je potrebno ponovno zagnati proces ustvarjanja zasebnega omrežja.

Postopek izgradnje konfiguracijskih datotek bi lahko avtomatizirali z uporabo funkcije, ki bi preverila strojne zmogljivosti nadzornika virtualnih strojev in na podlagi dobljenih podatkov ustvarila datoteko z ustrezno konfiguracijo.

Možnosti **nadaljnjega** razvoja vidimo predvsem v testiranju programske opreme, za katero bi bilo morda primernejše uporabiti sistem (CI, angl. *continuous integration*), ki bi za vsak implementirani del programske rešitve samodejno simuliral postopek rezervacije virtualnega stroja in tako povečal kvaliteto rešitve že v fazi razvoja opreme.

Literatura

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the clouds: A Berkeley View of Cloud Computing", University of California at Berkeley, št. UCB/EECS-2009-28, 2009. (Citirano 2.9.2011) Dostopno na naslovu: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[2] S. Averitt, M. Bugaev, A. Peeler, H. Shaffer, E. Sills, S. Stein, J. Thompson, M. A. Vouk, "Virtual computing laboratory (VCL)" v zborniku *In the proceedings of the international conference on virtual computing initiative*, North Carolina, Združene države Amerike, maj 2007, pp. 1-16

[3] F. Solina, Projektno vodenje razvoja programske opreme, 1. izdaja. Ljubljana: Fakulteta za računalništvo in informatiko, 1997, 212 str.

[4] M. A. Vouk, "Cloud computing-issues, research and implementations", Journal of computing and information technology, št. 16, zv. 4, str. 235-246, 2008

[5] (2011) Apache VCL. Dostopno na: https://cwiki.apache.org/VCL/

[6] (2011) VCL Architecture. Dostopno na: https://cwiki.apache.org/confluence/display/VCL/VCL+Architecture

[7] (2011) VMware consolidation. Dostopno na:
http://www.vmware.com/solutions/consolidation/

[8] (2011) Driver-based architecture of Libvirt. Dostopno na:
http://www.ibm.com/developerworks/linux/library/l-libvirt/

[9] (2011) Libvirt: The virtualization API. Dostopno na:
http://libvirt.org/

[10] (2011) About - QEMU. Dostopno na:
http://wiki.qemu.org/Main_Page

[11] (2011) Sys-Virt. Dostopno na:
http://search.cpan.org/dist/Sys-Virt/

[12] (2011) The Perl Programming Language. Dostopno na:
http://www.perl.org/

[13] (2011) PhpMyAdmin documentation. Dostopno na:
http://www.phpmyadmin.net/documentation/

[14] (2011) KVM. Dostopno na:
http://www.linux-kvm.org/page/Main_Page

[15] (2011) Eclipse-EPIC. Dostopno na:
http://www.epic-ide.org/

[16] (2011) An introduction to Virtualization. Dostopno na:
http://www.kernelthread.com/publications/virtualization/