UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Borič

Integracija virtualnega računalniškega laboratorija NCSU z oblačno platformo OpenStack

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Borič

Integracija virtualnega računalniškega laboratorija NCSU z oblačno platformo OpenStack

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Mojca Ciglarič

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite zasnovo in implementacijo odprtokodnega sistema – virtualnega laboratorija NCSU VCL. Preučite, kako uporablja spodaj ležeč virtualizacijski sistem in kakšno zasnovo ima modul za oskrbovanje uporabnikov z navideznimi računalniki. Preučite tudi odprtokodno oblačno platformo OpenStack. Razmislite, kako bi lahko namesto virtualizacijskega sistema trenutne izvedbe v NCSU VCL vključili oblačno platformo za oskrbovanje uporabnikov z navideznimi računalniki. Izberite najobetavnejšo zasnovo, izdelajte podrobnejši načrt izvedbe in razvijte prototip, ki bo služil kot dokaz izvedljivosti. Za ta dokaz zadostuje, da prek vmesnika VCL zaženete navidezni računalnik v OpenStacku in se povežete nanj. Integracijo kritično ovrednotite.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Boštjan Borič, z vpisno številko **63000328**, sem avtor diplomskega dela z naslovom:

Integracija virtualnega računalniškega laboratorija NCSU z oblačno platformo OpenStack

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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 30. oktobra 2014 Podpis avtorja:

Želel bi se zahvaliti doc. dr. Mojci Ciglarič, ki me je v zadnjem hipu spravila na prava pota. Njeno žuganje z Davidom Forsythom je padlo na plodna tla. Zahvaljujem se lektorju Primožu Zevniku, ki je ves čas neumorno bdel nad besedilom. Od sedaj naprej mu bom še z večjo vnemo popravljal njegove računalnike. Zahvaljujem se še Matjažu Boriču za njegove vestne popravke citatov, Kristijanu Sedlaku za izposojeno strojno opremo, Vesni Zevnik za lektoriranje povzetka v angleščini, Darku Joliču in vsem, ki so trepetali in trpeli skupaj z menoj.

Posebna zahvala gre staršem za njihovo potrpežljivost, ki je bila v poslednjih mesecih pred zaključkom diplomske naloge pogosto na hudi preizkušnji.

"Oblaki, rastline, zivali, minerali, svetovja, vse zivi v svojem miru-nemiru brez zavesti; delci se privlacijo in odbijajo brez slasti in muke; samo ljudje smo skrbno izumili lastna mucilna orodja, s katerimi se obdelujemo v budnosti in v sanjah."

Vitomil Zupan, Igra s hudičevim repom

Kazalo

Povzetek

V diplomski nalogi predstavimo koncept virtualizacije in računalništva v oblaku. Pri tem se osredotočimo na model storitve v računalniškem oblaku, ki uporabniku omogoča vzpostavitev virtualizirane infrastrukture. Ogledamo si dva sistema, ki omogočata rezervacijo sistema Virtual Computing Lab, ki je bil razvit na Državni univerzi Severne Karoline. Ta študentom in raziskovalcem omogoča rezervacijo in oddaljen dostop do fizičnih in virtualnih računalnikov. Predstavimo tudi odprtokodno oblačno platformo OpenStack, ki uporabnikom omogoča vzpostavitev zasebnega računalniškega oblaka. Ta sicer nudi širok nabor storitev, ne pa tudi časovno omejenih rezervacij, kot jih omogoča Virtual Computing Lab. Zato preučimo možnost integracije obeh sistemov in predstavimo razvoj modula za Virtual Computing Lab, ki uporabnikom omogoča oskrbovanje z računskimi viri iz zasebnega računalniškega oblaka, zgrajenega na platformi OpenStack.

Ključne besede: računalništvo v oblaku, virtualizacija, OpenStack, Virtual Computing Lab, infrastruktura kot storitev

Abstract

The thesis presents the concept of virtualization and cloud computing. It focuses on the infrastructure as a service model of cloud computing, which enables customers to set up their own virtualized infrastructure. The thesis takes a closer look at the Virtual Computing Lab, which was developed at the State University of North Carolina in order to provide students and researchers with remote access to computing resources. OpenStack, an open-source software cloud computing platform is also presented. Despite offering a wide array of services, it does not have the option of time-limited computing resource reservations as provided by Virtual Computing Lab. Therefore, the option of integrating the two systems is considered and a module is presented for the Virtual Computing Lab upgrade, which enables users to provide compute resources from a private cloud built on the OpenStack platform.

Keywords: cloud computing, virtualization, OpenStack, Virtual Computing Lab, infrastructure as a service

Poglavje 1 Uvod

Študentje se pri študiju računalništva in informatike pogosto srečujejo s problemi in nalogami, ki potrebujejo specifično kombinacijo programske in strojne opreme. V preteklosti so fakultete ob predhodni rezervaciji študentom omogočale sicer omejen dostop do računalniških laboratorijev. Da bi poenostavila postopke rezervacije in avtomatiziranega dodeljevanja računalnikov, je Državna univerza Severne Karoline (NCSU) razvila sistem, imenovan Virtual Computing Lab (VCL). Ta poleg dodeljevanja fizičnih računalnikov omogoča ustvarjanje navideznih računalnikov, kar omogoča t.i. virtualizacija, tvorba programsko simuliranih računalnikov. S pomočjo virtualizacije so na univerzi NCSU obstoječe fizične strežnike njihovega računskega centra preoblikovali v manjše virtualne računalnike, ki jih študentje in raziskovalci preko VCL-jevega spletnega vmesnika rezervirajo in uporabljajo na daljavo.

Medtem ko je bil VCL razvit primarno za raziskovalne in pedagoške namene, je dve leti po njegovem nastanku podjetje Amazon predstavilo komercialno storitev: Elastic Compute Cloud, ki ima podobno idejno zasnovo. Amazonova storitev je postala v svetu zelo popularna in je povzročila bliskovit razmah računalništva v oblaku. Ta storitev uporabnikom nudi možnost, da vzpostavljajo lastne infrastrukture na osnovi navideznih računalnikov. Za vzdrževanje in nemoteno delovanje fizičnih strežnikov, ki so podlaga navideznih, skrbi osebje podatkovnega centra tega velikega komercialnega ponudnika. Uporabnik z navideznimi strežniki rokuje na programskem nivoju ter jih izjemno hitro dodaja ali odvzema glede na lastne potrebe po zmogljivosti. Hkrati pa se ogne velikim stroškom investicije v strojno opremo.

Zato je uspeh Amazonove storitve je pritegnil številne posnemovalce. Eden izmed takih je OpenStack, projekt, ki nastal kot plod sodelovanja med podjetjem RackSpace in ameriško vesoljsko agencijo NASA. Projektu so se pridružila še velika imena softverske industrije kot so IBM, Oracle idr. OpenStack je v osnovi nekomercialne narave s prosto dostopno programsko kodo in omogoča vzpostavitev zasebnega računalniškega oblaka po vzoru Amazonove storitve.

Fakulteta za računalništvo in informatiko je v preteklosti s pomočjo NCSU-jevega Virtual Computing Laba študentom omogočala izposojo virtualnih računalnikov, na katerih so ti opravljali laboratorijske vaje pri pri predmetu Računalniške komunikacije. Fakluteta je pri tem kot virtualizacijsko komponento, s katero komunicira VCL, uporabljala komercialni produkt podjetja VMware. Slednje izobraževalnim ustanovam nudi brezplačne licence, a te omejujejo razvoju lastnih komercialnih rešitev na VMwarovi osnovi, prav tako pa njihova programska koda ni prosto dostopna. FRI-jev Laboratorij za računalniške komunikacije se je zato odločil za poizkus povezave VCL-ja z odprtokodnim OpenStackom kot alternativo VMwarovi infrastrukturi z bistveno svobodnejšimi pogoji licenciranja.

Cilj te diplomske naloge je bil, da z njo preučim možnost povezave Virtual Computing Laba z zasebnim oblakom vzpostavljenim na OpenStackovi platformi. OpenStack na zahtevo uporabnika sicer dodeljuje navidezne računalnike, ne nudi pa rezervacijskega sistema s časovno omejenimi rezervacijami, kot to omogoča VCL. Naloga je vključevala vzpostavitev miniaturnega testnega oblaka. Zato sem moral preučiti vmesnike, ki jih OpenStack nudi programerjem, se poglobiti v delovanje Virtual Computing Laba, slediti poteku programske kode, ki skrbi za VCL-jeve rezervacije in na podlagi pridobljenih znanj VCL nadgraditi z novim modulom za oskrbovanje z navideznimi računalniki iz OpenStackovega oblaka.

Poglavje 2 Virtualizacija

2.1 Kratka zgodovina

Pri obravnavi računalniških oblakov, predvsem takšnih, ki uporabniku omogočajo rezervacijo in poln nadzor nad dodeljenim navideznim računalnikom (t.i. infrastruktura kot storitev, angl. infrastructure as a service, IaaS), ne moremo mimo virtualizacije. Ideja poganjanja večih medsebojno izoliranih operacijskih sistemov znotraj t.i. virtualnih računalnikov, ki delujejo na strojni opremi enega samega računalnika, sega v 60. leta prejšnjega stoletja, ko so IBM-jevi inženirji razvili sistema SIMMON in CP-40 [1:474]. Z nižanjem cene strojne opreme v 70. letih in še posebej z lažjim dostopom do računalnikov - prihod osebnih računalnikov IBM PC konec 70. in v začetku 80. let - je hkrati tudi virtualizacija pričela izgubljati svoj pomen. K temu je dodatno prispevalo dejstvo, da ukazne arhitekture novo razvitih mikroprocesorjev v 80. in 90. letih, vključno z Intelovo x86, niso bile zasnovane v smeri olajšanja razvoja virtualizacijskih rešitev [3].

Velik skok naprej se je zgodil šele konec 90. let, ko je skupina raziskovalcev stanfordske univerze leta 1998 ustanovila podjetje VMware. Že leta 1999 so izdali VMware Workstation, ki je pomenil mejnik v virtualizaciji na osebnih računalnikih IBM PC [3]. Leta 2001 je skladno s širjenjem arhitekture x86 v podatkovne centre VMware izdal še strežniško različico, t.i. VMware ESX Server. VMwarovi izdelki so povzročili popularizacijo in razmah virtualizacije na ahrhitekturi x86. Leta 2003 se je pojavila odprtokodna rešitev Xen, ki pa v primerjavi z VMwarovimi izdelki ni nudila popolne virtualizacije, temveč t.i. paravirtualizacijo. Na naraščajoč pomen virtualizacije v letih, ki so sledila, kaže dejstvo, da v letu 2006 tako Intel kot njihov tedaj glavni konkurent AMD svojim mikroprocesorjem dodala razširitve (VT-x in AMD-V), ki so bistveno olajšale razvoj kasnejših virtualizacijskih rešitev (polna virtualizacija z Xen-om, odprtokodni KVM, Microsoftov Hyper-V) in nato z vsako naslednjo generacijo izboljšale učinkovitost izvajanja virtualizacije na arhitekturi x86.

2.2 Opis delovanja

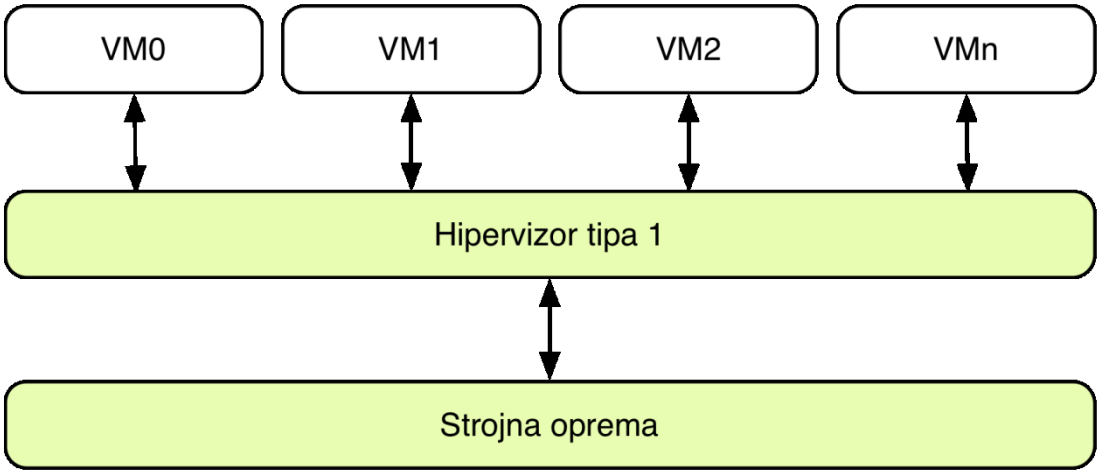
VMware Workstation in ESX Server, IBM-jev CP-40 ter druga imena zgornjega odstavka so t.i. nadzorniki virtualnih strojev/nadzorniki navideznih računalnikov (angl. Virtual Machine

Monitors, VMM), imenovani tudi hipervizorji. Hipervizorji tipa 1 (VMware ESX in ESXi, Xen, KVM, Hyper-V) predstavljajo tanek sloj programske opreme tik nad strojno opremo in zasedajo mesto, ki običajno pripada operacijskemu sistemu. Delujejo v t.i. priviligiranem procesorskem načinu in imajo popoln nadzor nad spodaj ležečo strojno opremo. Na zahtevo uporabnika tvorijo navidezne računalnike in jim dodeljujejo računske in pomnilniške vire, podobno kot tradicionalni operacijski sistem skrbi za razvrščanje oz. dodeljevanje virov uporabniškim procesom.

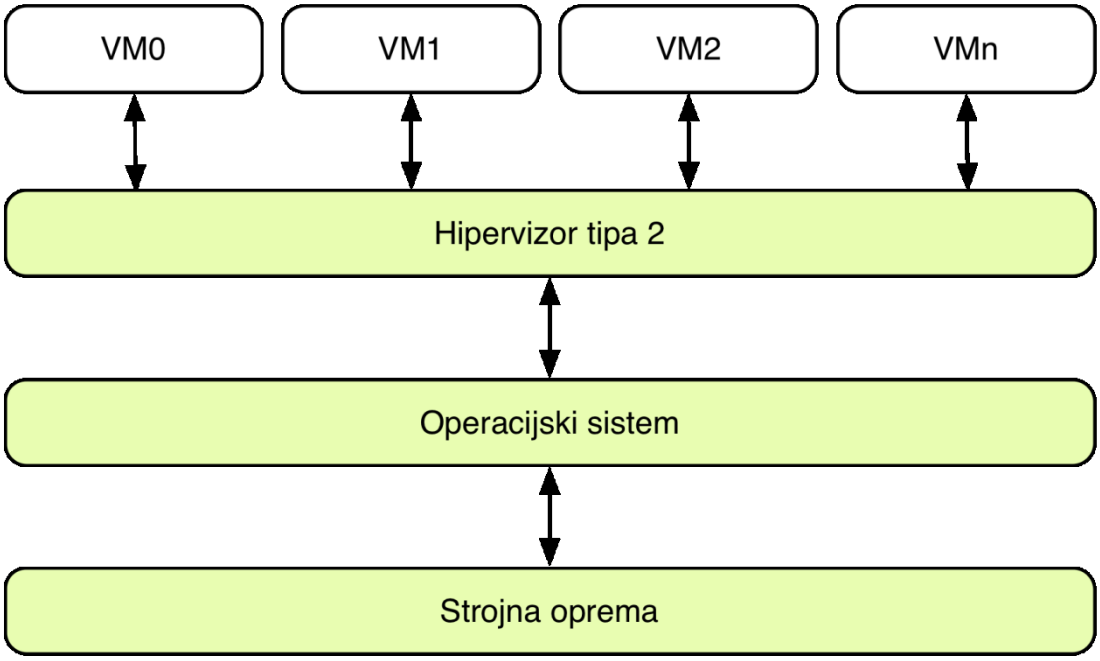
Hipervizorjevi navidezni računalniki so v svojem bistvu v resnici uporabniški procesi, znotraj katerih se namesto uporabniških programov izvaja programska koda celih operacijskih sistemov. Operacijski sistemi so izpodrinjeni iz svojega priviligiranega položaja in svoje ukaze izvajajo v depriviligiranem, uporabniškem načinu, kar predstavlja osrednji zaplet virtualizacije. Da bi hipervizorji za operacijske sisteme znotraj svojih virtualnih računalnikov ustvarili iluzijo, da v resnici delujejo v priviligiranem načinu in imajo popoln nadzor nad strojno opremo, morajo prestrezati in sproti prevajati njihove poizkuse izvajanja priviligiranih ukazov.

Glavni razlog, da so se virtualizacijske rešitve tako pozno pričele uveljavljati na osebnih računalnikih IBM PC, je bila ravno velika kompleksnost prestrezanja in prevajanja priviligiranih ukazov. Zelo je k temu pripomogla tudi nekooperativnost Intelove arhitekture x86 [3]. Xenov alternativni pristop t.i. paravirtualizacije zaobide problem prevajanja ukazov, a hkrati zahteva spremembe jedra virtualiziranega operacijskega sistema.

VMware Workstation skupaj s kasnejšimi produkti kot so VirtualBox in Parallels Desktop predstavjajo t.i. hipervizorje tipa 2, ki v primerjavi s hipervizorji prvega tipa za dostop do strojne opreme uporabljajo obstoječi že nameščeni operacijski sistem, ki je v vlogi t.i. gostiteljskega operacijskega sistema (angl. host operating system). Primernejši so za rabo na namiznih računalnikih, medtem ko so hipervizorji svoj dom našli predvsem v strežniškem okolju. Razliko med obema tipoma prikazujeta sliki 1 in 2.



Slika 1: Hipervizor tipa 1



Slika 2: Hipervizor tipa 2

2.3 Prednosti virtualizacije

Poleg osrednje funkcionalnosti, tj. vzporednega izvajanja več med seboj povsem izoliranih operacijskih sistemov na strojni opremi enega samega računalnika, virtualizacija prinaša še mnoge druge ugodnosti. Mnoge izmed njih s pridom izkorišča tako Virtual Computing Lab kot oblaki, ki nudijo storitev IaaS:

- možnost zajema stanja navideznega računalnika in kasnejšo povrnitev v stanje, ko je bil posnetek tega stanja narejen (angl. snappshotting, tudi checkpointing),

- prenos navideznih računalnikov z enega na drug fizični računalnik, ne da bi bilo pri tem potrebno spreminjati nastavitve programske opreme znotraj navideznega računalnika (v okviru omejitev, npr. ista ukazna arhitektura, isti hipervizor),

- možnost začasne ustavitve izvajanja navideznega računalnika, pri čemer se nadaljevanje izvajanja lahko nadaljuje na isti točki po prenosu na drug računalnik, ne da bi bil potreben ponoven zagon operacijskega sistema v navideznem računalniku, kar v strežniških centrih bistveno olajša vzdrževanje fizičnih strežnikov in omogoča razporejanje bremen med fizičnimi strežniki (angl. Load Balancing),

- več vzporednih operacijskih sistemov na enem samem strežniku omogoča redukcijo števila fizičnih strežnikov, boljši izkoristek razpoložljivih računskih in pomnilniških virov obstoječih fizičnih strežnikov in s tem pomembne prihranke pri stroških nakupa strojne opreme, porabi elektrike, vzdrževanju,

  hipervizorji so po številu vrstic programske kode za vsaj dva velikostna razreda manjši kot jedra tradicionalnih operacijskih sistemov; manj programske kode, ki ima priviligiran dostop, vsaj teoretično pomeni večjo zanesljivost[1:472].

Poglavje 3 Virtual Computing Lab

3.1 Zgodovina

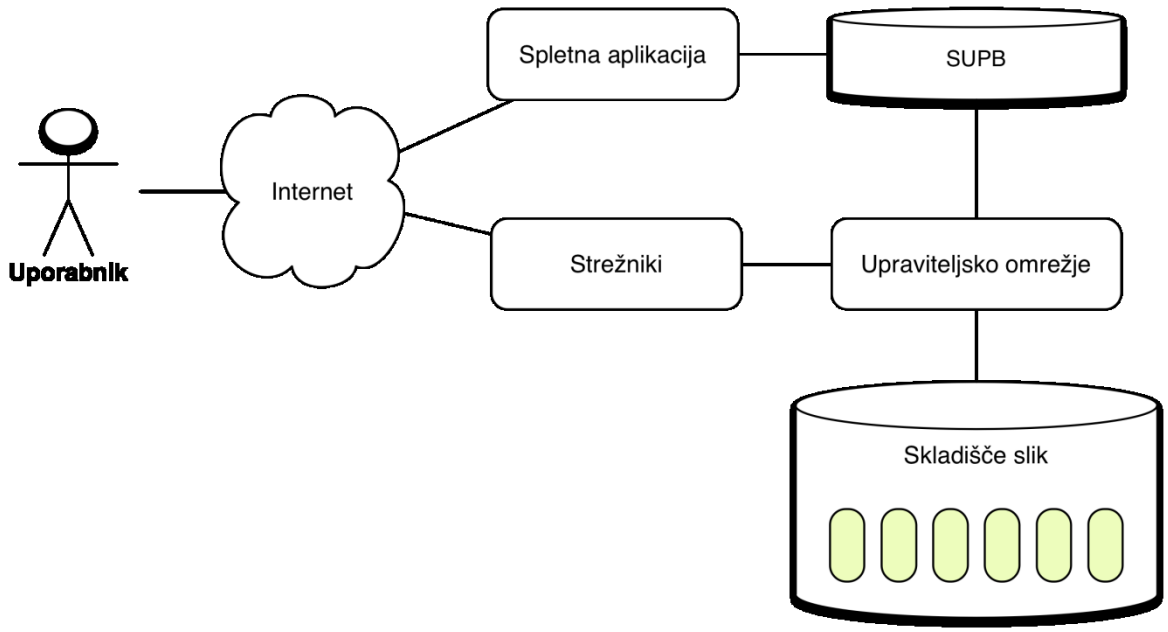
Programsko rešitev Virtual Computing Lab (VCL) so razvili na Državni univerzi Severne Karoline (NCSU) kot podporo za pedagoški proces. Težava, s katero so se soočili, je bila, da so imeli omejeno število laboratorijskih računalnikov in da so imeli preobremenjene posamične strežnike, do katerih so študentom nudili oddaljen terminalski dostop. Da bi tako študentom kot tudi raziskovalcem olajšali dostop do računskih virov, so leta 2004 pričeli z razvojem avtomatiziranega rezervacijskega sistema, ki bi uporabnikom na njihovo zahtevo iz univerzitetnega visokozmogljivostnega računskega centra dodeljeval računske vire. Sistem Virtual Computing Lab je z oddaljenim dostopom do rezerviranih računskih virov ob katerikoli uri in na katerikoli dan uspešno rešil problem omejenega dostopa do univerzitetnih računalniških laboratorijev. Projekt se je leta 2008 pridružil neprofitni fundaciji Apache Software Foundation, razvita programska koda pa je postala dostopna širši skupnosti [5].

3.2 Arhitekture sistema Virtual Computing Lab

- Glavne komponente sistema so:
- spletna aplikacija VCL,
- sistemski servis vcld,
- spletni strežnik Apache,
- sistem za upravljanje s podatkovnimi bazami MySQL,
- skladišče slik operacijskih sistemov,
- strojna oprema.

Spletna aplikacija VCL služi kot osrednja točka interakcije tako s končnim uporabnikom kot s skrbniki sistema. Napisana je v programskih jezikih PHP in JavaScript. Za transport dinamičnih spletnih strani, ki jih tvorijo skripte PHP, ji služi odprtokodni spletni strežnik Apache Web Server. Sistem za upravljanje s podatkovnimi bazami MySQL uporablja za

skladiščenje stanja sistema. Preko njega spletna aplikacija sporoča zahteve za rezervacije. Ločena komponenta vcld (vcl daemon), spisana v programskem jeziku Perl, deluje kot sistemska storitev in iz podatkovne baze periodično prebira ter procesira nove zahteve za rezervacije ter vodi podrobnosti rezervacije vse do trenutka, ko se uporabnik poveže na rezervirani vir. Skrbi za nalaganje slik operacijskih sistemov (angl. images) na fizične ali virtualne računalnike. Rezervacije so večinoma časovno omejene. Po poteku rezervacije sistemska storitev vcld sprosti rezervirane vire.



Slika 3: Komponente sistema VCL

3.3 Namestitev

Sistem VCL lahko deluje s širokim naborom platform - od računskih centrov s strežniškimi rezinami, do posamičnih namiznih računalnikov, delovnih postaj ali strežnikov [6]. Namestitev VCL-ja zahteva operacijski sistem GNU/Linux. Uradno sta podprti distribuciji Red Hat Enterprise Linux in CentOS. V primeru manjših gruč strežnikov je možno združiti spletno komponento, spletni strežnik, sistem za upravljanje s podatkovnimi bazami (SUPB), skladišče slik in sistemsko storitev vcld na enem samem fizičnem strežniku oz. strežniški rezini. Večje namestitve ponavadi ločijo spletno komponento ter SUPB in sistemsko storitev vcld namestijo na ločen strežnik, medtem ko se slike operacijskih sistemov skladiščijo na sistemih NAS/SAN. Strežnike, na katerih deluje sistemska storitev vcld, sistem imenuje upraviteljska vozlišča (angl. management nodes). Posamezno upraviteljsko vozlišče je

zadolženo za dodeljevanje samo enega tipa računalnikov, npr. samo fizičnih računalnikov ali pa samo virtualnih računalnikov na enem izmed podprtih hipervizorjev.

3.4 Spletna aplikacija sistema Virtual Computing Lab

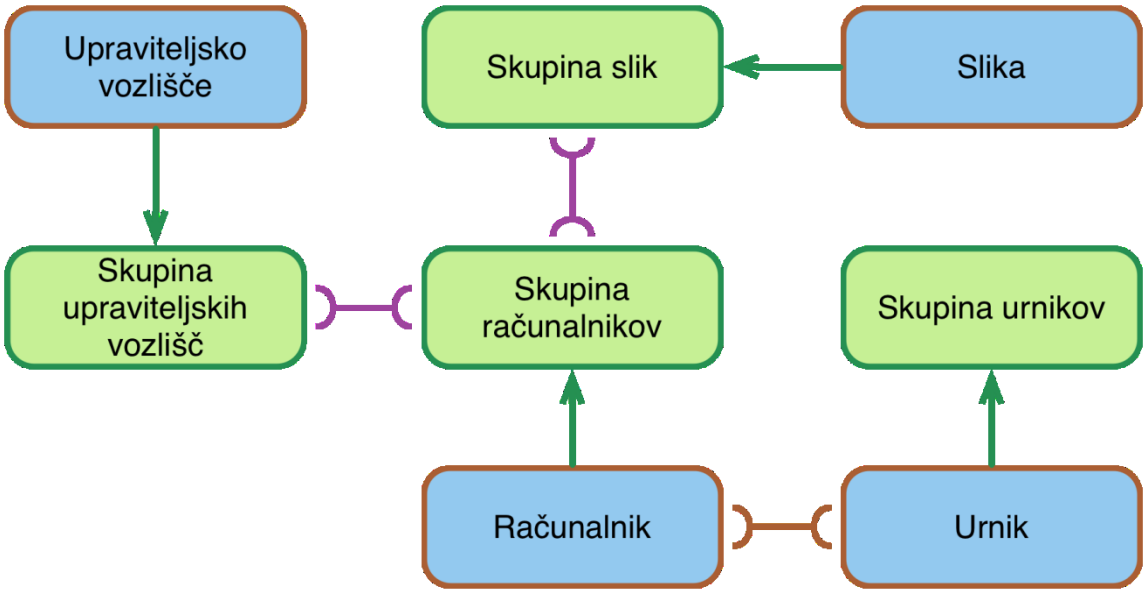
Osrednji namen spletne aplikacije VCL je, da na preprost način uporabnikom omogoči rezervacijo računalniških virov. Uporabnik preko spletnega vmesnika izbere željeno sliko operacijskega sistema. Spletna aplikacija preveri strojne zahteve izbrane slike, preveri razpoložljivost računalnikov, ki ustrezajo zahtevam in v primeru, da je rezervacija izvedljiva, zahtevo zapiše v podatkovno bazo. Sistemski servis vcld, ki ga opisujem v naslednjem podpoglavju, prevzame zahtevo in računalnik pripravi za oddaljeni dostop. Ob koncu priprav v podatkovno bazo zapiše ali je bil postopek uspešno zaključen. Spletna aplikacija z branjem stanj rezervacij zazna, da je računalnik pripravljen in uporabniku prikaže podrobne nastavitve za dostop na daljavo. Dostop do oddaljenih namizij rezerviranih računalnikov je možen preko protokolov RDP in VNC, medtem ko delo z oddaljeno uporabniško lupino poteka preko protokola SSH.

Rezervacije so časovno omejene in ponavadi trajajo 1 do 6 ur, medtem ko je uporabnikom z ustreznimi dodatnimi dovoljenji omogočena tudi t.i. strežniška rezervacija za daljši ali celo neomejen čas [6].

Spletna aplikacija hkrati služi kot centralni nadzorni vmesnik za upravljanje s sistemom. Skrbniki sistema preko spletne aplikacije upravljajo s 4 skupinami virov:

- upraviteljskimi vozlišči,
- računalniki,
- slikami operacijskih sistemov,
- urniki.

Skrbnik posamezne vire združuje v skupine in jih dodeljuje drugim skupinam virov, kot to prikazuje slika 4. Združevanje npr. omogoča, da se slikam s posebnimi strojnimi zahtevami priredi skupina računalnikov z zadostno računsko in pomnilniško kapaciteto [6].

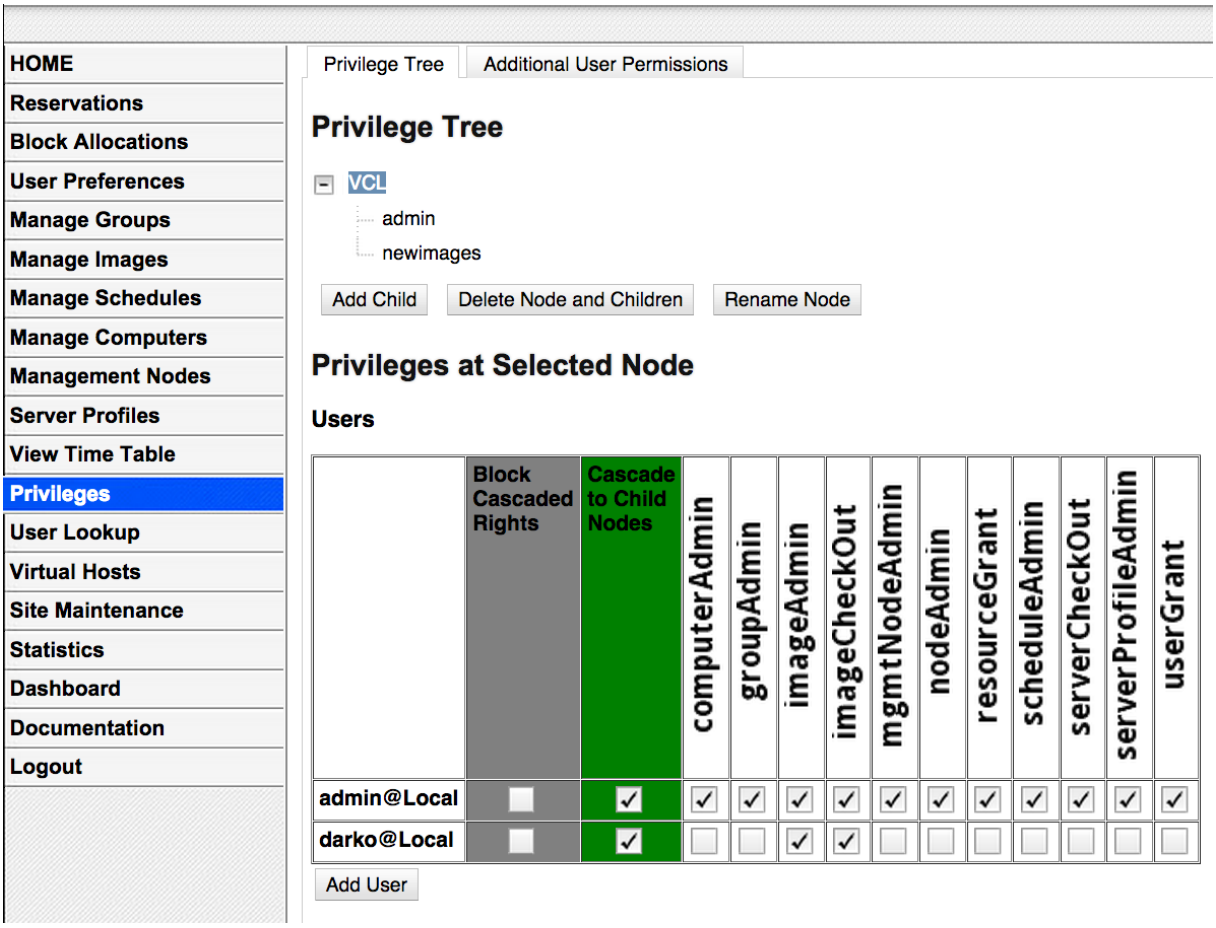


Slika 4: Združevanje virov VCL v skupine

Uporabniki, ki imajo status skrbnika slik, lahko dodajajo in odstranjujejo slike in s posebno rezervacijo za njihov zajem ustvarijo nove, spremenjene verzije obstoječih slik. Slike bodisi vsebujejo zgolj posnetek operacijskega sistema bodisi imajo nameščeno dodatno programsko opremo. V slednjem primeru sistem omogoča izposojo programske opreme na daljavo.

Računalnikom je mogoče dodeliti urnik, ki določa časovna obdobja, znotraj katerih so dovoljene rezervacije. Univerza NCSU tako zagotavlja, da strežniške se rezine njihovega podatkovnega centra uporabljajo za dva namena. Del časa so na voljo za pedagoške procese, preostanek časa pa so na voljo raziskovalcem za potrebe visokozmogljivostnega računalništva (angl. high performance computing) [6].

Pravice uporabnikov so urejene v drevesno strukturo. Posamezna vozlišča v grafu predstavljajo sklop uporabniških pravic, ki jih imajo uporabniki dodeljeni izbranemu vozlišču. Drevesna hierarhija omogoča dedovanje pravic iz očetovskih vozlišč oz. celotne hierarhije prednikov do korena drevesa. Spletno stran za urejanje pravic prikazuje slika 5.



Slika 5: VCL-jev uporabniški vmesnik za urejanje pravic uporabnikov

Omeniti velja še možnost t.i. blokovnih rezervacij (angl. block reservation), ki z izbrano sliko operacijskega sistema vnaprej pripravijo večje število računalnikov. Funkcija je koristna predvsem za hitro in učinkovito vnaprejšnjo pripravo računalniških učilnic za potrebe laboratorijskih vaj.

3.5 Sistemska storitev vcld

Sistemska storitev vcld s spletno aplikacijo komunicira izključno z zapisi v skupno podatkovno bazo. Skrbi za podrobnosti pri ustvarjanju novih virtualnih računalnikov, nalaganju slik in končnih nastavitvah na novo naloženih operacijskih sistemov.

Postopek, ki ga vcld izvrši med obdelavo rezervacij, je sledeč:

- najprej preveri morebitno časovno obravnavane rezervacije z drugimi rezervacijami,
- preveri odzivnost računalnika, ponavadi tako, da pošilja zahtevo ICMP ECHO,

- v primeru, da se rezervacija odziva, primerja ali se naloženi operacijski sistem ujema z zahtevano sliko,
- če je to potrebno, naloži nove slike operacijskega sistema,
- na oddaljenem računalniku spremeni nastavitve programskega strežnika SSH (sshd),
- na računalniku ustvari uporabniški račun,
- spremeni nastavitve požarnega zidu, tako da uporabniku omogoči oddaljen dostop.

V primeru, da uporabnik rezervira navidezni računalnik, tega vcld izbriše in s tem sprosti računske in pomnilniške vire.

Programska koda sistemske storitve vcld je objektno orientirana in zasnovana tako, da omogoča dodajanje novih razredov. Ti so v obliki modulov jezika Perl. Novi moduli npr. dodajo podoro za dodatne operacijske sisteme ali pa omogočajo komunikacijo z novimi hipervizorji. V poglavju 6 si bomo ogledali implementacijo oskrbovalnega modula, ki komunicira s programsko platformo OpenStack.

Poglavje 4 Računalništvo v oblaku

Ideja, da bi računske zmogljivosti postale javna dobrina, do katere bi uporabniki dostopali preko ločenega omrežja enostavno, tako kot uporabljajo elektriko preko električnega omrežja, se je pojavila že v 60. letih prejšnjega stoletja. Inštitut M.I.T. je skupaj z Bellovimi laboratoriji in podjetjem General Electric pričel delo na sistemu MULTICS (MULTiplexed Information and Computing Service). Njihova vizija je bila, da bi z računskimi viri enega samega superračunalnika oskrbovali celotno območje Bostona. Projekt je bil tako pred svojim časom, da ga Tanenbaum [1:13] primerja kar z računskim strojem Charlesa Babbagea. V nasprotju s slednjim je sicer sistem MULTICS ugledal luč sveta še za časa njegovih snovalcev, a širšega komercialnega uspeha ni doživel. Je pa pomembno prispeval k razvoju sodobnih operacijskih sistemov, še posebej sistema UNIX in njegovih odprtokodnih izpeljank GNU/Linux in FreeBSD. Potrebnega je bilo nadaljnega pol stoletja razvoja računalniških arhitektur, omrežij, operacijskih sistemov, teorij na področju porazdeljenega računalništva in prakse na področju izgradnje velikih računskih centrov, da je ideja ponovno zaživela v obliki računalniškega oblaka.

Leta 2003 je ekipa Amazonovih inženirjev pod vodstvom Chrisa Pinkhama in Christopherja Browna pričela z razvojem sistema, ki je bil javnosti prvič predstavljen leta 2006 pod imenom Amazon Elastic Compute Cloud (EC2)[8], [11]. Amazon je z EC2 del svojih velikanskih strežniških kapacitet ponudil v uporabo ljudem po svetu v obliki plačljive storitve. Uporabnik si preko spletnega ali pa programskega vmesnika v Amazonovih računskih centrih rezervira enega ali več navideznih strežnikov in plačuje glede na čas njihove uporabe, brez vnaprejšnjih stroškov nakupa in vzdrževanja strojne opreme. Dodaten prihranek pri taki uporabi predstavlja možnost samodejnega dodeljevanja in odvzemanja navideznih strežnikov. Uporabnikova virtualizirana infrastruktura raste in se krči glede na uporabnikove potrebe, na kar namiguje celo uporaba besede elastičnost v samem naslovu storitve.

Storitev EC2 se je izkazala za izjemno popularno, še posebej med razvijalci spletnih aplikacij in ponudniki spletnih storitev. Amazon je s storitvijo uvedel koncept infrastrukture kot storitve (angl. Infrastructure as a Service, Iaas) ter hkrati bistveno prispeval k popularizaciji računalništva v oblaku. Podoben koncept rezervacije računskih virov smo sicer videli že v poglavju o sistemu Virtual Computing Lab in univerza NCSU bi projekt najverjetneje poimenovala Cloud Computing Lab, če bi se ga lotila v letih po lansiranju EC2.

Uspehu Amazona so sledili mnogi drugi. Google je leta 2008 predstavil storitev Google App Engine, ki predstavlja alternativni model storitve v oblaku, t.i. platformo kot storitev (angl. Platform as a Service). Uporabnik se pri tem modelu odpove neposrednemu nadzoru nad virtualnim računalnikom in nima možnosti izbire operacijskega sistema, prav tako pa ga omejuje izbor programskega jezika in knjižnic. Infrastruktura ponudnika avtomatično dodeljuje računske in pomnilniške vire v uporabnikovi aplikaciji, pri čemer ima ta vtis, da njegovo aplikacijo izvaja računalnik z neomejenimi viri. Podobno rešitev je leta 2010 predstavil Microsoft v obliki storitve Windows Azure (kasneje Microsoft Azure). Leta 2008 so se pojavile prve odprtokodne rešitve, ki omogočajo vzpostavitev oblaka IaaS po vzoru Amazonovega EC2, med njimi sta bila prva projekt Eucalyptus in OpenNebula. Leta 2010 je sledil OpenStack, ki pa si ga bomo ogledali v naslednjem poglavju.

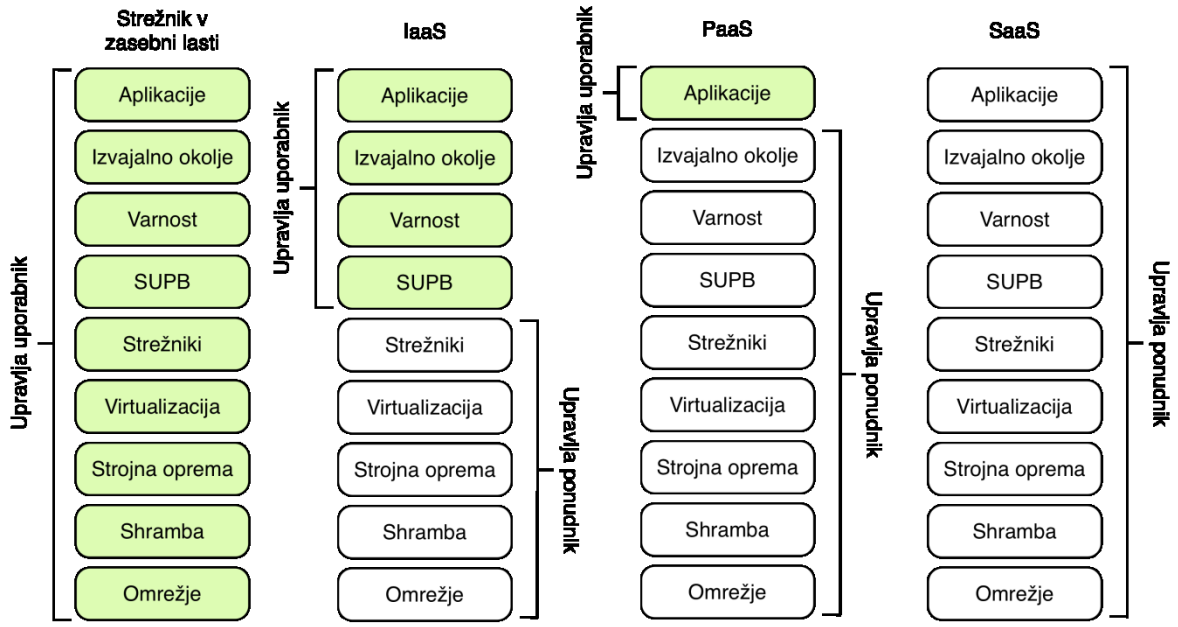
Nenadna popularizacija storitev v oblaku je zahtevala formalnejšo definicijo, kaj računalništvo v oblaku sploh je. Vaquero [10] npr. citira kar 22 različnih definicij, objavljenih v letu 2008. Leta 2011 je, v izogib dvoumnostim, ameriški National Institute of Standards and Technology (NIST) računalništvo v oblaku definiral kot model računalništva s sledečimi lastnostmi in funkcionalnostjo[7]:

- uporabnik si lahko sam dodeljuje računske vire, ne da bi posredovalo osebje ponudnika storitev,
- storitve so preko mrežnega dostopa dosegljive širokemu spektru naprav: od namiznih in prenosnih računalnikov, do mobilnih telefonov in tabličnih računalnikov,
- oblak združuje vire na enem mestu (angl. resource pooling). Do posameznega vira dostopa več uporabnikov (angl. multitenancy). Fizični in navidezni računski viri se dinamično razporejajo glede na obremenitve, pri čemer uporabnik nima natančnega nadzora nad lokacijo dodeljenega vira. Uporabnik ne more izbrati specifičnega fizičnega strežnika, lahko pa njegov ponudnik omogoči določi okvirno lokacijo strežnika nekje v posamezni državi ali na računskem centru,
- prožnost storitve (angl. elasticity) pomeni, da se viri lahko tudi samodejno dodeljujejo in sproščajo in sicer glede na uporabnika, ki jih potrebuje za povečanje svojih zmogljivosti. Uporabnik dobi vtis, da razpolaga z neomejenimi viri, ki si jih lahko ob kateremkoli času dodeli v poljubnih količinah,
- storitev se obračunava glede na porabo posameznega vira, npr. glede na kapaciteto dodeljenih diskovih kapacitet, računsko zmogljivost, hitrost omrežja, število uporabniških računov.

- NIST nadalje definira štiri modele storitev v oblaku:
- **programska oprema kot storitev** (angl. Software as a Service, SaaS) - uporabnik dostopa do končnih aplikacij, ki jih ponudnik gosti na svoji infrastrukturi. Dostopa lahko preko spletnega vmesnika ali pa namenskega programskega vmesnika. Uporabnik nima nadzora nad operacijskim sistemom, znotraj katerega se izvaja aplikacija, prav tako ne more upravljati z drugimi viri,
- S **platformo kot storitvijo** (angl. Platform as a Service, PaaS) uporabnik na ponudnikovi infrastrukturi poganja lastno programsko opremo, ki jo je razvil z uporabo programskih jezikov, knjižnic, storitev in orodij, ki jih podpira ponudnik. Podobno kot v primeru storitve SaaS, uporabnik ne nadzira operacijskega sistema, ima pa seveda nadzor nad lastno aplikacijo in potencialno tudi nad nastavitvami okolja, znotraj katerega se aplikacija izvaja,
- Z **infrastrukturo kot storitvijo** (angl. Software as a Service, IaaS) si uporabnik lahko dodeljuje procesorske vire, diskovne kapacitete, omrežne in druge vire, na katerih lahko namesti poljubne operacijske sisteme in njihovo poljubno programsko opremo. Uporabnik nima neposrednega nadzora nad fizično infrastrukturo, lahko pa mu ponudnik omogoči omejen nadzor nad mrežnimi nastavitvami in požarnim zidom.

NIST računalniške oblake klasificira še glede na model namestitve:

- zasebni oblak (angl. private cloud) je v lasti posamičnih organizacij,
- oblak, ki pripada skupnosti organizacij (angl. community cloud),
- javni oblak (angl. public cloud) omogoča dostop do računskih virov širši javnosti,
- hibridni oblak je kombinacija dveh ali več modelov navedenih zgoraj.



Slika 6: Primerjava modelov storitev računalništva v oblaku

4.1 OpenStack

OpenStack je odprtokodna programska platforma, ki omogoča vzpostavitev zasebnega oblaka tipa IaaS. Projekt je plod sodelovanja podjetja Rackspace in ameriške vesoljske agencije NASA. Sestoji iz različnih programskih komponent, ki delujejo kot samostojne storitve. Vsaka opravlja točno določeno vlogo in s svojo funkcionalnostjo podpira druge. Platforma se zgleduje po obstoječih odprtokodnih programskih rešitvah: npr. po sistemih za upravljanje s podatkovnimi bazami MySQL, MariaDB in PostgreSQL, po sporočilnih vrstah RabbitMQ, Qpid in 0mq. Podobno ima strukturirano knjižnico za upravljanje s hipervizorji libvirt in hipervizor KVM, ki je že vsebovan v jedru operacijskega sistema GNU/Linux. OpenStackove komponente so kot programsko lepilo, ki te samostojne odprtokodne rešitve poveže in so uporabniku preko spletnega in programskega vmesnika prikazane kot smiselna, zaključena celota. Platforma poleg dodeljevanja računskih virov - tega smo ga spoznali že v prejšnjih poglavjih- nudi še rezervacijo diskovnih kapacitet (storitev OpenStack Block Storage, imenovana tudi Cider), porazdeljeno skladišče objektov (OpenStack Object Storage s kodnim imenom Swift), avtentikacijo uporabnikov (OpenStack Identity, Keystone), virtualizacijo mrežnih virov (OpenStack Networking ali Neutron), nadzor nad porabo oz. zaračunavanje storitev (OpenStack Metering, Ceilometer), nadzorno ploščo v obliki spletne aplikacije (OpenStack Dashboard - Horizon) in storitev za t.i. orkestracijo oblaka (OpenStack Orchestration). Projekt naglo širi funkcionalnost platforme v izjemno agresivnih polletnih razvojnih ciklih, tako da dodaja nove komponente obstoječi infrastrukturi. Novejša storitev je

npr. OpenStack Database as a Service (Trove), ki po vzoru storitve Amazon Relational Database Service nudi SUPB, za katerega skrbi ponudnik storitev. Zadnja različica v času pisanja diplomskega dela, OpenStack 2014.2 (kodno ime Juno), je kot storitev dodala še možnost vzpostavitve porazdeljenega sistema procesiranja Hadoop v obliki komponente OpenStack Data Processing (Savana) [14].

Ob OpenStacku je zrasla zelo široka odprtokodna skupnost. NASA je iz projekta sicer izstopila leta 2013, so se pa projektu pridružili velikani računalniške industrije kot so IBM, Intel, HP, Cisco, telekomunikacijski ponudnik AT&T idr. [16] Močna podpora industrije tako OpenStacku zagotavlja dolgoročno perspektivo, kar je prednost v primerjavi z drugimi odprtokodnimi alternativami kot so Eucalyptus, CloudStack in OpenNebula [12].

V svojem diplomskem delu sem se osredotočil na manjšo podmnožico OpenStackovih komponent, ki zagotavlja osnovno upravljanje z računskimi viri in ki je funkcionalno najbližje obstoječemu modelu Virtual Computing Laba. Na osnovi teh komponent sem vzpostavil na treh fizičnih strežnikih miniaturen zasebni oblak IaaS. Na tem mestu velja omeniti, da o pravem zasebnem računalniškem oblaku lahko govorimo šele pri številu strežnikov, ki se giblje od 100 do 10,000 [12:65-76]. Šele pri takšnih številih je možno zagotoviti osnovne lastnosti računalniškega oblaka, kot jih predvideva NIST-ova definicija o prožnosti storitev. V primeru javnega oblaka so števila še za velikostni razred ali celo dva višja. Marinescu [15:77] tako navaja podatek, da je leta 2012 Amazon s svojo storitevijo EC2 razpolagal z več kot 450,000 fizičnimi strežniki (EC2 je sicer samo ena od mnogih Amazonovih storitev v oblaku pod skupnim imenom Amazon Web Services, tako da je število najverjetneje še bistveno večje), medtem ko naj bi se ocenjeno število Googlovih fizičnih strežnikov v letu 2013 gibalo pri vratolomnih 2,4 milijona. Zaradi neizmernih začetnih stroškov vzpostavitve globalne mreže med seboj povezanih računskih centrov, kot jih zahteva javni oblak IaaS, najverjetneje ne gre pričakovati večjih novincev na tem področju. Podjetja, kot so Google, Amazon in Microsoft so z dolgoletnim vlaganjem v infrastrukturo za lastne potrebe, ustrezne zmogljivosti že imela. Virtualizacija je nato odigrala ključno vlogo k vzponu javnega oblaka. Z njeno pomočjo so podjetja začela sproščati obstoječe vire svojih računskih centrov in jih ponujati zunanjim uporabnikom v zakup. Z virtualizacijo je možno en sam fizični strežnik razdeliti med več zunanjih uporabnikov. Brez prihrankov, ki jih prinaša, javni oblak tipa IaaS morda nikoli ne bi zaživel.

4.1.1 Arhitektura

Opis, ki sledi, povzema vire [12:65-76], [13], [17], [19], [20].

Komponente, ki smo jih izbrali za potrebe našega "oblaka" so:

- **OpenStack Compute** s kodnim imenom Nova. Je osrednja komponenta platforme. Upravlja s hipervizorji in na zahtevo uporabnika dodeljuje in odstranjuje virtualne računalnike, spreminja njihovo število navideznih jeder in količino dodeljenega pomnilnika, jih začasno ustavlja, ponovno zaganja idr. Seznam razpoložljivih operacij z virtualnimi računalniki je odvisen od izbire hipervizorja [13]. OpenStackova skupnost se najpogosteje odloča za hipervizor KVM, po popularnosti sledijo še Xen, LXC, VMwarov ESX in ESXi ter Hyper-V. OpenStackova storitev Nova je osnovana na NASINEM projektu Nebula in je bila poleg storitve Swift edina komponenta prve različice OpenStacka,

- **OpenStack Image Service** (Glance) - oskrbuje storitev OpenStack Compute s slikami operacijskih sistemov. Poleg tega, da dodaja slike v zbirko slik ali jih odstranjuje, nudi možnost zajemanja slik in ustvarjanja varnostnih kopij. Podprti so formati diskov QCOW2 (uporablja ga naveza QEMU in KVM), VHD (VMware, Xen, Hyper-V), AMI (Amazon Machine Image) in drugi.,

- **OpenStack Identity** (Keystone) - skrbi za avtentikacijo uporabnikov. Uporabnikove podatke bodisi skladišči v svoji podatkovni bazi bodisi se povezuje z zunanjimi imeniku uporabnikov preko protokolov oz. mehanizmov PAM, LDAP, OAuth. Uporabniku, ki se predstavi s pravilnimi podatki, dodeli časovno omejen varnostni žeton (angl. security token), ki ga uporabnik nato uporablja pri komunikaciji z ostalimi OpenStackovimi storitvami. Storitev OpenStack Identity hkrati opravlja funkcijo kataloga t.i. vstopnih točk programskega vmesnika (angl. API endpoints),

- **OpenStack Networking** (Neutron) - uporabnikom omogoča, da ustvarjajo navidezna omrežja, podomrežja, usmerjevalnike (hm, potrebno pred vsakim navesti navidezna?), na katera se priklapljajo navidezni računalniki. Definira t.i. vrata (angl. port), ki predstavljajo stično točko med navideznimi omrežji in računalniki. Vrata je moč povezati s t.i. plavajočimi IP naslovi (angl. floating IP addresses), ki uporabnikom omogočajo dostop preko javnega omrežja do svojih rezerviranih virtualnih računalnikov, katerih zasebni IP naslovi so sicer z zunanjih, javnih omrežij nedostopni,

- **OpenStack Dashboard** (Horizon) - spletna aplikacija, ki uporabnikom in skrbnikom sistema nudi grafični vmesnik za upravljanje z računalniškim oblakom. Komunicira z ostalimi OpenStackovimi storitvami in omogoča vse osnovne operacije, kot je npr. tvorba novih navideznih računalnikov, upravljanje z navideznimi omrežji, dodajanje novih uporabnikov in upravljanje z njihovimi pravicami.

Vse OpenStackove komponente svoje storitve odjemalcem nudijo v obliki programskih vmesnikov (angl. API), ki se podrejajo t.i. arhitekturnim omejitvam REST (angl. representational state transfer). Odjemalci OpenStackovih storitev so tako končni uporabniki - programerji oz. njihove aplikacije kot tudi OpenStackove lastne storitve, ki medsebojno komunicirajo preko teh programskih vmesnikov. Do storitev dostopajo preko protokola HTTP (HyperText Transfer Protocol). Dostopne točke storitev, ki jih hrani OpenStack Identity, so v obliki naslovov URI (Uniform Resource Identifier), ki jih odjemalci navedejo v sporočilih protokola HTTP, skupaj z metodami kot so GET, POST, PUT in DELETE. URI naslov hkrati določi, na kateri vir se nanaša navedena metoda. Telo sporočil HTTP, ki si jih izmenjavajo odjemalci in OpenStackove storitve, uporablja obliko zapisa JSON (JavaScript Object Notation).

Skrbniki sistema definirajo t.i. okuse (angl. flavors), konfiguracije virtualnih računalnikov, ki določajo največjo razpoložljivo količino virov, ki so na razpolago posameznemu navideznemu računalniku. Privzeto je definiranih več provizoričnih konfiguracij. Najmanjša, m1.tiny, virtualne računalnike denimo omeji na porabo največ 512 MiB glavnega pomnilnika, velikost navideznega trdega diska je 1 GB, omogočen je hkraten dostop zgolj do enega navideznega procesorskega jedra. Virtualni računalniki s konfiguracijo m1.xlarge medtem dostopajo do 16 GiB glavnega pomnilnika, imajo navidezni trdi disk velikosti 160 GB in so omejeni na uporabo navideznih 8 procesorskih jeder. Kot zanimivost naj omenimo, da OpenStackova dokumentacija kot začetno nastavitev razmerja med procesorskimi jedri fizičnih strežnikov in virtualnimi jedri navideznih računalnikov priporoča razmerje 1:16 [18]. En sam fizični strežnik z 8 procesorskimi jedri tako privzeto gosti kar 128 navideznih računalnikov.

Uporabniki so združeni v skupine oz. organizacijske enote, imenovane projekti ali stanovalci (angl. tenants). Uporabnik ob rezervaciji novega navideznega računalnika bodisi preko spletnega vmesnika storitve OpenStack Dashboard bodisi preko ukazne vrstice z odjemalcem imenovanim nova ali pa kar neposredno zahtevo preko protokola HTTP navede zgolj želeno sliko in okus. Od vseh odrugih podrobnosti, ki so potrebne za tvorbo novega navideznega računalnika in ki se razlikujejo od hipervizorja do hipervizorja, je tako povsem izoliran. Tovrstna abstrakcija upravljanja z računskimi viri je ena izmed poglavitnih prednosti računalniškega oblaka z modelom storitve IaaS. Žal operaterju, ki upravlja z računalniškim oblakom, konfiguracijske podrobnosti niso prizanešene. Nekatere izmed teh si bomo ogledali v naslednjem poglavju.

Poglavje 5 Načrt integracije

Za integracijo sistema Virtual Computing Lab z oblačno platformo OpenStack so bili predvideni naslednji koraki:

- pregled literature na temo računalništva v oblaku in dokumentacije OpenStacka,
- identifikacija strojnih zahtev za namestitev OpenStacka,
- pregled razpoložljive strojne in mrežne opreme in izposoja ali nakup dodatne opreme,
- konfiguracija pridobljene strojne opreme in vzpostavitev računalniškega omrežja,
- izbira in namestitev enega izmed operacijskih sistemov, ki jih podpira OpenStack,
- namestitev in konfiguracija OpenStacka,
- preučitev upravljanja z OpenStackom s stališča sistemskega skrbnika in običajnega uporabnika,
- podrobna preučitev programskih vmesnikov, ki jih nudi OpenStack in ki bi omogočili avtomatizacijo postopkov, ki jih sicer ročno vršita sistemski skrbnik in običajni uporabnik,
- pregled obstoječe dokumentacije sistema Virtual Computing Lab,
- identifikacija strojnih in programskih zahtev sistema VCL in njegova namestitev,
- učenje programskega jezika Perl, v katerem je spisana sistemska storitev vcld,
- pregled obstoječih poizkusov integracije obeh sistemov,
- razvoj razširitve sistemske storitve vcld z modulom za oskrbovanje z navideznimi računalniki iz vzpostavljenega zasebnega računalniškega oblaka.

Poglavje 6 Implementacija

6.1 Strojna oprema

Osnovna namestitev OpenStacka, ki vključuje mrežno storitev OpenStack Networking, zahteva tri fizične strežnike. Za vzpostavitev razvojnega okolja sem si izposodil strežnike starejšega datuma, ki so jih iz uporabe umaknili ravno zaradi optimizacije stroškov. Njihovi lastniki so zaradi cenejšega procerskega časa selili računske vire v Amazonov računalniški oblak EC2. Na voljo so bili strežniki:

- Tyan Transport GX28 B2881 (G28U4H) z dvema enojedrnima procesorjema Opteron 248 in 8 GiB pomnilnika
- Tyan Transport GT24 B2891 (B2891G24U4H) z dvema dvojedrnima procesorjema Opteron 265 in 8 GiB pomnilnika
- Supermicro A+Server MNL-1012C-MRF z enim 6-jedrnim procesorjem Opteron 4226 in 16 GiB pomnilnika

6.2 Računalniško omrežje

OpenStackova dokumentacija tudi v primeru najosnovnejše namestitve predvideva tri ločena fizična omrežja. Prvo je javno omrežje, na katerega je priklopljen strežnik z nameščenimi komponentami storitve OpenStack Networking, kjer do OpenStackovih storitev dostopajo končni uporabniki. Drugo je zasebno omrežje, ki izmenjava sporočila med OpenStackovimi storitvami in dostopu sistemskih skrbnikov. Na tretje fizično omrežje za t.i. tuneliranje (angl. tunneling) pa so priklopljeni strežniki, ki gostijo navidezne računalnike in strežnik s komponentami storitve OpenStack Networking. Za navidezne računalnike v posamičnih projektih ni nobenega zagotovila, da bodo dodeljeni istemu fizičnemu strežniku. Komunikacija med njimi poteka preko omenjenega tretjega omrežja za tuneliranje. Da bi dosegli popolno izolacijo med posameznimi projekti, storitev OpenStack Networking predvideva razdelitev omrežja bodisi s pomočjo tehnologije navideznih lokalnih omrežij (angl. VLAN) bodisi tako, da vzpostavi tuneliranje s pomočjo protokola GRE (angl. Generic Routing Encapsulation). Izbral sem slednjo možnost in zaradi premajhnega števila mrežnih

vmesnikov na strežnikih (vsak samo z dvema mrežnima karticama) združil zasebno omrežje z omrežjem za tuneliranje

6.3 Namestitev storitev oblačne platforme OpenStack

Na vse strežnike sem sem namestil distribucijo GNU/Linuxa Ubuntu 14.04 (LTS). Tyan Transport GX28 je odigral vlogo t.i. upraviteljskega vozlišča. Nanj sem namestil storitve OpenStack Identity, Compute, Networking (komponenti Neutron Server in vtič ML2), Image, Compute in Dashboard. Na strežnik sem namestil še sistem z upravljanje s podatkovnimi bazami MySQL in storitev za izmenjavo sporočil RabbitMQ. Strežnik Tyan Transport GT24 je prevzel agente storitve OpenStack Networking (Layer 2 Agent, Layer 3 Agent, DHCP Agent).

Supermicrov strežnik je bil edini s strojno podporo virtualizaciji, zato mu je pripadla vloga t.i. računskega vozlišča (angl. compute node), na katerem gostujejo navidezni računalniki. Za hipervizor sem izbral KVM.

Namestitev sem podrobno popisal in je na voljo na mojem repozitoriju GitHub na naslovu https://github.com/b0le/vclopenstack.

6.4 Namestitev Virtual Computing Laba

Sistem VCL sem namestil znotraj navideznega računalnika, ustvarjenega s Parallels Desktop, hipervizorjem tipa 2 za operacijski sistem OS X. Gostiteljski operacijski sistem je imel dodeljen naslov IP v "javnem" omrežju, znotraj katerega je bil dodeljen naslov tudi navideznemu računalniku. Virtual Computing Lab je tako smel komunicirati zgolj z javno dostopnimi OpenStackovimi storitvami. Za namestitev in konfiguracijo VCL-ja sem uporabil navodila za namestitev različice 2.3.2 [21].

6.5 Implementacija modula za oskrbovanje openstack.pm

Z razvojem modula openstack.pm se je pred menoj ukvarjal že Young-Hyun Oh z univerze NCSU in pri tem naletel na nekatere nepremostljive ovire [24]. Sistem VCL npr. zahteva vnaprejšen vnos podatkov o računalnikih, ki bodo na voljo za rezervacijo, med drugim imena (DNS) in statičnega naslova IP, tudi za navidezne računalnike, ki sploh še ne obstajajo. VCL zapovrh predvideva statično prevajanje definiranih imen računalnikov preko datoteke /etc/hosts. Takšen pristop je v nasprotju z dinamično naravo dodeljevanja naslovov IP, kot ga

predvideva OpenStack. Šele popravki NCSU-jevega razvijalca Andyja Kurtha so zmanjšali, ne pa tudi povsem razrešili tako odvisnost od datoteke /etc/hosts [26].

Dodaten zaplet je predstavljala zelo pomankljiva VCL-jeva dokumentacija. Opis javnega vmesnika razreda Provisioning, iz katerega dedujejo vsi razredi, ki se oskrbujejo z računskimi viri, podrobneje opiše samo metodo capture, ki je klicana med posebnim tipom rezervacij, namenjenih zajemu novih slik, ne opiše pa za delovanje modula za oskrbovanje najbolj pomembne metode load [25].

Young-Hyun Oh je sicer uspel razviti delujoč prototip modula openstack.pm, ki je deloval z OpenStackovo različico 2012.1 (Essex) in kasneje 2012.2 (Folsom), vendar z razvojem nato ni sledil razvoju OpenStacka. Njegova namestitev OpenStacka tako npr. predvideva uporabo komponente nova-network, ki jo OpenStackova skupnost med prehodom na sodobnejšo storitev OpenStack Networking počasi opušča in je bila sprva celo planirana za odstranitev v različici 2013.2 (Havana) [22].

V tem diplomskem delu sem se osredotočil predvsem na izboljšave obstoječega modula, testiranje ob novejši različici OpenStacka 2014.1 (Icehouse) in na razširitev modula s podporo za storitev OpenStack Networking.

Sistemska storitev vcld je spisana v Perlu, vendar za ta jezik ne obstaja ne obstaja uradno podprta knjižnica za komunikacijo z OpenStackovimi storitvami, zato sem moral preučiti OpenStackove programske vmesnike [23] in s storitvami OpenStacka komunicirati neposredno v obliki sporočil HTTP s pomočjo modula LWP::UserAgent [28], ki ga je v svoji programski kodi uporabljal tudi Young-Hyun Oh.

Storitev vcld ob novi rezervaciji, ki jo zazna s periodičnim povpraševanjem skupne podatkovne baze, ustvari objekt razreda Module::State::new. Ta najprej preveri morebitna časovna prekrivanja obravnavane rezervacije z drugimi rezervacijami. Z zahtevo ICMP ECHO preveri razpoložljivost računalnika, na katerega se nanaša rezervacija. V našem primeru virtualni računalnik v resnici nikoli ne obstaja, ker ga ob koncu rezervacije izbrišemo. Objekt razreda Module::State::new zato vsakič kliče metodo Module::Provisioning::openstack::load. Znotraj te metode poglavitno vlogo odigra Ohjeva obstoječa metoda _create_os_instance, ki komunicira s storitvijo OpenStack Compute in je tista, ki v računalniškem oblaku ustvari nove navidezne računalnike. V Ohjevi metodi sem preverjanje, ali je računalnik že v delujočem stanju, ki ga je izvajal z zahtevami ICMP ECHO, zamenjal s klici na novo implementirane metode _get_os_instance_status. Ob uporabi storitve OpenStack Networking in konfiguraciji, kot jo predvidevajo navodila za osnovno namestitev, so namreč navideznim računalnikom samodejno dodeljeni naslovi iz povsem izoliranega

zasebnega naslovnega prostora. Glede na našo namestitev VCL-ja, ki je lahko do OpenStackovih dostopala samo preko javnega omrežja, Ohjeve ICMP ECHO zahteve nikoli niso dosegle virtualnega računalnika.

Po uspešnem klicu metode _os_create_instance sem znotraj metode load dodal klic na novo implementirane funkcije _create_os_floating_ip. Ta komunicira s storitvijo OpenStack Networking in ustvari nov t.i. plavajoči naslov IP (angl. floating IP address), ki je dodeljen znotraj javnega omrežja. Plavajoči naslovi omogočijo dostop do zasebnih naslovov navideznih računalnikov preko javnega omrežja. Metoda _create_os_floating_ip hkrati plavajoči naslov IP dodeli na novo ustvarjenemu virtualnemu računalniku. Preko plavajočih naslovov IP se uporabnik v končni fazi poveže z navideznim računalnikom. Nabor metod sem razširil še s podporno funkcijo _get_os_floating_ip_ids, ki preveri ali je virtualnemu računalniku že dodeljen plavajoči naslov in pa metodo _delete_os_floating_ip, ki plavajoče naslove IP odstranjuje. Primer kode, ki demonstrira komunikacijo s storitvijo OpenStack Networking:

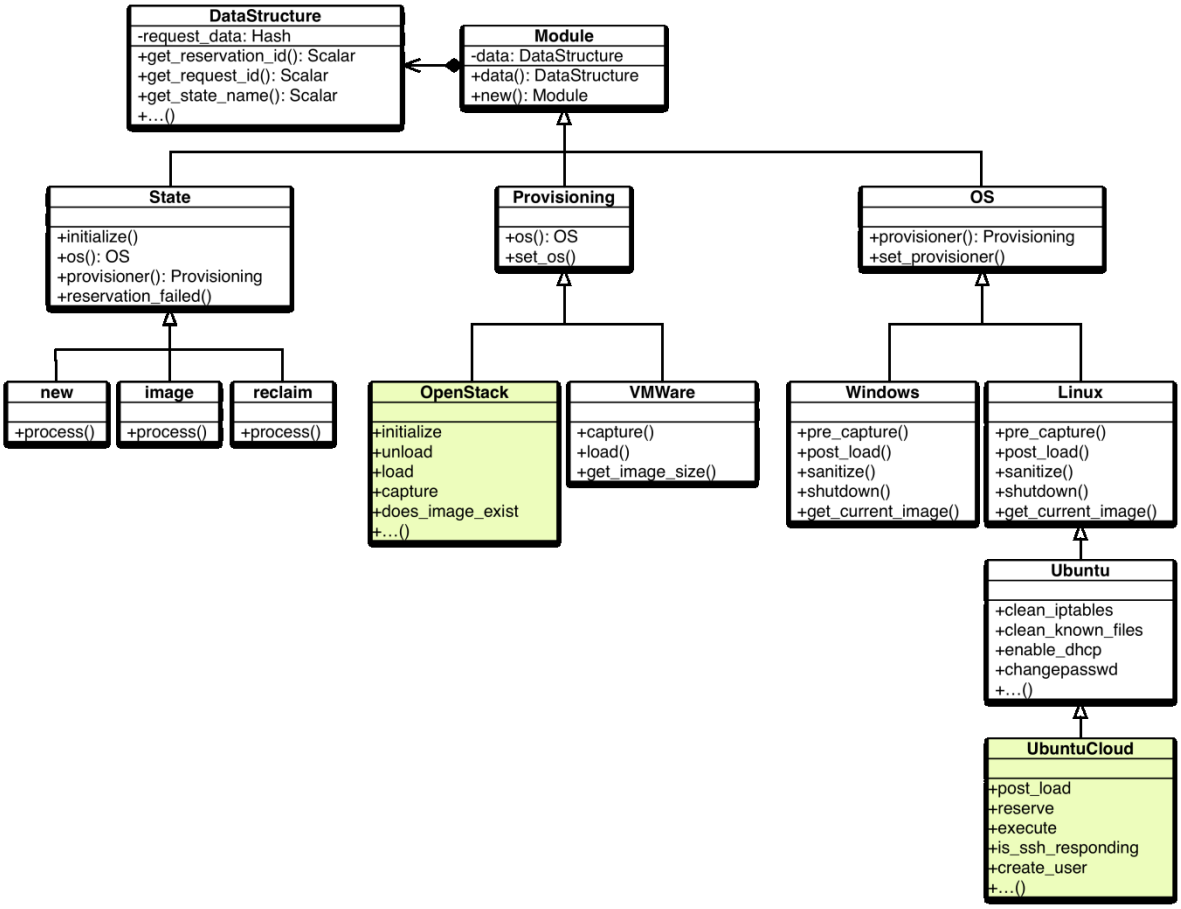
```
# pridobimo varnostni žeton
my $os_token = $self->_get_os_auth_token();
# pridobimo vstopno točko storitve OpenStack Network
my $os_networking_url = $self->_get_os_endpoint_network();

# kreiramo nov objekt razreda LWP::UserAgent
my $http_ua = new LWP::UserAgent;
my $resp = $self->{http_ua}->get(
      $os_networking_url . "/v2.0/floatingips?port_id=" . $os_port_id,
      x_auth_token => $os_token
);
unless ($resp->is_success) {
      notify($ERRORS{'WARNING'}, 0, "failed to get instance port data: " .
join("\n", $resp->content));
      return;
}
# odziv OpenStack Networking komponente je v obliki zapisa JSON,
# tu je uporabljena funkcija from_json modula JSON
my $output = from_json($resp->content);
unless (defined($output)) {
      notify($ERRORS{'WARNING'}, 0, "failed to parse json output");
      return;
}
# kreiramo nov seznam in nanj uvrstimo identifikacijske številke
# plavajočih naslovov IP
my @floating_ip_ids;
push @floating_ip_ids, $_->{id} for @{$output->{floatingips}};
return @floating_ip_ids;
```

Programska koda 1: Primer pridobitve seznama plavajočih naslovov IP

Nadalje sem Ohjevo metodo _get_os_compute_url_auth, ki je komunicirala s storitvijo OpenStack Identity, razdelil na dva dela, _set_os_auth_token in _get_os_auth_token. Metoda _set_os_auth_token je klicana ob inicializaciji objekta razreda Module::Provisioning::openstack. Storitvi OpenStack Identity se predstavi z uporabniškim imenom in geslom in v zameno prejme varnostni žeton. V primerjavi z izvirnikom, žeton skupaj z njegovim datumom veljave uskladiščim v dveh novih objektnih spremenljivkah, tako da ga je moč uporabiti večkrat in hkrati pred vsako njegovo uporabo preveriti ali je še veljaven. Preverjanje veljavnosti se zgodi v metodi _get_os_auth_token. Iz odgovora, ki ga pošlje OpenStack Identity, _set_os_auth_token hkrati razbere še naslove vstopnih točk vseh razpoložljivih OpenStackovih storitev. Ohjeva izvorna implementacija je predvidevala vnovično avtentikacijo ob vsaki zahtevi poslani OpenStackovim storitvam in je vračala zgolj vstopno točko programskega vmesnika storitve OpenStack Compute.

Da je modul naposled pričel delovati je bilo potrebno redefinirati še nekaj obstoječih metod. Kot podrazred razreda Module::OS::Linux:Ubuntu sem ustvaril nov razred UbuntuCloud, namenjen rabi s slikami Ubuntu Cloud, ki jih podjetje Canonical redno posodablja in so namenjene poganjanju v računalniških oblakih tipa IaaS. V novem razredu sem redefiniral metodo execute. Metoda med postopkom rezervacije preko protokola SSH izvaja ukaze na računalniku, ki je predmet obravnavane rezervacije. Uporabljajo jo številne druge metode razreda Module::OS in njegovih razredov za namene rekonfiguracije rezerviranih računalnikov. Kljub spremembam, omenjenim v [26], je metoda execute za povezovanje z rezerviranimi računalniki v času pisanja te diplome še vedno uporabljala imena računalnikov. Nova, redefinirana različica, se namesto tega na virtualne računalnike povezuje preko njihovih plavajočih naslovov IP. Popravkov je bila deležna še metoda update_public_ip_address. Ta v podatkovni bazi nadomesti vrednost, ki jo je uporabnik vnesel med konfiguriranjem računalnikov. Takrat pravi naslov IP namreč še ni znan. Prav tako sem preklical vse metode razreda Module::OS::Linux::Ubuntu, ki spreminjajo nastavitve požarnega zidu in nastavitve sistemske storitve sshd. Podjetje Canonical slike Ubuntu Cloud namreč že ustrezno pripravi za oddaljen dostop.



Slika 7: Hierarhija razredov sistemske storitve vcld

Poglavje 7 Rezultat

Rezultat tega diplomskega dela je delujoč prototip modula za VCL-jevo sistemsko storitev vcld, ki omogoča oskrbovanje z navideznimi računalniki iz oblačne platforme OpenStack. Modul še zdaleč ni brez težav, a so te deloma sistemske narave. Virtual Computing Lab namreč nekoliko nespretno obravnava skoraj povsem enako tako fizične laboratorijske računalnike s statično dodeljenimi imeni in naslovi kot tudi navidezne računalnike, ki se dinamično ustvarjajo na zahtevo. Dokumentacija sistema VCL je zelo pomanjkljiva in daje vtis, da so jo naknadno za silo spisali vzdrževalci sistema, nikakor pa ne izvorni razvijalci. Opis podatkovnega modela je v obliki linearnega seznama brez diagramov ER, prav tako tako manjkajo opisi atributov številnih entitet. Zunanjemu razvijalcu, ki bi želel prispevati k projektu, ne preostane drugega kot ročno sledenje programski kodi skorajda od prve do zadnje vrstice. Koda je zapovrh nenavadno strukturirana, z obsežnim centralnim modulom DataStructure.pm, v katerega se vedno znova ob vsakem zahtevku preslika dobršen del podatkovnega modela. Ta modul v nasprotju z vsemi ostalimi uporablja alternativen pristop k objektno orientiranemu programiranu z uporabo modula Object::InsideOut [27], od katerega se je Perlova skupnost pričela oddaljevati ob prihodu modula Moose. Razred, ki ga implementira modul DataStructure.pm vsebuje skoraj 500 metod za dostopanje do atributov entitet. Razvijalci ne uporabljajo nobenega avtomatiziranega testiranja, kaj šele konceptov sprotne integracije (angl. continuous integration), kot jih navaja Duvall[29]. V kombinaciji z 12 let staro različico Perla, ki jo uporablja programska koda, je projekt neprijazen do novincev. Ocenjujem, da je projekt zgolj še v fazi vzdrževanja z oslabljeno ekipo razvijalcev-vzdrževalcev in da nadaljni poizkusi v smeri uporabe VCL-ja kot komponente, ki bi omogočala časovno omejeno rezervacijo za potrebe podpore pedagoškega procesa, niso smiselni. Mojo tezo potrjuje osebje Laboratorija za računalniške komunikacije Fakultete za računalništvo in informatiko, ki je prenehalo z uporabo VLC-ja in na novo implementiralo prototip rezervacijskega sistema, ki uporablja oblačno platformo OpenStack.

Poglavje 8 Zaključek

V diplomskem delu smo predstavili osnovne principe virtualizacije, ki je pomembno pripomogla k vzponu računalništva v oblaku. Ogledali smo si različne modele storitev računalništva v oblaku in se pri tem osredotočili na t.i. infrastrukturo kot storitev, ki jo je popularizirala Amazonova storitev Elastic Compute Cloud. Predstavili smo dva sistema, ki uporabnikom omogočata rezervacijo in oddaljen dostop do računskih virov. Prvega, Virtual Computing Lab, so razvili na Državni univerzi Severne Karoline za podporo pedagoškemu procesu. Drugi, odprtokodna oblačna paltforma OpenStack, omogoča vzpostavitev zasebnega računalniškega oblaka in se funkcionalno zgleduje po Amazonovi storitvi. V naglih razvojnih ciklih njeni razvijalci širijo nabor storitev, vendar OpenStack zaenkrat ne omogoča časovno omejenih rezervacij računskih virov, kot to počne Virtual Computing Lab. V nadaljevanju smo predstavili poizkus integracije obeh sistemov in usposobili ter nadgradili obstoječ prototip modula za sistem VCL, ki omogoča oskrbovanje z računskimi viri v zasebnem oblaku, zgrajenem na platformi OpenStack. Zaključili smo, da zaradi VCL-jeve slabo strukturirane kode, spisane v stari različici programskega jezika Perl, ki v zadnjih letih izgublja na veljavi, razvoja v smeri integracije obeh sistemov ni smiselno nadaljevati. Laboratorij za računalniške komunikacije je v času nastajanja te naloge sistem VCL že opustil in razvil nov rezervacijski sistem na osnovi platforme OpenStacka. Slednja uživa izjemno podporo velikih podjetij računalniške in telekomunikacijske industrije ter širše odprtokodne skupnosti, kar ji zagotavlja dolgoročno perspektivo.

Literatura

[1] A. S. Tanenbaum in H. Bos, *Modern Operating Systems*, 4. izd., Upper Sadle River, New Jersey: Prentice Hall, 2014.

[2] K. Nance, B. Hay in M. Bishop, "Virtual Machine Introspection: Observation or Interference?", *IEEE Security & Privacy*, vol. 6, št. 5, str. 32-37, oktober 2008.

[3] E. Bugnion at al. (2012). Bringing Virtualization to the x86 Architecture with the Original VMware Workstation, *ACM Transactions on Computer Systems* [Online]. *30(4)*, str. 1-51. Dosegljivo: http://dl.acm.org/citation.cfm?id=2382553&CFID=590538884&CFTOKEN=98048700. [Dostopano 26. 10. 2014].

[4] VMware. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. (10. november 2007). *VMware* [Online]. Dosegljivo: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf [Dostopano: 26. 10. 2014].

[5] Virtual Computing Lab. VCL History. (30. junij 2008). *Virtual Computing Lab: powered by Apache VCL* [Online]. Dosegljivo: http://vcl.drupal.ncsu.edu/vcl-history. [Dostopano: 26. 10. 2014].

[6] A. Kurth in J. Thompson. Resources, Groups & Privileges. (21. januar 2014). *Confluence: Apache VCL* [Online]. Dosegljivo: https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=31819796. [Dostopano: 26. 10. 2014].

[7] P. Mell in T. Grance. The NIST Definition of Cloud Computing. (september 2011). *National Institute of Standards and Technology: Information Technology Laboratory* [Online]. Dosegljivo: http://www.nist.gov/itl/cloud/. [Dostopano: 26. 10. 2014].

[8] Amazon Web Services. Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta. (24. avgust 2004). *Amazon Web Services* [Online]. Dosegljivo: https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/. [Dostopano: 26. 10. 2014].

[9] Amazon Web Services. Regions and Availability Zones. (september 2014). *Amazon Elastic Compute Cloud: User Guide for Linux (API Version 2014-09-01)* [Online]. Dosegljivo: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html. [Dostopano: 26. 10. 2014].

[10] L. M. Vaquero et al. (2009). A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review* [Online]. *39(1)*, str. 50-55. Dosegljivo: http://ccr.sigcomm.org/online/files/p50-v39n1l-vaqueroA.pdf. [Dostopano: 27. 10. 2014].

[11] J. Clark. How Amazon exposed its guts: The History of AWS's EC2. (7. junij 2012). *ZDNet* [Online]. Dosegljivo: http://www.zdnet.com/how-amazon-exposed-its-guts-the-history-of-awss-ec2-3040155310/.[Dostopano: 27. 10. 2014].

[12] J. Rhoton, J. D. Clercq in F. Novak, *OpenStack Cloud Computing: Arhitecture Guide*, 3. izd., London: Recurssive Press, 2014.

[13] OpenStack Wiki. HypervisorSupportMatrix. (2014). *OpenStack Wiki* [Online]. Dosegljivo: https://wiki.openstack.org/wiki/HypervisorSupportMatrix#Known_bugs.2C_by_hypervisor. [Dostopano 28. 10. 2014].

[14] OpenStack Wiki. ReleaseNotes/Juno. (2014). *OpenStack Wiki* [Online]. Dosegljivo: https://wiki.openstack.org/wiki/ReleaseNotes/Juno. [Dostopano: 28. 10. 2014].

[15] D. Marinescu, *Cloud Computing: Theory and Practice*. Burlington, Massachusetts: Morgan Kaufmann, 2013.

[16] OpenStack Foundation. Companies Supporting The OpenStack Foundation. (2014) *OpenStack Foundation* [Online]. Dosegljivo: http://www.openstack.org/foundation/companies/. [Dostopano: 28. 10. 2014].

[17] OpenStack Foundation. OpenStack Operations Guide. (28. oktober 2014). *OpenStack Foundation* [Online]. Dosegljivo: http://docs.openstack.org/openstack-ops/content/. [Dostopano: 28. 10. 2014].

[18] OpecStack Foundation. OpenStack Operations Guide - Chapter 5. Scaling. (28. oktober 2014). *OpenStack Foundation* [Online]. Dosegljivo: http://docs.openstack.org/openstack-ops/content/. [Dostopano: 28. 10. 2014].

[19] OpenStack Foundation. OpenStack Cloud Administration Guide. (28. oktober 2014). *OpenStack Foundation* [Online]. Dosegljivo: http://docs.openstack.org/admin-guide-cloud/content/index.html. Dostopano: 28. 10. 2014].

[20] OpenStack Foundation. OpenStack Virtual Machine Image Guide. (28. oktober 2014). *OpenStack Foundation* [Online]. Dosegljivo: http://docs.openstack.org/image-guide/content/index.html. [Dostopano: 28. 10. 2014].

[21] The Apache Software Foundation. VCL 2.3.2 Installation Guide. (26. april 2013). *The Apache Software Foundation* [Online]. Dosegljivo: https://vcl.apache.org/docs/VCL232InstallGuide.html. [Dostopano: 29. 10. 2014].

[22] OpenStack Foundation. Depracation of Nova Network. (29. oktober 2014). *OpenStack Foundation* [Online]. Dosegljivo: http://docs.openstack.org/openstack-ops/content/nova-network-deprecation.html. [Dostopano: 29. 10. 2014].

[23] OpenStack Foundation. OpenStack API Complete Reference. (29. oktober 2014). *OpenStack Foundation* [Online]. Dosegljivo: http://developer.openstack.org/api-ref.html. [Dostopano: 29. 10. 2014].

[24] Young-Hyun Oh. VCL-590 OpenStack Module. (2012). *The Apache Software Foundation* [Online]. Dosegljivo: https://issues.apache.org/jira/browse/VCL-590. [Dostopano: 29. 10. 2014]

[25] A. Kurth in A. Peeler. Provisioning Engine Module Interface Specification. (8. oktober 2014). *Confluence: Apache VCL* [Online]. Dosegljivo:https://cwiki.apache.org/confluence/display/VCL/Provisioning+Engine+Module+Interface+Specification. [Dostopano: 26. 10. 2014].

[26] A. Kurth. Allow dynamic private IP addresses, remove /etc/hosts requirement. (2014). *The Apache Software Foundation* [Online]. Dosegljivo: https://issues.apache.org/jira/browse/VCL-767. [Dostopano: 29. 10. 2014].

[27] J. D. Hedden. Object::InsideOut. (2014). *CPAN Search*. Dosegljivo:http://search.cpan.org/~jdhedden/Object-InsideOut-3.98/lib/Object/InsideOut.pod. [Dostopano: 29. 10. 2014].

[28] Michael Schilli. LWP::UserAgent - Web user agent class. (2014). Dosegljivo: http://search.cpan.org/dist/libwww-perl/lib/LWP/UserAgent.pm. [Dostopano 29. 10. 2014].

[29] P. M. Duvall, S. Matyas in A. Glover. *Continuous Integration: Improving Software Quality and Reducing Risk*. Boston: Addison-Wesley Professional, 2007.