Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Igor Lalić

Ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00431/2013
Datum: 12.04.2013

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

Kandidat: **IGOR LALIĆ**

Naslov: **OGRODJE ZA PRIKAZ PORABE IN BELEŽENJE VIROV NAD PONUDNIKOM INFRASTRUKTURE TIPA IAAS**

A FRAMEWORK FOR RECORDING AND DISPLAY OF RESOURCES FOR IAAS PROVIDERS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS. Ogrodje mora omogočati interaktivno vizualizacijo podatkov ne glede na vrsto podatka ter podpirati kateregakoli ponudnika infrastrukture. Ogrodje mora omogočati izbiro časovnega intervala, metrik in strežnikov. Delovati mora v novejših brskalnikih, ki podpirajo spletne standarde.

Mentor:
izr. prof. dr.Patricio Bulić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Igor Lalić, z vpisno številko **63100077**, sem avtor diplomskega dela z naslovom:

Ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- 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 11. septembra 2013 Podpis avtorja:

Rad bi se zahvalil mentorju doktorju Patriciu Buliću, Alešu Černivcu in ostalim sodelavcem iz podjetja XLAB d.o.o., ki so mi pomagali pri izdelavi diplomske naloge. Zahvalil bi se tudi staršem za vso podporo med študijem.

Svoji dragi Zali.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Obstoječa literatura **3**

3 Opis ponudnikov **5**

3.1 OpenStack 5

4 Arhitektura rešitve **17**

5 Ovrednotenje arhitekture **27**

6 Možnosti nadaljnega dela **29**

7 Sklepne ugotovitve **31**

Seznam uporabljenih kratic

IaaS - ang. Infrastructure as a Service

API - ang. Aplication Programming Interface

EC2 - ang. Elastic Compute Cloud

LDAP - ang. Lightweight Directory Access Protocol

KVS - ang. key-value store

XCP - ang. Xen Cloud Platform

KVM - ang. Kernel-based Virtual Machine

QEMU - ang. Quick EMUlator

VNC - ang. Virtual Network Computing

IP - ang. Internet Protocol

S3 - ang. Simple Storage Service

AMPQ - ang. Advanced Message Queuing Protocol

SQL - ang. Structured Query Language

REST - ang. Representational state transfer

HTTP - ang. Hypertext Transfer Protocol

RADOS - ang. reliable autonomic distributed object store

L3 - ang. Layer 3

DHCP - ang. Dynamic Host Configuration Protocol

CPE - centralna procesna enota

CSS - ang. Cascading Style Sheets

SVG - ang. Scalable Vector Graphics

D3 - ang. Data-Driven Documents

DOM - ang. Document Object Model

JSON - ang. JavaScript Object Notation

Povzetek

Ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS vizualizira podatke o porabi virov iz baze. Ogrodje podpira tri tipe vizualizacij: kolobarni, črtni in ploščinski graf. Kolobarni graf prikazuje zadnje vrednosti metrik za vse strežnike vseh ponudnikov. Črtni graf prikazuje zgodovino porabe virov za eno metriko in en strežnik. Ploščinski graf prikazuje porabo virov ene metrike za vse strežnike določenega ponudnika infrastrukture. Ko uporabnik izbere časovno obdobje, metrike in strežnike, se naložijo podatki in se prikažejo manjši grafi. Ob kliku na manjši graf se pokaže večji graf, ki je interaktiven. Črtni in ploščinski graf imata pod glavnim grafom prikazan še nižji graf, ki je namenjen izbiri časovnega obdobja za določen del prikazanega grafa. Ob premiku miške preko črtnega grafa se prikaže čas in vrednost v tistem času. Ogrodje deluje v spletnih brskalnikih, ki podpirajo spletne standarde.

Ključne besede: poraba virov, IaaS, D3.js, vizualizacija, Contrail

Abstract

A framework for recording and display of resources for IaaS providers visualizes recorded data of resource usage. Framework supports three types of visualizations: donut, line and areal charts. Donut chart shows the last value of metrics for all servers of all providers. Line chart shows history of resource usage for one metric and one server. Areal chart shows resource usage of one metric and all servers for specific provider. When user selects a time period, metrics and servers, the data is loaded and small charts are shown. If small chart is clicked, big interactive chart is drawn. Line and areal charts have below main chart drawn another low chart, which is used to select time period of drawn chart. On mouse move over line chart, time and value is shown for that moment. Framework works in browsers that supports web standards.

Keywords: resource usage, IaaS, D3.js, visualization, Contrail

Poglavje 1

Uvod

V podjetju XLAB d.o.o. je potrebno za evropski projekt Contrail razviti ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS (ang. Infrastructure as a Service).

To ogrodje mora omogočati vizualizacijo podatkov ne glede na vrsto podatka in mora podpirati kateregakoli ponudnika infrastrukture. Ogrodje mora omogočati izbiro časovnega intervala, metrik in strežnikov. Delovati mora v novejših brskalnikih, ki podpirajo spletne standarde.

Prikazati mora tri različne grafe: kolobarni, črtni in ploščinski. Kolobarni graf prikazuje zadnje vrednosti za vse strežnike vseh ponudnikov. Črtni graf prikazuje zgodovino porabe virov za eno metriko in en strežnik. Ploščinski graf prikazuje porabo virov ene metrike za vse strežnike določenega ponudnika infrastrukture.

Ko se podatki naložijo, se prikažejo manjši grafi. Ob kliku na manjši graf se pokaže večji graf, ki je interaktiven. Črtni in ploščinski graf imata pod glavnim grafom prikazan še nižji graf, ki je namenjen izbiri časovnega obdobja za določen del trenutno prikazanega grafa. Ob premiku miške nad črtni graf se pokaže oblaček, v katerem piše čas in vrednost metrike ob tem času.

Poglavje 2

Obstoječa literatura

Napisanih je že veliko rešitev za prikaz porabe in beleženja virov, a vsaka ima točno določene funkcionalnosti, ki ne rešujejo vseh zahtev te diplomske naloge.

Ganglia [1] je odprtokodni skalabilen sistem za nadzor računalniških sistemov sestavljenih iz več računalnikov, kot je na primer oblak. A funkcionalnost te rešitve je omejena, saj je težko dodati nove metrike, ker so grafi prilagojeni le za točno določene metrike. Grafi niso interaktivni, in jih zato tudi ni možno razširiti z dodatnimi podatki o tekočih instancah v določenem trenutku.

Nobena od obstoječih rešitev ne ponuja vseh zahtev, zato je lažje narediti novo, ki izpolnjuje vse zahteve te diplomske naloge in omogoča enostavno razširljivost ob morebitnih kasnejših dodatnih zahtevah. Ker je ta rešitev bolj splošna od prej naštetih, se jo lahko uporabi tudi za prikaz porabe in beleženja virov, ki ne sodijo k infrastrukturi IaaS.

Poglavje 3

Opis ponudnikov

Obstaja veliko različnih ponudnikov IaaS, med katerimi so najbolj znani Amazon Web Services [2], Google Compute Engine [3], OpenNebula [4] in OpenStack [5].

3.1 OpenStack

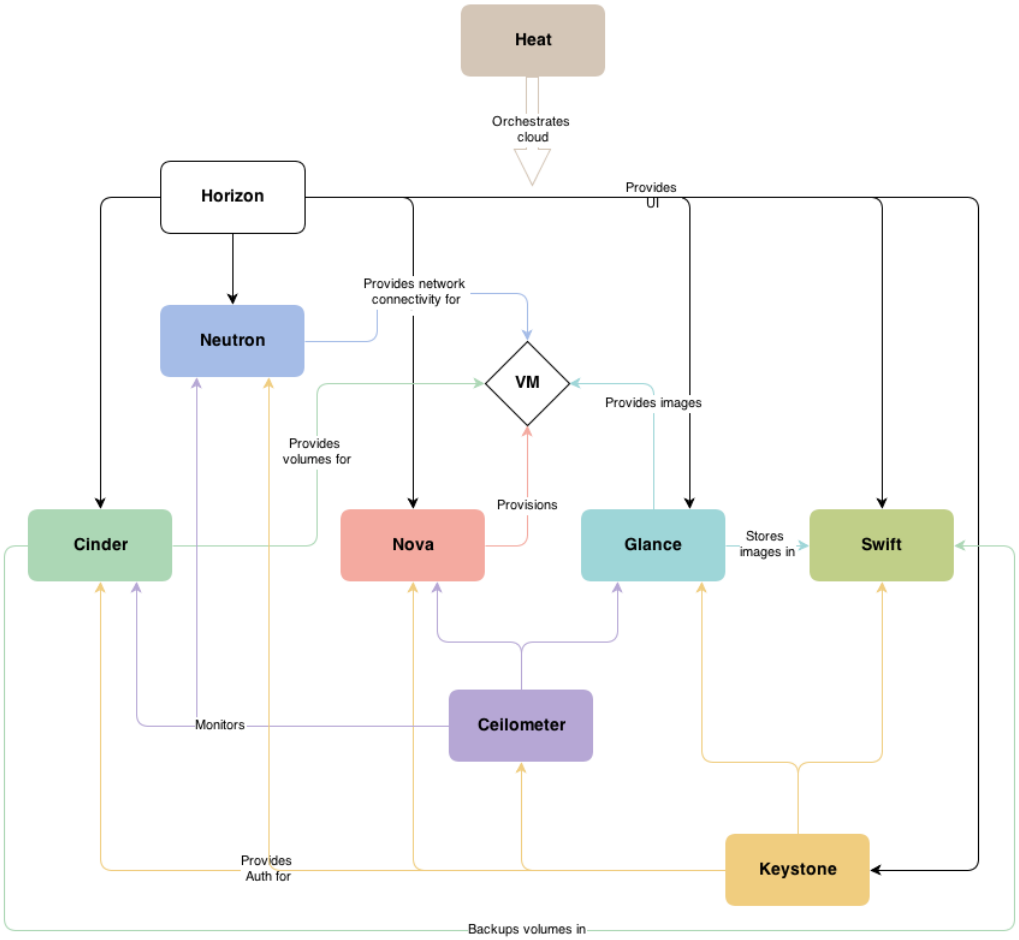
OpenStack [6] je odprtokodni projekt, ki ponuja infrastrukturo kot storitev za javne in zasebne oblake preko medsebojno povezanih storitev. Cilj projekta je zagotoviti prilagodljivo, elastično računalništvo v oblaku za velike in male oblake, ki jih je enostavno implementirati in so zelo skalabilni.

3.1.1 Arhitektura OpenStack

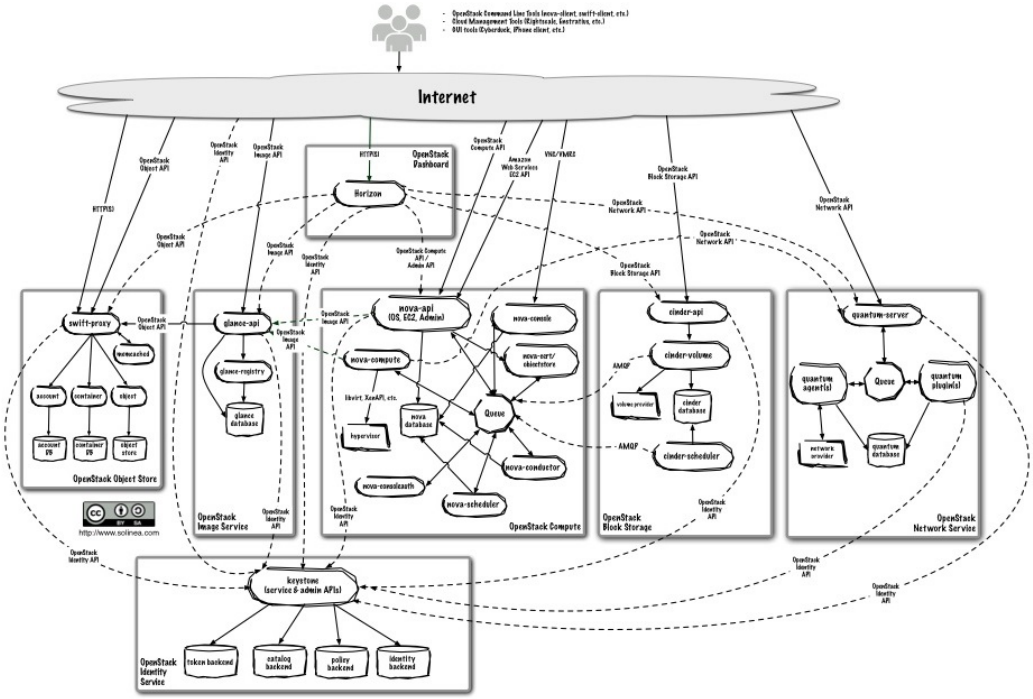
Projekt OpenStack je sestavljen iz povezanih modulov, ki pa so zamenljivi. Namesto privzetih modulov Swift za shranjevanje objektov in Cinder za bločno shrambo, se lahko uporabi na primer Ceph [7], ki je odprtokodni projekt katerega funkcionalnost je porazdeljena shramba objektov, bločna shramba in datotečni sistem.

Konceptualna arhitektura

Slika 3.1 prikazuje povezanost med procesi OpenStack.



Slika 3.1: Konceptualni načrt OpenStacka [6].



Slika 3.2: Logični načrt OpenStacka [6].

Logična arhitektura

Moduli OpenStack so naslednjih tipov: proces, ki teče v ozadju, skripta, ki namesti in testira virtualno okolje, in orodje, s katerim uporabnik povzroči klice vmesnika API (ang. Aplication Programming Interface) na OpenStacku.

Na sliki 3.2 je prikazana najbolj pogosta arhitektura za oblak OpenStack.

3.1.2 Storitve OpenStack

OpenStack je sestavljen iz naslednjih storitev:

Nadzorna plošča (angl. Dashboard) je modularna spletna aplikacija narejena v ogrodju Django [8], ki je grafični vmesnik za storitve OpenStack.

Storitev identitete (angl. Identity Service) ponuja storitve identitete, že-

tonov, kataloga in pravic drugim projektom OpenStack. Storitev sestavljajo:

- Keystone-all, ki zažene storitev in administrativni vmesnik API, da zagotovi storitve kataloga, avtorizacije in avtentikacije za ostale storitve OpenStacka.
- Funkcije storitve identitete, od katerih ima vsaka možnost vključitve vtičnikov, ki omogočajo različno uporabo storitve, od katerih večina uporablja LDAP (ang. Lightweight Directory Access Protocol) ali SQL (ang. Structured Query Language) ter shrambo ključ-vrednost - KVS (ang. key-value store).

Storitev računanja (angl. Compute Service) upravlja z navideznimi stroji, in je zato glavni del sistema IaaS. Sestavljena je iz naslednjih funkcijskih področij in njihovih komponent:

Vmesnik API

- Storitev nova-api sprejema in odgovarja na uporabnikove klice vmesnika API. Podpira OpenStack Compute API, Amazon EC2 (ang. Elastic Compute Cloud) API in poseben administratorski vmesnik API
- Storitev nova-api-metadata sprejema zahteve z instanc, ki se večinoma uporablja le takrat, ko storitev teče na več gostiteljih.

Računsko jedro

- Proces nova-compute, ki deluje v ozadju, ustvarja in ustavlja instance virtualnih strojev preko nadzornikovega vmesnika API. Podprti vmesniki API za nadzornike so na primer libvirt [9] za KVM (ang. Kernel-based Virtual Machine) [10] in QEMU (ang. Quick EMUlator)[11], VMwareAPI za VMware. Proces ustvarjanja in ustavljanja je zapleten, a osnovna ideja je sprejemanje ukazov iz vrste in njihovo izvajanje. En

ukaz iz ukazne vrste pomeni zaporedje več sistemskih ukazov, na primer zagon nove instance KVM, in posodabljanje stanja v podatkovni bazi.

- Proces nova-sheduler, ki je konceptualno najenostavnejši del kode v storitvi računanja, saj jemlje ukaze za ustvarjanje novega navideznega stroja iz ukazne vrste in določi na katerem računskem strežniku naj se zažene.
- Modul nova-conductor, ki posreduje interakcije med nova-compute in podatkovno bazo. Namenjen je odpravi neposrednega dostopa do podatkovne baze, ki ga zahteva nova-compute. Modul nova-conductor je skalabilen.

Omrežje za navidezne stroje

- Proces nova-network deluje v ozadju in podobno kot nova-compute sprejema omrežne naloge iz vrste in jih izvaja. Te naloge so spreminjanje omrežja, kot na primer nastavljanje omrežnih vmesnikov in spreminjanje pravil požarnega zidu. Funkcionalnost tega procesa se seli v ločeno storitev omrežja.
- Skripta nova-dhcpbridge sledi uporabi dinamičnih naslovov IP (ang. Internet Protocol) in jih beleži v podatkovni bazi. Tudi ta funkcionalnost se seli v storitev omrežja, ki uporablja drugo skripto.

Vmesnik konzole

- Proces nova-consoleauth avtorizira žetone za uporabnike, ki jih uporablja posredovalni strežnik konzole.
- Proces nova-novncproxy je posredovalni strežnik, ki omogoča dostop do izvajajočih instanc preko protokola VNC (ang. Virtual Network Computing). Podpira odjemalce novnc, ki delujejo v brskalniku.
- Proces nova-xvpnvncproxy je posredovalni strežnik, ki omogoča dostop do izvajajočih instanc preko protokola VNC. Podprt

je odjemalec v Javi, narejen posebej za OpenStack.

- Proces nova-cert upravlja s certifikati x509.

Upravljanje s slikami (kompatibilnost z vmesnikom EC2)

- Proces nova-objectstore ponuja vmesnik S3 [12] za vnos slik v storitev shranjevanja slik, kar je potrebno recimo za podporo orodju euca2ools [13], ki komunicira z nova-objectstore v jeziku S3 in nova-objectstore prevede S3 zahteve v zahteve, ki jih razume storitev shranjevanja slik.
- Orodje euca2ools je zbirka ukazov za konzolo, ki se uporabljajo za upravljanje z viri v oblaku. Čeprav to ni modul OpenStacka, se lahko nastavi nova-api, da podpira ta vmesnik EC2. Orodje euca2ools je bilo primarno razvito za uporabo s ponudnikom infrastrukture Eucalyptus [14].

Vmesnik ukazne vrstice

- Odjemalec nova omogoča uporabniku, da vnese ukaz kot najemnik, administrator ali končni uporabnik.
- Odjemalec nova-client omogoča vnos ukazov administratorjem oblaka.

Ostale komponente

- Vrsta je osrednji del pri posredovanju sporočil med procesi. Običajno se uporabi RabbitMQ [15], a se lahko uporabi katerokoli drugo AMPQ sporočilno vrsto, kot na primer Apache Qpid [16].
- Podatkovna baza SQL je komponenta, v kateri so shranjena stanja infrastrukture oblaka. Vsebuje tipe instanc, ki so na voljo za uporabo, instance v uporabi, omrežja in projekti. Storitev računanja lahko teoretično podpira katerokoli podatkovno bazo, ki jo podpira SQL Alchemy [17].

Storitev računanja komunicira z drugimi storitvami OpenStacka,

na primer s storitvijo identitete za avtentikacijo, s storitvijo shranjevanja slik in z nadzorno ploščo za spletni vmesnik.

Storitev shranjevanja objektov je zelo skalabilna, trpežna shramba objektov za veliko količino nestrukturiranih podatkov preko REST (ang. Representational state transfer) HTTP vmesnika API. Vsebuje naslednje komponente:

- Swift-proxy-server, ki razume vmesnik API shrambe objektov in HTTP zahteve za nalaganje datotek, spremembo metapodatkov in ustvarjanje vsebovalnikov. Spletnim brskalnikom tudi vrača datoteke ali spisek vsebine vsebovalnikov. Za izboljšano učinkovitost lahko posredovalni strežnik uporabi neobvezen predpomnilnik, ki je ponavadi memcache.
- Strežnik računov, ki upravlja z računi definiranimi v storitvi shranjevanja objektov.
- Strežnik vsebovalnikov, ki upravlja preslikavo vsebovalnikov v storitvi shranjevanja objektov.
- Strežnik objektov, ki upravlja dejanske objekte na shranjevalnih vozliščih.
- Številni periodični procesi, ki čistijo veliko podatkovno shrambo in storitev zrcaljenja, ki skrbi za konsistentnost in dosegljivost v gruči.

Storitev bločne shrambe omogoča upravljanje diskovnih pogonov, slik pogonov in tipov pogonov. Vsebuje naslednje komponente:

- Cinder-api, ki sprejema zahteve vmesnika API in jih preusmeri na cinder-volume.
- Cinder-volume, ki odgovori na zahtevo za branje in pisanje v podatkovno bazo objektne shrambe za ohranitev stanja in komunicira z drugimi procesi prek sporočilne vrste.

- Proces cinder-scheduler, ki podobno kot nova-scheduler, izbere optimalno vozlišče shranjevanja objektov, na katerem naredi nov pogon.
- Sporočilna vrsta, ki usmerja podatke med procesom storitve bločne shrambe in podatkovno bazo, ki shranjuje stanje pogona.

Storitev bločne shrambe komunicira s storitvijo računanja, kateri ponuja pogone za instance navideznih strojev.

Storitev shranjevanja slik vsebuje naslednje komponente:

- Glance-api, ki sprejema klice vmesnika API za odkrivanje, prejemanje in nalaganje slik.
- Glance-registry, ki shranjuje, obdeluje in pridobiva metapodatke o slikah. Metapodatki vsebujejo velikost, tip in druge lastnosti o slikah.
- Podatkovna baza, ki shranjuje metapodatke o slikah. Možno je uporabiti katerokoli SQL podatkovno bazo, na primer MySQL ali SQLite.
- Skladišče za shranjevanje slikovnih datotek. Na sliki 3.2, ki prikazuje logično arhitekturo OpenStacka, je razvidno, da je za skladišče uporabljena storitev shranjevanja objektov, a je možno uporabiti drugo skladišče. Storitev shranjevanja slik podpira navadne datotečne sisteme, bločne naprave RADOS (ang. reliable autonomic distributed object store), Amazon S3 in HTTP. Nekatere od naštetih možnosti omogočajo le bralni dostop.

Na storitvi shranjevanja slik se izvaja veliko periodičnih procesov, ki omogočajo predpomnjenje. Storitve za replikacijo zagotavljajo konsistentnost in razpoložljivost v gruči.

Kot je razvidno iz slike 3.1, ki prikazuje konceptualno arhitekturo, je storitev shranjevanja slik bistvenega pomena za celotno storitev infra-

strukture, saj za vmesnik API sprejema zahteve za slike ali metapodatke o sliki od končnega uporabnika ali storitve računanja in lahko shrani njihove diskovne slike v storitev shranjevanja objektov.

Storitev omrežja omogoča omrežno povezljivost kot storitev med mrežnimi vmesniki, ki jih upravljajo druge storitve OpenStacka.

- Neutron-server sprejema klice vmesnika API in jih usmerja na ustrezen vtičnik, ki izvede zahtevano operacijo.
- Vtičniki in agenti, ki priklaplajo ali odklaplajo vmesnike, ustvarjajo omrežja ali podomrežja in omogočajo naslavljanje IP. Ti vtičniki in agenti se razlikujejo glede na proizvajalca in uporabljeno tehnologijo v posameznem oblaku.

  Pogosti agenti so L3 (ang. Layer 3), DHCP (ang. Dynamic Host Configuration Protocol) in agent za vtičnike.
- Sporočilna vrsta je uporabljena za usmerjanje informacij med neutron-server in drugimi agenti ter za shranjevanje stanja za določen vtičnik.

Storitev omrežja komunicira večinoma s storitvijo računanja, kjer omogoča omrežja in povezljivost instancam.

Storitev merjenja in nadzorovanja je namenjena učinkovitemu zbiranju podatkov o zasedenosti CPE (centralna procesna enota) in omrežja. Podatke zbira tako, da spremlja obvestila, ki jih pošiljajo strežniki, ali pa s periodičnim branjem podatkov iz infrastrukture. Konfigurira tip podatkov, ki se zbirajo. Ogrodje je možno razširiti z vtičniki, ki podpirajo dodatne podatke o porabi virov. Sporočila so podpisana, tako da jih ni mogoče zavrniti.

Sistem sestavljo naslednje komponente:

- Agent, ki se izvaja na vsakem vozlišču in periodično zahteva statistiko porabljenih virov.

- Centralni agent, ki se izvaja na centralnem upravljalnem strežniku, kjer periodično zahteva statistiko zasedenosti virov, ki niso vezani na instance ali računska vozlišča.
- Zbirni agent, ki se izvaja na enem ali več centralnih strežnikih in nadzira sporočilno vrsto za obvestila ali podatke o porabi virov na agentih. Obdelana obvestilna sporočila spremeni v sporočila o porabi virov in jih pošlje nazaj v sporočilno vrsto. Sporočila o porabi virov se shranijo v podatkovno skladišče brez sprememb.
- Podatkovno skladišče, ki je podatkovna baza, katera je zmožna vzporedno obdelovati pisanja iz več zbirnih agentov in branja iz strežnika API.
- Strežnik API, ki se izvaja na enem ali več centralnih upravljalnih strežnikih in omogoča dostop do podatkov iz podatkovnega skladišča.

Te storitve komunicirajo preko standardnega OpenStack sporočilnega sistema. Samo zbirni agent in strežnik API imata dostop do podatkovnega skladišča.

Storitev samodejne konfiguracije zagotavlja samodejno konfiguracijo aplikacije v oblaku glede na opis delovanja aplikacije v predlogi. Storitev samodejne konfiguracije kliče OpenStackov vmesnik API, ki ustvari delujočo aplikacijo v oblaku. Ta storitev vključuje v eni predlogi vse ostale komponente OpenStacka. Predloge omogočajo ustvarjanje večine tipov virov, kot na primer instance, plavajoči naslovi IP, pogoni, varnostne skupine in uporabniki. Omogoča tudi napredno funkcionalnost, kot je visoka razpoložljivost instanc, samodejno skaliranje instanc in gnezdenje predlog. Podprt format za predloge je AWS CloudFormation.

Omogoča razvijalcem aplikacije integracijo neposredno s storitvijo samodejne konfiguracije ali pa preko vtičnika.

Storitev samodejne konfiguracije je sestavljena iz sledečih komponent:

- Orodje heat, ki se ga uporablja v ukazni vrstici in komunicira s heat-api, da lahko uporablja AWS CloudFormation API. Razvijalci lahko uporabijo tudi vmesnik API, ki ga ponuja heat.
- Heat-api, ki ponuja OpenStackov vmesnik API, ki procesira zahteve tako, da jih pošlje na heat-engine.
- Heat-api-cfn, ki ponuja AWS Query API, ki je kompatibilen z AWS CloudFormation in procesira zahteve tako, da jih pošlje na heat-engine.
- Heat-engine, ki organizira zagon predlog in vrača dogodke nazaj uporabniku vmesnika API.

Poglavje 4

Arhitektura rešitve

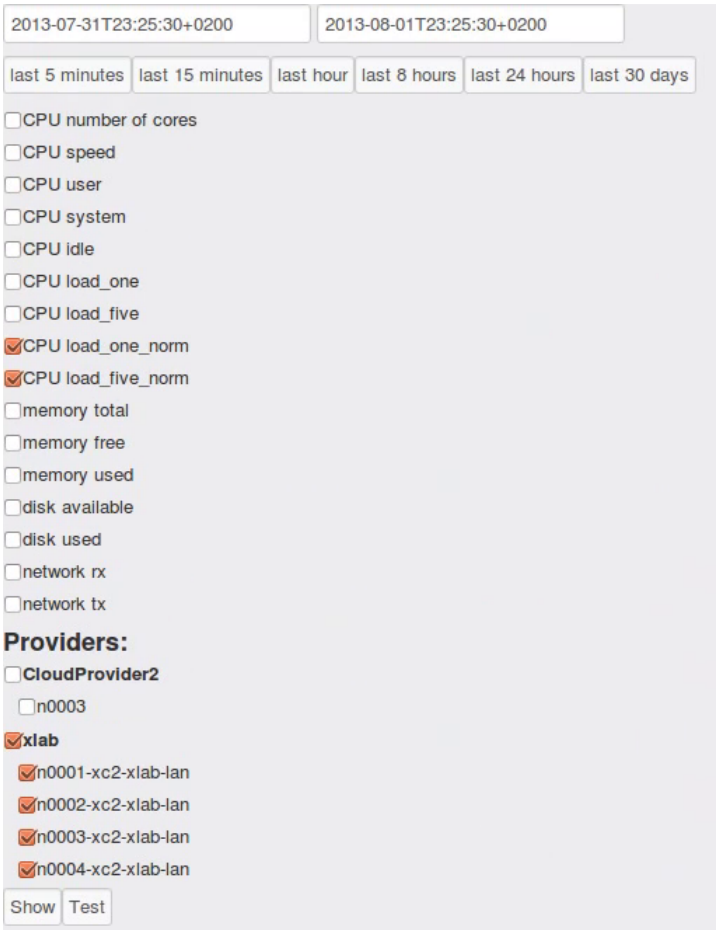
Ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS deluje tako, da se na vsakem strežniku beležijo podatki in se shranjujejo v podatkovno bazo. Brskalnik zahteva podatke iz podatkovne baze preko asinhrone REST zahteve, na katero odgovori strežnik, napisan v Javi.

Spletna aplikacija za upravljanje je napisana v ogrodju Django [8], zato je bil v njej narejen nov pogled (ang. view), na katerem se prikazuje zgodovina porabe virov. Osnovna stran je napisana v označevalnem jeziku HTML 5, oblika pa je opredeljena v ločeni datoteki v formatu LESS [18].

LESS je dinamičen jezik za opis oblike dokumenta. Sintaksa jezika LESS je gnezden metajezik, saj je koda v jeziku CSS veljavna tudi v jeziku LESS. LESS podpira spremenljivke, vključke, gnezdena pravila, funkcije, operacije in vključevanje drugih datotek LESS.

Ker brskalniki za oblikovanje podpirajo le jezik CSS [19], je potrebno datoteke LESS prevesti v jezik CSS. Prevajalnik za LESS je odprtokodni projekt in je napisan v jeziku JavaScript. Ker večina modernih brskalnikov podpira JavaScript, se lahko datoteko s kodo LESS prevede v brskalniku, ko se stran naloži. Ker prevajanje v brskalniku upočasni odzivnost spletne aplikacije, je to uporabno le za testiranje. Za spletno stran, ki je v produkciji, se datoteko LESS predhodno prevede v CSS.

Za vizualizacijo podatkov je uporabljena odprtokodna knjižnica D3.js na-



Slika 4.1: Izbira časovnega intervala, metrik in strežnikov.

pisana v jeziku JavaScript in ima poudarek na spletnih standardih. Knjižnica je namenjena manipulaciji objektov v DOM (ang. Document Object Model) glede na vezane podatke [21].

Knjižnica D3.js je klicana iz JavaScript kode, kjer je vsa logika ogrodja za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS.

Slika 4.1 prikazuje izbiro časovnega intervala, metrik in strežnikov. Izbira časovnega intervala, za kateraga nas zanima zgodovina porabe virov, ima več gumbov za hitrejšo izbiro časovnega obdobja, ki nas pogosto zanima. Izbirna polja z možnimi metrikami se zgradijo dinamično iz tabele možnih metrik.

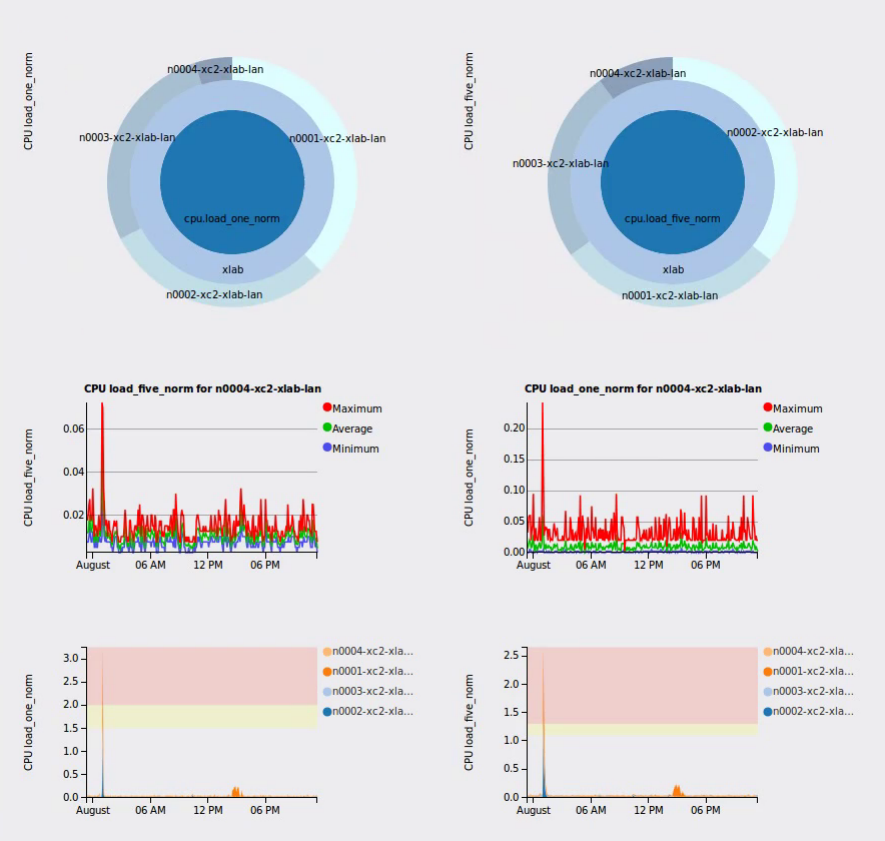
Ponudniki infrastrukture tipa IaaS in njihovi strežniki se dinamično naložijo. Za vsakega ponudnika posebej se izvede asinhron klic vmesnika API na strežniku, na katerem se izvaja Federation API, ki vrne imena strežnikov, ki jih ima določen ponudnik. Imena ponudnikov so odebeljena za boljšo razpoznavo, njihovi strežniki pa so zamaknjeni v desno, da je bolj opazno kateremu ponudniku pripadajo. Ob izbiri ponudnika se samodejno označijo vsi njegovi strežniki in ob ponovnem kliku na ponudnika se odznačijo vsi njegovi strežniki. Le ko so izbrani vsi strežniki nekega ponudnika, je označen tudi ta ponudnik.

Vse asinhrone zahteve uporabljajo protokol REST in uporabljajo format JSON (ang. JavaScript Object Notation). Ob kliku na gumb se za vsakega ponudnika, ki ima izbran vsaj en strežnik, zgradi zahteva, ki vsebuje: začetek in konec izbranega intervala, največje število podatkov, ime ponudnika in izbrane strežnike. Največje število podatkov se dinamično izračuna glede na širino grafa, da je število točk optimalno, saj v primeru prevelikega števila točk ne bi bilo mogoče ničesar razbrati iz grafa. Med nalaganjem podatkov je prikazana animacija da uporabnik ve, da še niso vsi grafi izrisani.

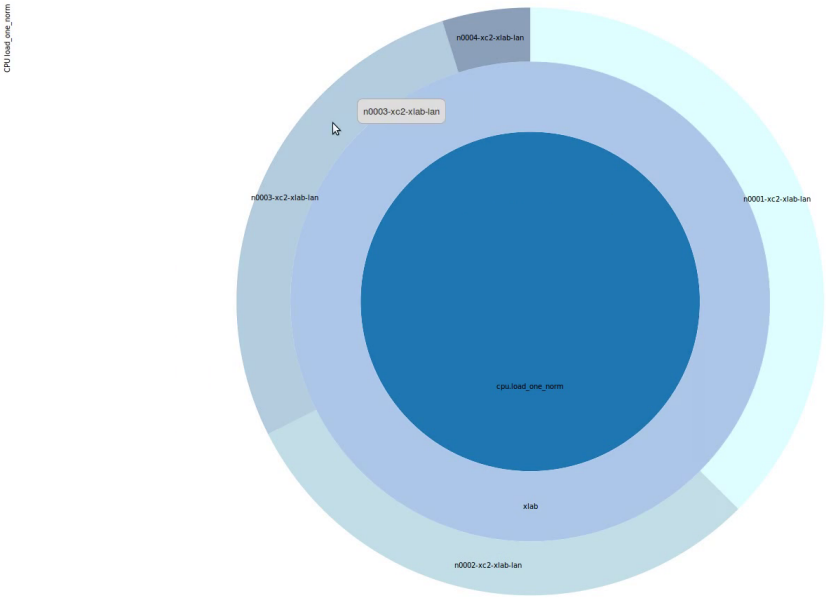
Ko se podatki naložijo, se na spodnjem delu strani pokažejo manjši grafi, na katere lahko uporabnik klikne, da se prikaže povečana verzija tega grafa. Majhni grafi niso le pomanjšana različica velikih grafov, saj ima njihova skala manjše število referenčnih točk, tekst na oznakah pa je enake velikosti, tako da je iz njih dobro razvidna poraba virov. Primer majhnih grafov je na sliki 4.2.

Kot tehnologija za izris grafov je uporabljen SVG (ang. Scalable Vector Graphics), saj je v brskalniku podprt kot del DOM. Zato se lahko graf spreminja enako kot ostale elemente DOM in se uporablja dogodke nad posameznimi elementi slike SVG.

Če uporabnika bolj podrobno zanima uporaba virov, prikazana na določenem grafu, klikne nanj, da se prikaže povečan graf, ki ni statična slika - tako kot majhen graf - ampak je interaktivna vizualizacija, ki s pomočjo animacij izboljša spremljanje sprememb na grafu. Obstajajo tri vrste grafov:



Slika 4.2: Majhni grafi.



Slika 4.3: Kolobarni graf.

kolobarni, črtni in ploščinski.

Kolobarni graf prikazuje zadnjo vrednost v izbranem obdobju za vsako metriko posebej. Iz njega je razvidna porazdeljenost porabe virov po ponudnikih infrastrukture in hkrati porazdeljenost po strežnikih posameznega ponudnika.

Vsak ponudnik ima svojo barvo. Za ponudnikove strežnike se skalira barvo ponudnika od malo svetlejše do malo temnejše. Ker so strežniki enega ponudnika podobne barve, je iz grafa dosti bolj razvidno, kateremu ponudniku pripada strežnik.

Ob premiku miškinega kazalca preko ponudnika ali strežnika se barva posvetli in se prikaže oblaček z imenom strežnika ali ponudnika. Na sliki 4.3 je prikazan graf za ponudnika xlab, ki ima štiri strežnike, katerih barve so podobne ponudniku.

Črtni graf prikazuje porabo virov za en strežnik in eno metriko. Ker je podatkov veliko in so zato zgoščeni, se na grafu prikaže največjo, povprečno in najmanjšo vrednost. Ob premiku miškinega kazalca preko grafa se v tisti

časovni točki nariše pokončna črta za lažje spremljanje višine črt, ki prikazujejo vrednosti. Pokaže se tudi oblaček, ki izpiše čas v tisti točki ter največjo, povprečno in najmanjšo vrednost.

Ozadje grafa je lahko obarvano belo, rumeno ali rdeče. Rumena barva opozarja, da je poraba virov presegla opozorilno mejo. Rdeča barva pa obvesti uporabnika, da je poraba virov višja od kritične meje in je treba primerno skalirati infrastrukturo. Meja, ko se obarva ozadje, je nastavljiva za vsako metriko vsakega ponudnika posebej. Meje se asinhrono naložijo le enkrat pri pridobivanju podatkov o obstoječih ponudnikih, saj bi kasnejše nalaganje mej upočasnilo odzivnost.

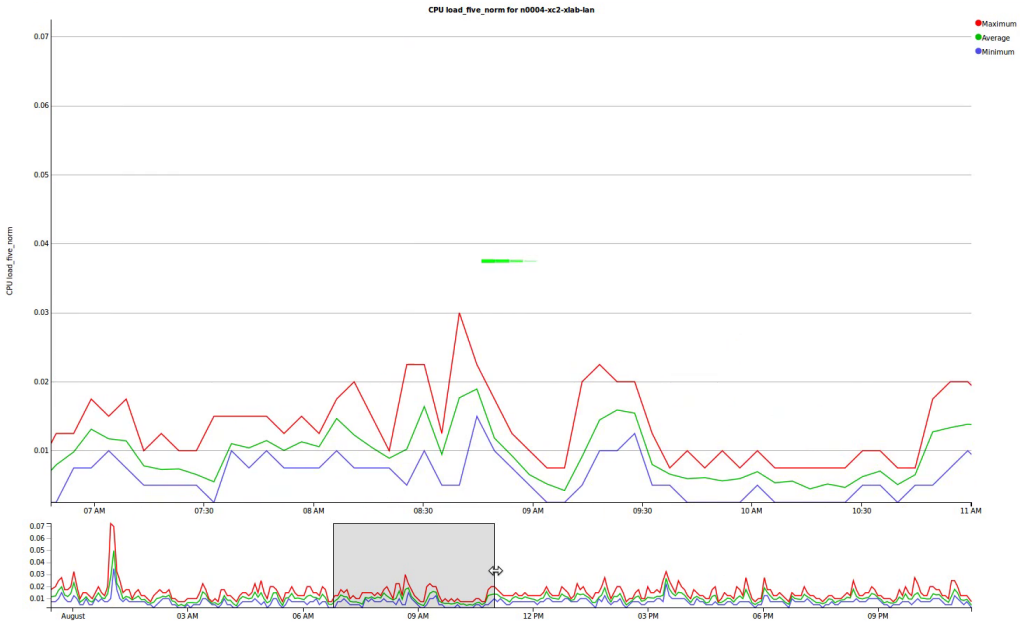
Na desni strani grafa je legenda, ki prikazuje, kaj pomeni katera barvna črta, nad grafom pa oznaka, ki prikazuje, katero metriko in kateri strežnik prikazuje graf. Pod dejanskim grafom je narisan še en graf, ki je zelo nizek, in se uporablja za izbiro časovnega področja zanimanja. Ko uporabnik izbira želeni časovni interval se zgornji graf sproti skalira tako, da je vedno prikazan le del grafa, ki je izbran na spodnjem grafu.

Ko uporabnik konča z izbiro področja, se začnejo nalagati bolj podrobni podatki. Med nalaganjem podatkov je prikazana animacija, ki uporabniku sporoča, da novi podatki še niso naloženi. Primer te animacije je prikazan na sliki 4.4.

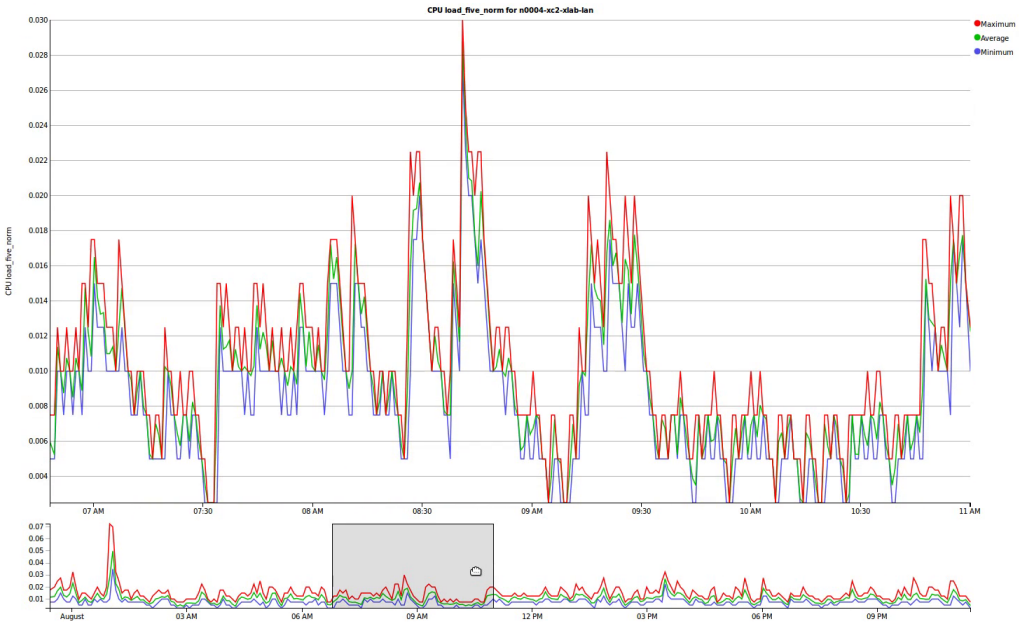
Ko se novi podatki naložijo, se prikažejo na grafu. Ker pa sta lahko nova najmanjša in največja vrednost manjši ali večji od trenutne skale, se skala animira na najmanjšo in največjo vrednost med vsemi novimi podatki. Ta animacija pripomore k lažjemu spremljanju in razumevanju podatkov, saj zmanjša število utripanj (kar na primer pri epileptičnih bolnikih lahko zmanjša verjetnost epileptičnega napada [20]).

Slika 4.5 prikazuje graf z novimi podatki po animaciji. Če uporabnik med nalaganjem novih podatkov spremeni področje zanimanja, se trenutno nalaganje prekliče.

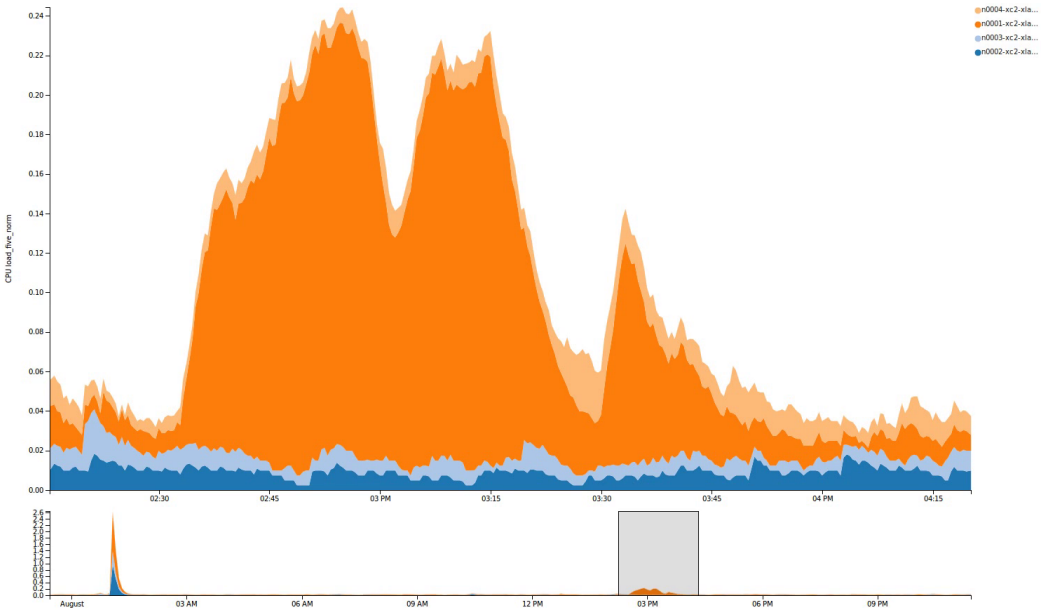
Podobno se ob začetku izbire novega časovnega področja zanimanja podatki zamenjajo s prvotnimi in skala se animira nazaj na prvotno. Ob hitrem



Slika 4.4: Animacija na črtnem grafu, ki pomeni, da se podatki nalagajo.



Slika 4.5: Prikazani podrobni podatki za izbran časovni interval.



Slika 4.6: Ploščinski graf z naloženim izbranim časovnim obdobjem.

premiku na novo vrednost se animacija prekliče, da ne moti pri spremljanju spremembam časovne skale.

Ploščinski graf prikazuje eno metriko za vse strežnike v oblaku. Iz njega je razvidna porazdlejenost porabe virov glede na čas. Vsak strežnik je narisan kot ploščina določene barve, ki v vsaki časovni točki zavzema višino, določeno v podatkih. Strežniki so narisani en nad drugim. Iz grafa je razvidna skupna poraba virov vseh izbranih strežnikov določenega ponudnika. Ob premiku miškinega kazalca preko strežnika, narisanega na grafu, se v oblačku prikaže strežnikovo ime.

Ploščinski graf ima enako kot črtni obarvano ozadje, če je presežen prag za določeno vrsto opozorila. To je vidno na sliki 4.2 na obeh ploščinskih grafih.

Na desni strani grafa je legenda, ki prikazuje kateri strežnik je prikazan s katero barvo. Če je ime strežnika predolgo, ga skrajša in doda tri pike na konec imena. Ob premiku miške preko strežnika, se pokaže njegovo polno ime. Na levi strani grafa je oznaka, ki prikazuje katero metriko prikazuje

graf.

Pod dejanskim grafom je narisan še en graf, ki je zelo nizek. Uporablja se za izbiro časovnega področja zanimanja, ki deluje enako kot pri črtnem grafu.

Poglavje 5

Ovrednotenje arhitekture

Za krajše časovne intervale se grafi prikažejo skoraj v trenutku, za daljše pa lahko traja tudi nekaj sekund. Čas med poslano zahtevo in prikazom je odvisen od čakanja na podatke s strežnika in časa za izris grafov iz pridobljenih podatkov. Čas izrisovanja grafov je konstanten (približno 5 milisekund), saj je število podatkov vedno omejeno na neko vrednost, ki je razmeroma majhna.

Čas pridobivanja podatkov s strežnika pa je odvisen od števila podatkov v bazi. Ker je v zahtevi največje število podatkov omejeno, je potrebno v primeru prevelikega števila podatkov, na strežniku podatke zgostiti. Zgoščevanje je počasno, saj se računa povprečna vrednost na vsakem intervalu.

Podatki o porabi virov se v bazo shranjujejo vsakih 5 sekund. V eni uri je torej shranjenih 720 podatkov, za katere je potrebnih nekaj milisekund, da se zgostijo. V 30 dneh pa je shranjenih 518400 podatkov, ki pa se obdelujejo nekaj sekund. Hitrost obdelave je odvisno tudi od števila izbranih metrik in števila izbranih strežnikov. Tudi število uporabnikov, ki istočasno zahtevajo podatke, vpliva na dolžino nalaganja zgoščenih podatkov.

Iz istega razloga kot pri nalaganju podatkov za vse grafe, je čas nalaganja podatkov pri črtnih in ploščinskih grafih, kjer se izbira podrobno časovno obdobje, odvisnen od dolžine izbranega časovnega intervala. Le da se pri

tem grafu zahteva le eno metriko, kar lahko nekoliko pohitri nalaganje. Pri črtnem grafu se naložijo podatki le za en strežnik, pri ploščinskem pa je potrebno naložiti podatke za vse strežnike izbranega ponudnika.

Poglavje 6

Možnosti nadaljnega dela

Ker se med samim razvojem vedno najde veliko novih idej, velikokrat ni mogoče vseh uresničiti. Lahko bi se naredilo mobilno aplikacijo ali spletno stran, prilagojeno mobilnim napravam.

Aplikaciji bi lahko bil dodan modul, ki bi s pomočjo pridobljenih podatkov poskušal napovedati prihodnjo porabo virov in bi s tem prikazal tudi čas, ko bosta preseženi opozorilna in kritična meja. S pomočjo tega modula bi ponudniki približno vedeli, kdaj morajo nadgraditi svojo infrastrukturo, s čimer bi svojim uporabnikom omogočili nemoteno uporabljanje njihovih virov.

Pri ploščinskih grafih bi se lahko, ob kliku na graf, na strežnik poslal čas, strežnik pa bi vrnil instance, ki so se izvajale ob tistem času, ki bi bile uporabniku prikazane v oblačku nad grafom.

Pri kolobarnem grafu bi se lahko namesto zadnje vrednosti uporabilo povprečje zadnjih nekaj minut, kar bi morda prikazalo bolj realno stanje porazdeljenosti porabe virov med strežniki.

Za pogoste dolžine intervalov bi se lahko povprečja računala sproti, kar bi zelo pohitrilo zahteve za daljša časovna obdobja.

Vsi grafi bi se lahko osveževali v realnem času. Če bi bil prikazan časovni interval majhen, bi se podatki lahko prikazovali takoj, ko se shranijo v bazo. Za bolj tekoče prikazovanje v realnem času bi se lahko podatki zajemali tudi

bolj pogosto, a bi se shranjevali le v predpomnilnik, da ne bi bilo preveč podatkov v bazi. Sprememba časovne skale bi bila animirana za lažje spremljanje dogajanja. Če bi bil časovni interval daljši, bi se spremembe lahko kazale z nekajminutnim zamikom.

Poglavje 7

Sklepne ugotovitve

Diplomska naloga vključuje ogrodje za prikaz porabe in beleženja virov nad ponudnikom infrastrukture tipa IaaS. To ogrodje prikazuje podatke za izbrano časovno obdobje. Grafi so interaktivni, saj je možno na že prikazanem grafu izbrati novo časovno obdobje, ki uporabnika bolj podrobno zanima.

Prednost tega ogrodja je splošnost, kar pomeni, da ni vezano le na enega ponudnika računalništva v oblakih. Ogrodje je zato možno enostavno prenesti tudi na druge sisteme, ki potrebujejo beleženje porabe virov. Možno je tudi uporabiti le del ogrodja, saj je koda razdeljena na več funkcij.

V ogrodju ni kode, ki bi obravnavala vsako metriko posebej, zato je možno prikazu dodati katerokoli metriko brez spreminjanja kode. Ogrodje uporablja spletne standarde, zato deluje v vseh brskalnikih, ki jih podpirajo.

Literatura

[1] (2013) Ganglia Monitoring System. Dostopno na: http://ganglia.sourceforge.net/

[2] (2013) Amazon Web Services. Dostopno na: http://aws.amazon.com/

[3] (2013) Google Compute Engine. Dostopno na: https://cloud.google.com/products/compute-engine

[4] (2013) About the OpenNebula.org Project. Dostopno na: http://opennebula.org/about:about

[5] (2013) Open source software for building private and public clouds. Dostopno na: http://www.openstack.org/

[6] (2013) OpenStack Compute Administration Guide. Dostopno na: http://docs.openstack.org/trunk/openstack-compute/admin/bk-compute-adminguide-trunk.pdf

[7] (2013) Ceph storage. Dostopno na: http://ceph.com/ceph-storage/

[8] (2013) Django - The Web framework for perfectionists with deadlines. Dostopno na: https://www.djangoproject.com/

[9] (2013) The virtualization API Dostopno na: http://libvirt.org/

[10] (2013) Kernel Based Virtual Machine. Dostopno na: http://www.linux-kvm.org/page/Main_Page

[11] (2013) QEMU opensource processor emulator. Dostopno na: http://wiki.qemu.org/Main_Page

[12] (2013) Amazon Simple Storage Service (Amazon S3). Dostopno na: http://aws.amazon.com/s3/

[13] (2013) Euca2ools Guide. Dostopno na: http://www.eucalyptus.com/docs/euca2ools/3.0/euca2ools-guide/

[14] (2013) The Eucalyptus Cloud. Dostopno na: http://www.eucalyptus.com/eucalyptus-cloud/iaas

[15] (2013) What can RabbitMQ do for you. Dostopno na: http://www.rabbitmq.com/features.html

[16] (2013) Apache Qpid. Dostopno na: http://qpid.apache.org/

[17] (2013) The Python SQL Toolkit and Object Relational Mapper. Dostopno na: http://www.sqlalchemy.org/

[18] (2013) LESS - The dynamic stylesheet language. Dostopno na: http://lesscss.org/

[19] (2013) Cascading Style Sheets. Dostopno na: http://www.w3.org/Style/CSS/

[20] (2013) Web Content Accessibility Guidelines. Dostopno na: http://www.w3.org/TR/WCAG20/#seizure

[21] (2013) Data-Driven Documents. Dostopno na: http://d3js.org/