UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Žiga Maretič

EVALVACIJA PERFORMANS IN NADZOR STREŽNIKOV V OBLAKU

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVA IN INFORMATIKE

MENTOR: doc. dr. Zoran Bosnić

Ljubljana 2012

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Žiga Maretič, z vpisno številko 63060185, sem avtor diplomskega dela z naslovom:

Evalvacija performans in nadzor strežnikov v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Zorana Bosnića,
- 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 Podpis avtorja :

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: 00160/2011
Datum: 06.09.2011

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

Kandidat: **ŽIGA MARETIČ**

Naslov: **EVALVACIJA PERFORMANS IN NADZOR STREŽNIKOV V OBLAKU**

PERFORMANCE ANALYSIS AND MANAGEMENT OF SERVERS IN CLOUD COMPUTING

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V zadnjih letih je področje storitev v oblaku v velikem porastu, saj veliko podjetij seli svoje aplikacije in podatkovne shrambe v okolja z dinamičnim zakupom in razporejanjem resursov.

Kandidat naj v diplomski nalogi naredi pregled izbranih ponudnikov storitev v oblaku. Izbere naj konkreten računski problem, metode merjenja performans oblačnih storitev in naj izmeri ter statistično ovrednoti rezultate meritev. Poda naj priporočila za izbiro ponudnika glede na izmerjene optimalnosti časa procesiranja in uporabe delovnega spomina.

Mentor:

doc. dr. Zoran Bosnić

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Za pomoč in usmerjanje pri pisanju diplomske naloge se v prvi vrsti zahvaljujem svojemu mentorju, doc. dr. Zoranu Bosniću.

Posebna zahvala gre tudi staršem, ki so me v času študija in pisanja diplomske naloge brezpogojno podpirali in me spodbujali.

Kazalo

Slike

Tabele

Grafi

Seznam uporabljenih kratic in simbolov

ISO/OSI: referenčni model, ki predstavlja modularno zgradbo protokolov
TCP/IP: internetni sklad protokolov, preko katerega teče internet
IP (angl. *Internet Protocol*): internetni protokol
PDU (angl.*Protocol Data Unit*): protokolarna podatkovna enota
IT: informacijska tehnologija
SLA (angl. *Service Level Agreement*): pogodba o zagotavljanju storitev
FCAPS (angl. *fault, configuration, accounting, performance, security*): referenčni model za nadzor omrežij
MIB (angl. *Management Information Base*): baza upravljalskih informacij
OID: objektni identifikator
MAC (angl. *Media Access Control*) strojna številka, zapečatena v vsaki omrežni napravi
SNMP (angl. *Simple Network Management Protocol*): protokol za nadzor omrežja
SSH (angl. *Secure Shell*): protokol za upravljanje računalnika na daljavo
WMI (angl. *Windows Management Instrumentation*): Microsoftov protokol za nadzor
JRE (angl. *Java Runtime Environment*): javansko okolje za poganjanje javankih aplikacij
PaaS (angl. *Platform as a Service*): platforma kot storitev
SaaS (angl. *Software as a Service*): programska oprema kot storitev
IaaS (angl. *Infrastructure as a Service*): infrastruktura kot storitev
OS: operacijski sistem
CPE: centralna procesna enota
RAM (angl. *Random Access Memory*): bralno-pisalni pomnilnik
XML (angl. *Extensible Markup Language*): jezik za grajenje strukturiranih dokumentov
RRD (angl. *Round Robin Database*): podatkovna baza Round Robin

Povzetek

V zadnjih letih je področje storitev v oblaku v velikem porastu, saj veliko podjetij seli svoje aplikacije in podatkovne shrambe v okolja z dinamičnim zakupom in razporejanjem virov. Kljub selitvi v oblak, želi vsak adminitrator IT nadzirati te aplikacije in strežnike kot običajne, v sistemski sobi podjetja. Prav zaradi porasta priljubljenosti storitev v oblaku, se na trgu pojavlja vse več ponudnikov teh storitev, ki ponujajo različno zmogljive primerke strežnikov v oblaku.

V diplomskem delu bomo s pomočjo meritev opravili evalvacijo performans na strežnikih različnih ponudnikov. Ob tem bomo z orodjem za nadzor tudi izvajali nadzor teh strežnikov. V zaključku diplomske naloge bomo podali ocene zmogljivosti primerkov strežnikov v oblaku in ponudnikov ter podali priporočilo za najboljšega ponudnika na podlagi naših meritev.

Ključne besede: nadzor omrežja, računalništvo v oblaku, evalvacija performans.

Abstract

In recent years, cloud computing has gained a lot in popularity, as many companies moved their applications and data storage into environments with dynamic resource allocation. Despite moving into the cloud, every IT administrator wants to monitor those applications and servers in the same way they monitor the traditional ones. And because of the popularity of cloud services the number of cloud providers is increasing. These providers offer a variety of cloud server instances with different performances.

This thesis will be based on measurments performed to evaluate performance of cloud servers of different vendors. At the same time we will use network management tool to perform monitoring of these servers. In conclusion of the thesis, we will make the evaluations of cloud servers and vendors and make recommendations for the best cloud services provider based on our measurements.

Keywords: network management, cloud computing, performance evaluation.

1. Uvod

1.1 Predstavitev problema

V sedemdesetih letih prejšnjega stoletja, ko so se pojavila prva računalniška omrežja, so bila le-ta namenjena raziskovalnim potrebam. Prenos podatkov je bil asinhron in počasen. Z razvojem tehnologije so se omrežja iz majhnih raziskovalnih omrežij razvila v veliko globalno infrastrukturo. Povečale so se tudi hitrosti. Naš način življenja je z leti postal odvisen od omrežij in interneta, ki ga uporabljamo v zasebnem življenju, še večji pomen pa ima omrežje v poslovnem svetu. V podjetju postane omrežje ključna povezava za prenos podatkov znotraj podjetja, med njegovimi dislociranimi enotami in strankami. Cilj vsakega podjetja je, da se širi in razvija, z njim pa se širi tudi omrežje, ki postaja vedno kompleksnejše. Vsaka izguba podatkov med prenosom, nepredviden izpad omrežja, okužba ali celo vdor v omrežje lahko resno ohromijo poslovanje podjetja in ogrozijo njegovo finančno stabilnost. Zaradi odvisnosti poslovanja podjetja od zanesljivega in varnega omrežja, se pojavi potreba po nadzoru omrežja. To omrežje sestavljajo računalniki, usmerjevalniki, stikala, strežniki itd., ki so večinoma različnih proizvajalcev. V odgovor temu je bil razvit protokol SNMP, ki predstavlja standardizirano orodje za nadzor omrežja. Protokol SNMP in njegove kasnejše verzije bomo podrobneje predstavil v diplomskem delu.

Formalna definicija pravi, da nadzor omrežja zajema dejavnosti, metode, procedure in orodja, ki se nanašajo na delovanje, administracijo in vzdrževanje omrežnih sistemov [1].

Človek sam ne mora biti kos zanesljivemu upravljanju velikih omrežij. Kompleksnost teh sistemov zato zahteva uporabo avtomatiziranih orodij, oziroma programskega paketa za upravljanje in nadzor [2].

Poleg tega, da je upravljanje omrežja izredno pomembno, pa za podjetje predstavlja tudi velik strošek. Podjetje mora zaposliti primerno usposoben kader, kupiti vso potrebno sistemsko in programsko opremo in programski paket za nadzor omrežja, skupaj z vsemi potrebnimi licencami. Vsako investicijo v upravljanje omrežja je treba upravičiti, zato sta izbira in nakup primerne rešitve za nadzor omrežja poslovna odločitev. Upravičimo jo s tem, da s pomočjo izbranega paketa konfiguriramo in nadziramo omrežje tako, da le-to deluje brez napak, da je omrežna oprema čim bolj izkoriščena in na voljo uporabnikom ter procesom takrat, ko jo potrebujejo. Na trgu obstajajo različni odprtokodni in komercialni paketi za nadzor omrežja. Komercialni paketi so običajno dragi in kompleksni, zato si jih lahko privoščijo le večja podjetja. Za mala in srednja podjetja so bolj primerne odprtokodne rešitve za nadzor omrežja, saj so običajno sredstva, namenjena za nadzor omrežja, omejena.

V zadnjih letih, ko sta razvoj in ponudba storitev računalništva v oblaku v velikem porastu, se veliko podjetij odloča, da svoje aplikacije in podatkovne shrambe (vsaj tiste nekritične za poslovni proces) preselijo v oblak. S tem se izognejo velikim stroškom nakupa in postavitve strojne in programske računalniške opreme, saj vire in zmogljivosti najamejo pri ponudnikih storitev v oblaku ter jih plačujejo le toliko, kot jih dejansko porabijo. Ker pa želi imeti vsak uporabnik običajno tudi nadzor nad storitvami, ki jih uporablja, tudi tistimi v oblaku, bomo v diplomskem delu predstavil možne povezave odprtokodnih orodji za nadzor omrežja z nadzorom storitev v oblaku.

Poleg samega nadzora storitev v oblaku s pomočjo sistema za nadzor omrežij bomo v diplomski nalogi tudi primerjali zmogljivosti strežniških primerkov v oblaku posameznih ponudnikov. Ob izvajanju teh testov bomo obenem spremljali, kako se sistem za nadzor odzove oziroma zabeleži obremenjenost teh primerkov. Za izvajanje testov bomo uporabili orodje CloudCmp, ki je bilo razvito na univerzi Duke prav z namenom primerjave posameznih ponudnikov storitev v oblaku.

2. Osnove računalniških komunikacij

Povezovanje računalnikov in drugih naprav z namenom, da bi bi si med njimi prenašali podatke, ni nekaj novega. Tehnologija lokalnih omrežij se je razvila že v sedemdesetih letih prejšnjega stoletja z namenom povezovanja terminalov in računalnikov v skupno omrežje. S tem smo v lokalnem omrežju omogočili prenos in dostop do podatkov ter računalnikov iz različnih oddaljenih lokacij, medtem ko se fizično nahajajo le na eni lokaciji. Lokalno omrežje predstavlja komunikacijsko pot med več računalniki, strežniki, delovnimi postajami, terminali in drugimi omrežnimi napravami, ki jih imenujemo gostitelji. Taka povezava omogoča uporabnikom dostop do vseh naprav v omrežju. Napravo v omrežje dodajamo preko tako imenovanega omrežnega vozlišča. Vsako vozlišče ima dodeljeno edinstveno naslovno številko, s katero se ta naprava predstavlja v omrežju. Vsako sporočilo, poslano po omrežju, mora vsebovati to naslovno številko, naprava priključena v omrežje pregleduje te podatke in jih primerja s svojo naslovno številko, če je sporočilo namenjeno njej. Prenos podatkov po lokalnih omrežjih poteka z velikimi hitrostmi (Mbps in hitreje) po skupnem prenosnem mediju, ki je omejen na majhno (lokalno) zemljepisno območje. Zaradi delitve skupnih virov v omrežju ima pomembno vlogo mehanizem, ki opravlja prenos in usmerjanje podatkov obenem pa vpeljuje pravila, kako in kdaj lahko naprave dostopajo do omrežja, da se pojavi čim manj napak in izgub pri prenosu teh podatkov.

2.1 Komunikacija v omrežju

V omrežju poznamo dva načina prenosa podatkov, to sta vodovni način (angl. *circuit switching*) in paketni način (angl. *packet switching*).

2.1.1 Vodovni način

V vodovnem načinu se vzpostavi neprekinjena povezava med dvema točkama. To je začasna povezava, ki obstaja dokler obe točki želita komunicirati, to je dokler povezava ni prekinjena. Vsi viri omrežja so na voljo samo tej povezavi, neodvisno od tega, ali se po povezavi prenašajo podatki ali ne. Ko se povezava prekine se omrežni vir ponovno sprosti drugim uporabnikom. Primer take povezave je telefonski klic. Prednost takega načina prenosa podatkov je ta, da imata obe strani rezervirano povezavo za prenos podatkov, dokler je povezava vzpostavljena. Očitna slabost take povezave pa so visoki stroški, ko se po njej prenaša malo oziroma nič podatkov. Taka povezava je tudi neučinkovita, če imamo veliko računalnikov, ki želijo komunicirati med seboj. Vzpostavitev povezave je sestavljena iz treh faz: vzpostavitev povezave, prenos podatkov in rušenje povezave [3].

2.1.2 Paketni način

Paketni način izboljša učinkovitost prenosa večjega števila podatkov po omrežju, saj si komunikacijski vod deli več uporabnikov. Tu se sporočilo razbije v več manjših paketov z določeno maksimalno velikostjo. Vsak paket vsebuje izvorni in ponorni naslov ter sekvenčno številko. Paketi so poslani v komunikacijski vod, kjer se pomešajo med ostale pakete drugih uporabnikov. Sprejemniki na komunikacijskem vodu preverjajo ponorne naslove paketov na vodu in sprejmejo samo tiste, ki vsebuje njihov naslov. Paketi običajno ne potujejo po istih poteh, zato je lahko vrstni red sprejema paketov drugačen, kot so bili poslani. Sprejemnik nato s pomočjo sekvenčne številke sestavi pakete v prvotno sporočilo. Paketni način lahko primerjamo z delovanjem pošte [3].

2.1.3 Datagrami in navidezni vodi

Znotraj lokalnega omrežja bodo vsi paketi slej kot prej prispeli na svoj ciljni naslov. Pri pošiljanju paketov v drugo lokalno omrežje, torej preko prostranega omrežja (angl. *wide area network* – WAN) je treba te pakete pravilno usmeriti. Paketni način prenosa podatkov podpira dva načina usmerjanja paketov – s pomočjo datagramov in s pomočjo navideznih vodov.

Datagrami omogočajo, da je vsak paket v omrežju neodvisno usmerjen. Ponorni naslov v podatkovni glavi paketa poskrbi, da se paket usmeri na svoj cilj. Zagotovila, da bo paket prišel na cilj pa ni, prav tako lahko pridejo paketi na cilj v drugačnem vrstnem redu, kot so bili poslani. Tak način prenosa se smatra kot nezanesljiv. Usmerjanje paketov po omrežju s pomočjo datagramov imenujemo nepovezavni način usmerjanja.

Usmerjanje paketov s pomočjo navideznega voda smatramo kot zanesljivo usmerjanje paketov. Tu se med pošiljateljem in prejemnikom vzpostavi navidezna povezava, vnaprej določena pot, po kateri se bodo paketi usmerjali. Zato tako usmerjanje imenujemo povezavno usmerjanje [3].

2.2 Tipi omrežij

2.2.1 Lokalna omrežja

Lokalno omrežje (angl. *local area network* – LAN) omogoča hiter prenos podatkov na omejenem geografskem ozemlju, običajno v zgradbi oziroma več sosednjih zgradbah. Omogoča hitrosti do 10 Mb/s na razdaljah do 500 metrov [3].

2.2.2 Prostrana omrežja

Medtem, ko so lokalna omrežja omejena na kratke razdalje, so prostrana omrežja (WAN) namenjena povezovanju lokalnih omrežij, ki so med seboj ločena z večjimi razdaljami. Delujejo s hitrostmi med 9600bps do 45 Mbps. Prostrana omrežja običajno uporabljajo

lokalna telekomunikacijska omrežja, ki so stroškovno učinkovita povezava med lokalnimi omrežji [3].

2.2.3 Velemestna omrežja

Velemestna omrežja (angl. *metropolitan area network* – MAN) so vmesni tip omrežij. Delujejo pri hitrostih med 56 kbps in 100Mbps, običajno višja hitrost kot pri prostranih omrežjih, a nižja kot pri lokalnih. Uporabljajo optična vlakna lokalnih telekomunikacijskih omrežij na razdalji nekaj sto kilometrov [3].

2.2.4 Navidezna zasebna omrežja

Navidezna omrežja so cenejša alternativa prostranim omrežjem, ki uporabljajo samostojno namensko paketno usmerjeno povezavo med dvema lokalnima omrežjema. Za povezavo uporabljajo obstoječo internetno infrastrukturo. Ker je internet javno omrežje, je zaradi varnosti podatkov, ki se prenašajo po navideznih zasebnih omrežjih, potrebno vpeljati varnostne mehanizme z enkripcijo [3].

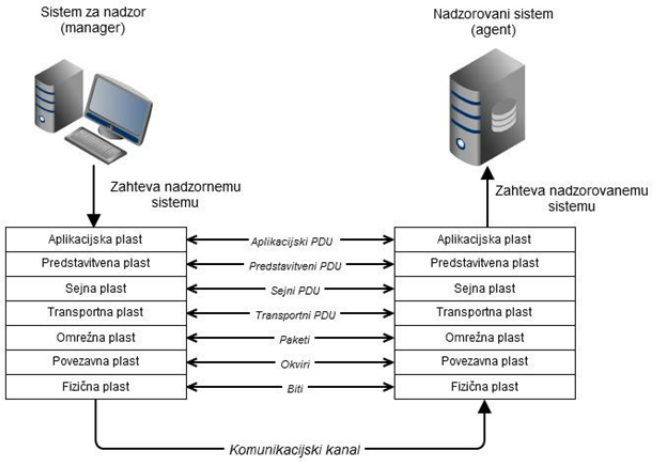
2.3 Referenčni model za povezavo odprtih sistemov (ISO/OSI)

Z vzpostavitvijo digitalne povezave med dvema napravama smo naredili prvi korak k postavitvi omrežja. Poleg strojnih zahtev, ki smo jih opisali zgoraj, moramo preprečiti tudi programske probleme v komunikaciji. Ob nastanku prvih omrežij to ni bil problem, saj so bile vse omrežne naprave enega samega proizvajalca. Taka omrežja imenujemo zaprt sistem. Z razvojem omrežij so se pojavili tudi novi proizvajalci opreme, s tem so se pojavile tudi težave v kompatibilnosti strojnega in programskega dela naprav in s tem potreba po standardizaciji. Tako imenujemo omrežja, ki uporabljajo naprave več proizvajalcev, katere sledijo določenim standardom, odprti sistem.

Zaradi širitve zaprtih sistemov je mednarodna agencija za standardizacijo (angl. *International Stadards Organisation – ISO*) leta 1987 izdala referenčni model za povezavo odprtih sistemov, v nadaljevanju model OSI (angl. *Open Systems Interconnection – OSI*). Model OSI je struktura, ki razdeli podatkovno komunikacijo na sedem obvladljivih plasti. Vsaka plast ima točno določen namen in vmesnik, preko katerega poteka komunikacija z višjo oziroma nižjo plastjo. Z upoštevanjem OSI modela se lahko naprave različnih proizvajalcev povezujejo med seboj iz vseh koncev sveta.

Prenos podatkov skozi plasti modela OSI (Slika 1) poteka tako, da sporočilo na oddajnikovi strani, ko ga ta pošlje v sistem, začne prehajati skozi plasti, od najvišje do najnižje. Na prejemnikovi strani se ta proces odvije v obratnem vrstnem redu. Sporočilo se ob oddaji razbije na pakete. Vsaka izmed sedmih plasti ob prehodu temu paketu doda svoje značilne podatke, ki jih shrani v glavo paketa. Kombinacijo podatkovnega dela paketa in njegove glave imenujemo protokolarna podatkovna enota (angl. *Protocol Data Unit – PDU*). Tako vsaka

nižja plast enkapsulira PDU višje plasti in skupaj s svojo dodano glavo tvori nov PDU, ki ga pošlje naprej po modelu OSI. Na prejemnikovi strani se ob sprejemu podatkov iz njih izlušči glava, preostali podatki pa se pošljejo višji plasti vse do najvišje plasti.



Slika 1: OSI model.

Plasti OSI modela si sledijo v naslednjem zaporedju:

- **fizična plast**: Na fizičnem nivoju določa prenosni medij, po katerem se bodo podatki prenašali. Skrbi za generiranje signalov in vrsto teh signalov v odvisnosti od prenosnega medija;
- **povezavna plast**: Ta plast je odgovorna za pošiljanje okvirja od sistema do sistema. Skrbi za odkrivanje in odpravljanje napak;
- **omrežna plast**: Skrbi za omrežno povezavo med naslovnikom in prejemnikom. Določa pot prenosa, skrbi za naslavljanje, fragmentacijo večjih paketov, skrbi za pretok in odpravlja zamašitve;
- **transportna plast**: Upravlja komunikacijo med dvema sistemoma;
- **sejna plast:** Nadzira komunikacijo med pošiljateljem in naslovnikom. Zagotavlja povezavo med dvema vozliščema, jo vzpostavi, vzdržuje in nato prekine. To povezavo imenujemo seja. Določa vrsto komunikacije;
- **predstavitvena plast**: Z različnimi načini šifriranja, stiskanja in kodiranja omogoča pretvorbo podatkov, ki so nato lahko pravilno interpretirani tudi na prejemnikovi strani;
- **aplikacijska plast**: Ta plast predstavlja vmesnik med uporabnikom in sistemom. Uporabnik preko nje komunicira s programom, ki se sestoji iz dveh delov: aplikacije in procesa. Aplikacija je uporabniški vmesnik, proces pa je program, ki v ozadju opravlja določene naloge na podatkovnem omrežju. Ta plast definira določene protokole za svetovni splet, elektronsko pošto, itd. [3].

2.4 Protokol TCP/IP

Začetki protokola TCP/IP segajo v zgodnja šestdeseta leta prejšnjega stoletja, ko je ameriško obrambno ministrstvo financiralo projekt v katerem naj bi agencija ARPA (angl. *Advanced Research Projects Agency*) na podlagi nove tehnologije prenosa podatkov oblikovala in razvila omrežje imenovano ARPANET (predhodnik današnjega interneta). Prvo vozlišče je bilo postavljeno na univerzi Berkley v Kaliforniji, do leta 1972 je bilo preko protokola TCP/IP povezanih že 40 vozlišč čez celotne ZDA. Leta 1973 pa sta bile vzpostavljeni tudi prvi mednarodni povezavi v Evropi z Anglijo in Norveško. Sprva je bilo omrežje namenjeno vladnim, vojaškim in izobraževalnim ustanovam. Počasi pa je protokol TCP/IP prehajal v hrbtenico današnjega interneta in do leta 1990 postal uporaben tudi širši javnosti ter postal *de facto* standard za svetovni splet [4].

2.4.1 Primerjava modelov ISO/OSI in TCP/IP

Čeprav sta se komunikacijska modela razvila v različnih državah in časovnih obdobjih, sta si zelo podobna. Model TCP/IP namesto sedmih plasti uporablja štiri. Zanaša se predvsem na srednji dve plasti. To sta transportna (gostitelj – gostitelj) in omrežna plast. Glavna naloga omrežne plasti je usmerjanje paketov, transportne plasti pa zagotavljanje celovitosti podatkov med pošiljateljem in prejemnikom, s tem da skrbi za zaznavanje in odpravljanje napak. Znotraj transportne plasti delujeta dva protokola:

- **UDP** (angl. *User Data Protocol*): To je nepovezaven protokol, ki je nezanesljiv, ker paketi potujejo po omrežju po principu datagramov. Ni preverjanja, ali so bili paketi dostavljeni ali ne in v kakšnem vrstnem redu so bili dostavljeni;
- **TCP** (angl. *Transmission Control Protocol*): To je povezavno usmerjen protokol. Zahteva potrditev povezave, zato je prenos med pošiljateljem in prejemnikom zanesljiv, brez napak ter podvajanja.

Model TCP/IP spada med t.i. *de facto* standarde, ki jih razvijajo in medsebojno potrdijo proizvajalci računalniške opreme.

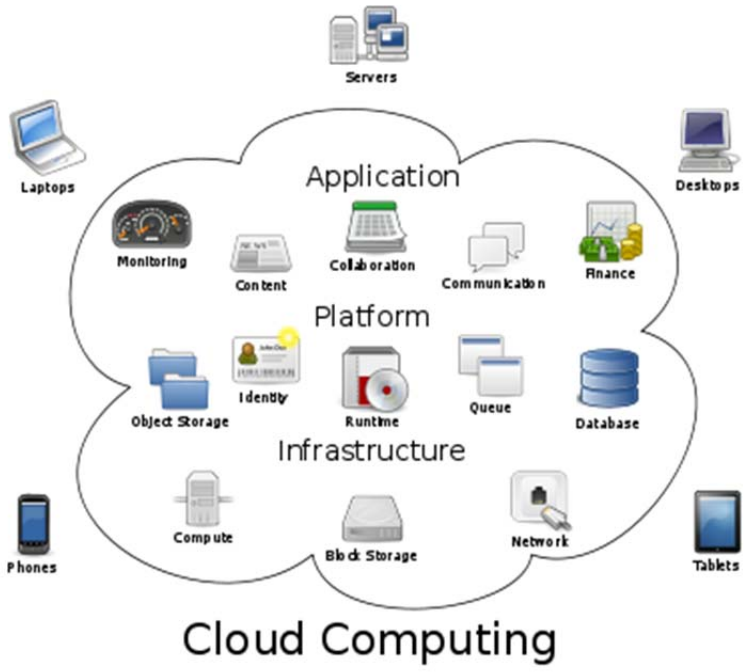
<table>
<tr><th>ISO/OSI</th><th colspan="2">TCP/IP</th></tr>
<tr><td>Aplikacijska plast</td><td rowspan="3">Aplikacijska plast</td><td rowspan="3">(telnet, FTP, SMTP, HTTP, DNS, POP,...)</td></tr>
<tr><td>Predstavitvena plast</td></tr>
<tr><td>Sejna plast</td></tr>
<tr><td>Transportna plast</td><td>Transportna past</td><td>(TCP, UDP)</td></tr>
<tr><td>Omrežna plast</td><td>Omrežna plast</td><td>IP</td></tr>
<tr><td>Povezavna plast</td><td rowspan="2">Povezavna plast</td><td rowspan="2"></td></tr>
<tr><td>Fizična plast</td></tr>
</table>

Tabela 1: Primerjava modela ISO/OSI in TCP/IP

3. Računalništvo v oblaku

V zadnjih letih računalništvo v oblaku (angl. *cloud computing*) prihaja vse bolj do izraza. Namen računalništva v oblaku je, da uporabniku ponuja dinamično razširljive, zmogljive, zanesljive in cenovno ugodne storitve, ki so dostopne preko preprostih spletnih vmesnikov kot storitev in ne kot produkt. Te pa se izvajajo na neki oddaljeni lokaciji, v oblaku, ki končnemu uporabniku ni znana.

Ime računalništvo v oblaku izhaja iz shematskega prikaza interneta in internetnega omrežja (Slika 2), ki ga poenostavljeno rišemo kot oblak, na katerega so priključene računalniške naprave.



Slika 2: Diagram računalništva v oblaku (vir: Wikipedia)

3.1 Značilnosti

Značilnost računalništva v oblaku je, da računalniške zmogljivosti zagotavlja ponudnik storitev, uporabnik pa za uporabo le teh potrebuje le osnoven računalnik z dostopom do interneta. Računalništvo v oblaku tako združuje tri osnovne pojme: programska oprema kot storitev (angl. *software as a service – SaaS*), platforma kot storitev (angl. *platform as a service – PaaS*) in infrastruktura kot storitev (angl. *infrastructure as a service – IaaS*). Ponudniki takih storitev so Amazon, Google, Microsoft, RackSpace in drugi [5]. Glavne značilnosti računalništva v oblaku so [10]:

- **samopostrežno povpraševanje po storitvah** (angl. *On demand self service*) – ponudniki storitev kot so spletna pošta, aplikacije, omrežne in strežniške storitve so uporabniku po potrebi avtomatsko dodeljene;
- **širok spletni dostop** (angl. *Broad network access*) – storitve so na voljo preko spleta s pomočjo računalnikov, pametnih telefonov, dlančnikov, tablic;
- **združevanje virov** (angl. *Resource pooling*) – ponudnikove računalniške kapacitete so realizirane tako, da lahko iste primerke strežnikov, platform in aplikacij strežejo množici uporabnikov (angl. *multy-tenacy*). Te se v fizični in virtualizirani obliki dinamično dodeljujejo glede na njihove potrebe;
- **elastičnost** (angl. *Rapid elasticity*) – uporabniki lahko hitro povečajo ali zmanjšajo porabo računalniške zmogljivosti ponudnika, tako da zagotavljanje teh kapacitet običajno izgleda kot neomejeno in so lahko zakupljene kadarkoli jih potrebujemo;
- **merjene storitve** (angl. *Measured service*) – storitve ponudnikov lahko izmerimo, kontroliramo in zagotavljamo preglednost porabe. Tako uporabnik plača samo tiste vire, ki jih je dejansko uporabljal in čas uporabe le teh (angl. *pay per use*).

3.2 Plasti računalništva v oblaku

Računalništvo v oblaku sestavlja pet ključnih plasti in dva zunanja akterja (Slika 3) [5]:

Slika 3: Plasti računalništva v oblaku

- **uporabnik** – podjetje, ki pri ponudniku najame oziroma zakupi storitve računalništva v oblaku;
- **odjemalec** – računalniška strojna in/ali programska oprema, ki za delovanje izkorišča storitve računalništva v oblaku in je brez njih neuporabna (prenosniki, pametni telefoni, dlančniki, itd.);
- **aplikacija** – aplikacija računalništva v oblaku je programska oprema kot storitev (*SaaS*), ki je dostavljena preko spleta in za katero ni potrebno lokalno nameščanje ter

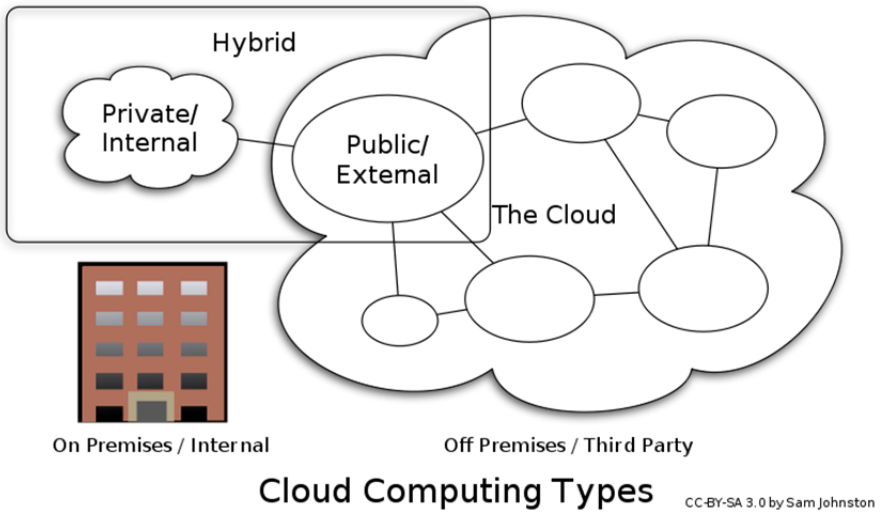
poganjanje na uporabnikovem odjemalcu. S tem sta poenostavljena tudi vzdrževanje in podpora;

- **platforma** – platforma kot storitev (*PaaS*) zagotavlja uporabniku platformo v oblaku za namestitev aplikacij brez velikih stroškov nakupa, kompleksnosti postavitve in vzdrževanja potrebne strojne ter programske opreme;
- **infrastruktura** – infrastruktura kot storitev (*IaaS*) uporabniku zagotavlja infrastrukturo, običajno kot platformo za virtualizacijo. S tem se izognemo nakupu strežnikov, mrežne opreme, prostora za sistemsko sobo, saj uporabnik to najame kot zunanjo storitev, ki se zaračuna po dejanski uporabi;
- **strežnik** – strežniška plast sestoji iz računalniške strojne in programske opreme, ki je specializirana za dostavo storitev v oblaku. Ta oprema vsebuje več jedrne procesorje, diskovna polja, operacijske sisteme prirejene za oblak, itd.;
- **ponudnik** – podjetje, ki zagotavlja in trži storitve v oblaku. Zagotavlja brezhibno delovanje takega sistema.

3.3 Različni modeli postavitev

Uporabnik storitev računalništva v oblaku lahko izbira med štirimi različnimi modeli postavitve [5]:

- **javni oblak** (angl. *public cloud*) predstavlja tradicionalni model računalništva v oblaku. Ponudnik storitev svoje vire in zmogljivosti ponuja uporabnikom preko spleta. Te so lahko brezplačne ali pa se zaračunajo po dejanski porabi;
- **zasebni oblak** (angl. *private cloud*) se od javnega oblaka razlikuje po tem, da so viri in zmogljivosti na voljo samo eni organizaciji, ki mora te vire in zmogljivosti kupiti, postaviti v svoji zgradbi in tudi opravljati ter vzdrževati;
- **hibridni oblak** (angl. *hybrid cloud*) je mešanica javnega in zasebnega oblaka. To je več oblačni sistem, ki je povezan na način, da so lahko programi in podatki uporabni v obeh povezanih modelih. V praksi to pomeni, da podjetje kritične procese vzdržuje v zasebnem oblaku, za nekritične pa ima najete vire in zmogljivosti v javnem oblaku;
- **oblaki skupnosti** (angl. *community cloud*) predstavljajo oblačno infrastrukturo, ki si jo deli več organizacij s skupnimi interesi in področji delovanja (npr. univerze). Prednost te oblike je, da se stroški najema računalniških storitev v oblaku razdeli med več organizacij, uporabnikov.



Slika 4: Različni modeli oblakov (vir: Wikipedia)

3.4 Prednosti in slabosti računalništva v oblaku

Računalništvo v oblaku je relativno nov koncept zunanjega izvajanja IT storitev in ima že na prvi pogled veliko prednosti, ki pa obenem, zanimivo, predstavljajo tudi njegove slabosti. V teh težkih ekonomskih časih podjetja krčijo sredstva, namenjena informacijski tehnologiji in se z najemom storitev v oblaku izognejo velikim investicijam v svoje lastne računalniške in strežniške zmogljivosti ter programsko opremo. S tem, ko podjetja preselijo svoje poslovne aplikacije, storitve, datotečne sisteme in podatkovne arhive v oblak lahko prihranijo veliko finančnih sredstev, vendar morajo pretehtati vse prednosti in slabosti, ki jih računalništvo v oblaku prinaša, te pa bomo opisali v nadaljevanju poglavja.

3.4.1 Prednosti

Računalništvo v oblaku podjetju s selitvijo podpornih funkcij v oblak omogoča ***poenostavitev in optimizacijo IT*****,** s tem podjetje skrči svoje lastne strežniške kapacitete kjer podpira optimizirano izvajanje strateško pomembnih procesov. Zaradi načina zaračunavanja storitev »*pay per use*« se ***odložijo in zmanjšajo stroški***, saj podjetje ne potrebuje več velikih podatkovnih centrov in njihovega vzdrževanja. Ni bojazni, da bi investirali v preveč zmogljivo strojno in programski opremo, ali še slabše, da bi kupili prešibko opremo, ki ne bi bila kos našim zahtevam. Potrebe in zmogljivosti, ki jih podjetje potrebuje za izvajanje procesov, se dinamično dodeljujejo trenutnim potrebam, te pa se plačajo po dejanski porabi. S tem obenem ***povečamo agilnost IT***, saj samo-oskrba z viri omogoči, da se informatika znebi kompleksnosti lastnih virov in morebitno nezmožnost odziva opreme na poslovne zahteve.

Zaradi dostopnosti storitev preko spleta je omogočen ***dostop do storitev in dokumentov potrebnih za delo od kjerkoli***. Uporabnik potrebuje le osnovno zmogljiv računalnik in dostop do interneta, da pride do potrebnih storitev, kar nam omogoča večjo fleksibilnost in mobilnost delovne sile. ***Izboljša neprekinjeno poslovanje in zagotavljanje okrevanja po katastrofi***, saj

ponudniki storitev v oblaku skrbijo za nemoteno delovanje naših storitev in njihove varnostne kopije. Tako podjetje ne potrebuje geografsko ločenega podatkovnega centra za omogočanje hitrega okrevanja po katastrofi, prav tako potrebuje manj strokovnjakov IT za upravljanje z opremo IT, ker nam ponudnik ***nudi ekspertno upravljanje IT*** [7, 8, 9, 10].

3.4.2 Slabosti

Največja slabost računalništva v oblaku je ***zagotavljanje zadovoljivega nivoja storitev*,** saj ima večina ponudnikov storitev v oblaku ***pomanjkljivo sestavljeno pogodbo o zagotavljanju storitev (***angl. *Service Level Agreement - SLA)* ***oziroma ta sploh ne obstaja***. Glede na to, da je računalništvo v oblaku nova in še neuveljavljena panoga računalništva, bi z izboljšanjem SLA ponudniki dosegli večje zaupanje potencialnih uporabnikov. Zanimivo je tudi vprašanje ***pravnega lastništva podatkov***. Ko se odločamo za prenos podatkov v oblak moramo biti pozorni, kaj se zgodi s temi podatki, če ponudnik storitve propade ali če se ti podatki izgubijo – so zagotovljene varnostne kopije? Kaj se zgodi s podatki, če prenehamo z uporabo storitev, saj podatki, ki so naša last ostanejo na podatkovnih poljih, ki so last ponudnika. Storitve v oblaku so ***odvisne od internetne povezave***, kar pomeni, da podjetje ne mora uporabljati svojih storitev v oblaku, če pride do izpada omrežja in kar je še bolj moteče, če omrežje deluje počasi, bo tudi delovanje storitev počasno. Zanimivo je, da se stroški pojavijo tudi kot slabost, saj se ne moramo izogniti ***vprašanju dejanskih stroškov***, saj kljub temu, da lahko na dolgi rok zmanjšamo stroške, nas lahko presenetijo nepredvideni dodatni stroški licenciranja morebitne programske opreme, ki se uporablja v oblaku in posodobitve storitev v oblaku. Velikokrat je vprašljiva tudi ***varnost***, saj so varnostni pristopi velikokrat pomanjkljivi. Do storitev namreč velikokrat dostopamo samo preko prijave z uporabniškim imenom in geslom, s čimer mislimo na morebitno odsotnost šifriranja in generacijo prešibkih gesel. Veriga pa je tako močna kot njen najšibkejši člen. Uporabnik storitev računalništva v oblaku mora imeti v mislih tudi ***odvisnost od ponudnika***, ki lahko čez čas pomeni velik problem. Pojavi se namreč vprašanje, ali lahko aplikacije in podatke, ki jih hranimo pri enem ponudniku računalništva v oblaku, na kakšen način brez večjih zapletov in dodatnih stroškov prenesemo k drugemu. Moramo pa se zavedati tudi, da je računalništvo v oblaku še v začetnih fazah razvoja kar vpliva na ***nezrelost trga ponudnikov***. Na trgu se namreč hitro pojavljajo novi ponudniki, obstoječi pa izginjajo oziroma se združujejo z drugimi ponudniki. Na to pa vpliva tudi ***težava z upravljanjem in nadzora oblaka***. Saj je tudi razvoj aplikacij za nadzor zmogljivosti in spremljanje storitev v oblaku še v začetni fazi razvoja, to področje pa bomo razdelali v nadaljevanju diplomske naloge [7, 8, 9, 10].

3.5 Nadzor oblaka

Računalništvo v oblaku revolucionizira omrežje in omrežne storitve, kar strokovnjakom IT, ki so navajeni nadzora svojega lokalnega omrežja in storitev, predstavlja velik izziv. Nadzor infrastrukture in storitev v oblaku je sicer podobno tradicionalnem nadzoru informacijske

infrastrukture, le da je obsežnejše, težje za nadzirati in bolj kompleksno. Ko enkrat prenesemo aplikacije in podatkovne centre preko zasebnih meja svojega lokalnega omrežja nekam v internet, v oblak, postane vprašanje nadzora še kako zanimivo. Namesto, da imamo strežnike, programsko opremo, aplikacije in podatkovni prostor namenjen za določena opravila na lokalnih virih, sedaj vse to postane abstraktno za uporabnika kot tudi administratorja IT. Tega sedaj bolj kot strežniki na katerih tečejo določene aplikacije zanimajo ali so te aplikacije dostopne uporabnikom in če delujejo pravilno. Programski paketi za nadzor omrežja se nahajajo na fizičnih strežnikih znotraj organizacije in zagotavljajo vidnost ter nadzor nad vso strojno in programsko opremo v omrežju, ta vidnost virov v omrežju pa je z računalništvom v oblaku odvzeta.

Pred računalništvom v oblaku se je že razvila storitveno orientirana arhitektura (angl. *Service Oriented Architecture – SOA*), ki nam je predstavila koncept abstrakcije virov. Glavno sporočilo SOA je bilo, da strojna in programska omrežna oprema nista več tako pomembni kot poslovno orientirane aplikacije, ki jih ti viri zagotavljajo. To je povzročilo, da so se programski paketi za nadzor prilagodili temu konceptu in poleg možnosti nadzora strojne in programske opreme omogočil tudi nadzor nad poslovnimi aplikacijami. Druga sprememba je prišla z razvojem virtualizacije, s čimer smo naredili delovno okolje abstraktno in s tem zmanjšali pomen fizične lokacije naših virov. Virtualizacija pa je zelo podobna oziroma skoraj enačena z zasebnim oblakom. Tudi v tem primeru so se morali programski paketi za nadzor prilagoditi tako, da je bil možen nadzor tudi nad virtualiziranimi strežniki.

V prihodnosti se bo vse več podjetji odločalo za hibridni pristop k računalništvu v oblaku. Tako, da bodo podporne storitve najverjetneje prenesli v oblak, pomembne strateške funkcije pa bodo še vedno izvajali na virih znotraj organizacije. To bo povzročilo, da se bodo morali programski paketi za nadzor prilagoditi in podpirati nadzor obeh pristopov na enoten način, saj v okolju računalništva v oblaku ne bo pomembno ali aplikacija teče v privatnem ali zasebnem oblaku.

Pri storitvah SaaS ne bomo morali nadzirati drugega kot dostopnosti in odzivnost aplikacije oziroma storitve. Zahtevneje bo izvesti nadzor nad IaaS in PaaS, saj običajno ne vemo natančno, kje se izvajajo, ker ponudnik dinamično dodeljuje vire teh storitev, da zagotovi največjo izkoriščenost svojih virov. Tako se poraja tudi vprašanje, kam natančno namestiti agenta za nadzor naprave [12].

Schultz nam predlaga nekaj smernic, ki naj se jih držimo, da lažje premagamo izzive, ki nam jih nalaga nadzor virov in storitev v oblaku [11]:

- **konsistenten podatkovni model** – večina podjetji ima standardno terminologijo v podatkovnih zapisih in bazah in to terminologijo morajo upoštevati tudi aplikacije v oblaku (zapis podatkov z enotnim, standardnim ID številkam) kar omogoča lažjo

integracijo. To je lahko kontrolirati, če oddelek IT vodi nakup in uvajanje aplikacije oziroma storitve v oblaku. Tega v nobenem primeru ne smemo prepustiti oddelku, ki bo to storitev uporabljalo;

- **integracija podatkov** – izbirati moramo takega ponudnika, ki bo omogočal integracijo podatkov trenutnega in morebitnega prihodnjega ponudnika aplikacij SaaS. Pomembno je tudi, da podpira enotno prijavo (angl. *Single Sign-on – SSO*) v naš hibridni oblačni sistem in dostop do podatkovnih baz;
- **zagotavljanje ekosistema ponudnikov** – eden večjih problemov nadzora storitev v oblaku, kjer imamo več aplikacij SaaS in internih aplikacij, ki se prepletajo med seboj, je koordinacija nameščanja popravkov in odpravljanja težav na eni aplikaciji, ker ta lahko vpliva na druge. Najbolje je, ko en ponudnik objavi popravek za svojo aplikacijo, ta popravek testirajo vse aplikacije, ki so od nje odvisne. S tem vzpostavimo ekosistem ponudnikov;
- **vzpostavitev t.i. ekipe *DevOps*** – ena večjih skrbi administratorjev IT je, da v določenem trenutku ne morejo nadzorovati svojega sistema v primeru izpada. Vzpostavitev tako-imenovane ekipe *DevOps* reši ta problem. To je ekipa strokovnjakov, sestavljena iz programskih inženirjev in sistemskih administratorjev, ki se zavedajo pomembnosti in medsebojne odvisnosti obeh vej informatike za nemoteno poslovanje podjetja. Zagotavljajo hitro oskrbovanje z viri in konfiguracijo v primeru težav, kar je ključnega pomena v oblačno usmerjenem poslovanju;
- **»drag and drop simplicity«** – da lahko v celoti izkoristimo dinamičnost zasebnega ali hibridnega oblaka, potrebujemo orodje za nadzor, ki nam bo omogočalo skrajšati čase, potrebne za določeno akcijo, zagotovilo večjo avtomatiziranost, omogočilo boljši nadzor nad porabo in zaračunavanjem virov, zagotovilo upoštevanje varnostnih standardov in seveda omogočalo hitro ustvarjanje novih delovnih okolij ter povečanje virov. To pomeni, da postavimo aplikacijo za nadzor oblaka nad že obstoječo aplikacijo za nadzor virtualiziranih strežnikov, saj bi tako prekrivanje omogočalo večjo avtomatizacijo in boljšo porabo virov ter možnost ustvarjanja novega delovnega okolja na navideznih strežnikih, ki ga nato preprosto povlečemo in spustimo (angl. *drag and drop*) v oblak;
- **kompatibilnost z več ponudniki** – izbrati moramo tako orodje za nadzor oblaka, ki bo podpiralo več ponudnikov storitev računalništva v oblaku, tudi, če trenutno uporabljamo storitve samo enega. To nam bo ob dodajanju novih storitev v oblaku omogočilo kasnejše neodvisno odločanje pri izbiri ponudnikov;
- **nadzor stroškov** – potrebujemo tako orodje, ki bo znalo beležiti porabo virov in aplikacij v oblaku ter to porabo stroškovno ovrednotiti. Tu moramo premagati problem različnega načina zaračunavanja, ki je odvisen od ponudnika in aplikacije, ki jo uporabljamo.

V splošnem so torej izzivi pri nadzoru oblaka zelo podobni standardnem nadzoru informacijske infrastrukture. Prav tako vključuje nadzor nad strojno in programsko omrežno opremo ter aplikacijami, ki tečejo na njej, kot tudi upravljanje s politiko informatike in upravljanje pogodb SLA [11].

V nadzoru oblaka bo pomembno izoblikovati nek ekosistem med ponudniki storitev računalništva v oblaku, kot tudi ponudniki aplikacij za nadzor [12].

4. Nadzor omrežja

4.1 Referenčni model FCAPS

Kot smo že omenili v uvodu, nadzorovanje omrežja zajema dejavnosti, metode, procedure in orodja, ki se nanašajo na delovanje, administracijo in vzdrževanje omrežnih sistemov. Ker človek sam ne mora biti kos zanesljivemu upravljanju velikih omrežij, si pri tem pomaga z uporabo avtomatiziranih orodji, ki se poslužujejo referenčnega modela FCAPS (*fault* – napake, *configuration* – nastavitve, *accounting* – kalkulacije, *performance* – zmogljivost, *security* – varnost). Ta model je v osemdesetih letih prejšnjega stoletja organizacija OSI definirala v standardu ISO. Model, kot je razvidno iz imena, razdeli nadzor omrežja na pet samostojnih in lažje obvladljivih kategorij: upravljanje z napakami, upravljanje z nastavitvami, upravljanje s kalkulacijami, upravljanje z zmogljivostvmi in upravljanje varnosti [1].

4.1.1 Upravljanje z napakami

Upravljanje z napakami se ukvarja z zaznavanjem in poročanjem o napakah v omrežju, kot so na primer napake na napravah, programski opremi in tudi komunikacijskih storitvah, ki ne delujejo pravilno. Z nadzorom omrežja skrbi, da vse deluje pravilno in da, ko to ni tako, o tem obvesti upravljalca omrežja. Dober sistem z dobro definiranim pragom (angl. *treshold*), omogoča tudi predvidevanje napak, tako da obvesti upravljalca o doseženem pragu, ta pa lahko dovolj hitro ukrepa, da ne pride do resnejših težav na omrežju. Funkcije nadzora napak vključujejo nadzor omrežja, upravljanje z alarmi, diagnosticiranje napak, iskanje izvora napak, odpravljanje napak, upravljanje z zgodovino alarmov in prijavo napak.

4.1.2 Upravljanje z nastavitvami

Da bo omrežje opravljalo svojo nalogo, ga je najprej potrebno nastaviti. Upravljanje z nastavitvami skrbi za načrtovanje, nastavljanje, spreminjanje konfiguracij in vzdrževanje naprav v omrežju od prvega priklopa dalje, da to lahko deluje kot celota. Poleg konfiguracije naprav moramo vedeti, katere naprave imamo v omrežju, kako so nastavljene in katere storitve delujejo na njih. Temu primerno moramo zagotavljati ažurne varnostne kopije vseh nastavitev naprav v omrežju, da lahko ob izpadu omrežja le-te ponovno pravilno konfiguriramo. Upravljnanje z nastavitvami tesno sodeluje z nadzorom z napak in zmogljivosti. Če pride do napake na omrežni napravi jo lahko z rekonfiguracijo omrežja obidemo, ali pa če nadzor zmogljivosti zazna neuravnoteženo obremenitev omrežja ponovno z rekonfiguracijo dosežemo, da se promet uravna.

4.1.3 Upravljanje s kalkulacijami

Upravljanje s kalkulacijami v največji meri skrbi za beleženje porabe omrežnih virov. Podjetja, ki svoje storitve ponujajo širši javnosti, morajo z zaračunavanjem teh storitev

ustvarjati prihodek in s tem dobiček, da lahko preživijo. To sta tudi glavni nalogi upravljanja kalkulacij. Tudi če podjetje uporablja omrežne vire samo za lastne potrebe svojih oddelkov, mora beležiti porabo le-teh in to porabo nekako ovrednotiti. Te stroškovne ocene omogočajo, da ima podjetje pregled nad stroški, ki jih ima z omrežnimi storitvami. Na podlagi teh pa se lahko odloči, ali bo te storitve še naprej izvajalo znotraj podjetja ali pa po za to najelo zunanjega izvajalca. Od upravljanja s kalkulacijami se zahteva najvišja stopnja razpoložljivosti in zanesljivosti, saj ob njegovem izpadu izgubimo nadzor nad porabo virov in s tem izpad prihodka od svojih storitev.

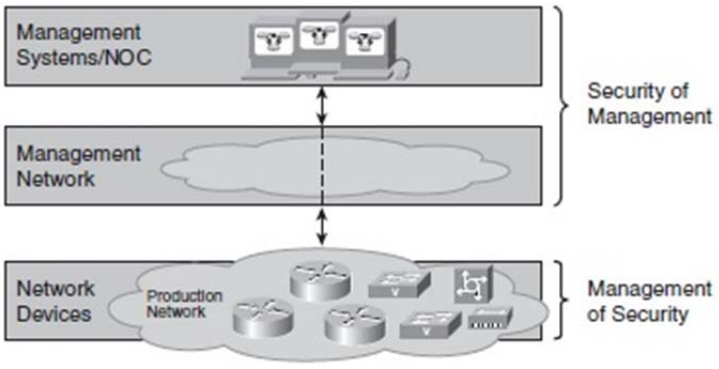
4.1.4 Upravljanje z zmogljivostvmi

Predpogoj nadzora omrežja je merjenje oziroma upravljanje z zmogljivostvmi, saj ne moremo nadzirati sistema in aktivnosti na njem, če ne nadziramo njegovih zmogljivosti. Ko imamo v mislih nadzor zmogljivosti govorimo o več parametrih, kot so [2]:

- **dostopnost**, ki jo izrazimo kot procent časa, ko je omrežje, omrežna naprava ali aplikacija na voljo uporabnikom,
- **odzivni čas** je čas, ki ga sistem potrebuje, da se odzove na določeno zahtevo,
- **natančnost** oziroma pravilnost podatkov, ki so prenešeni po omrežju z merjenjem napak pri prenosu,
- **prepustnost** (angl. *throughput*) z merjenjem prenosa podatkov v časovni enoti, opravljenih transakcijah, številu obdelanih klicev, itd.,
- **izkoriščenost** virov z iskanjem in odpravljanjem ozkih grl v omrežju, saj zasičenost določenega vira vpliva na odzivni čas sistema in tudi ostale parametre, ki jih merimo znotraj upravljanja zzmogljivostvmi.

4.1.5 Upravljanje varnosti

Upravljanje varnosti skrbi za zaščito omrežja pred zlonamernimi vdori in neželenimi okužbami z virusi in črvi. Znotraj upravljanja varnosti moramo ločiti med varnostjo upravljanja, ki skrbi, da je sam proces upravljanja varen in upravljanjem varnosti, ki skrbi, da je varno omrežje in naprave v njem (Slika 5).



Slika 5: Varnost upravljana proti upravljanju varnosti [1]

Varnost upravljanja se ukvarja z zagotavljanjem varnosti operacij upravljanja in nadzora omrežja. Velik del tega je zagotavljanje, da imajo dostop do sistema za nadzor samo za to pooblaščeni uporabniki. Nepooblaščen dostop do sistema lahko pomeni zlorabo sistema, degradacijo in prekinitev storitev ali omogoči dostop nepooblaščenim uporabnikom do določenih storitev. Moramo se zavedati, da sistem ni ogrožen samo z zunanjimi grožnjami, ampak tudi pred notranjimi zlorabami, pred katerimi se je težje obvarovati. Te preprečimo z dodeljevanjem pravic uporabnikom za dostop do aplikacij in omrežnih virov, ki ji ti res potrebujejo za opravljanje dela. Od uporabnikov moramo zahtevati uporabo kompleksnih gesel, ki jih morajo redno spreminjati. Zagotoviti moramo pregled nad spremembami uporabnikov in njihovimi pravicami ter omogočiti varnostne kopije teh podatkov.

Upravljanje varnosti zajema varovanje samega omrežja. Običajno so tarče zunanjih napadov na omrežja strežniki in računalniki končnih uporabnikov in ne omrežja sama. Najbolj tipične grožnje se pojavljajo v obliki virusov in črvov, vdori hekerjev ter napadi DOS (angl. *Deinal Of Service*), ki poskušajo obremeniti omrežje oziroma strežnike z nepooblaščenim prometom in tako onemogočijo prenos legitimnih podatkov po omrežju. Proti vdorom in grožnjam se zavarujemo tako, da spremljamo promet na omrežju in poskušamo z različnimi vzorci zaznati nenavadne spremembe, ki bi lahko kazale na vdor. Pomembna je tud uporaba požarnih pregrad in protivirusnega programa.

4.2 Programski paket za nadzor omrežja

Programski paket za nadzor omrežja deluje po modelu upravljalec – agent. Ta model je sestavljen iz nadzornega sistema (angl. *manager*), nadzorovanega sistema (ang. *agent*) in baze upravljalskih informacij (angl. *Management Information Base - MIB*) kjer so shranjene informacije o nadzorovani napravi. Ta programski paket v obliki aplikacije predstavlja vmesnik med administratorjem omrežja in fizičnimi napravami. Preko te aplikacije administrator lahko opravlja nadzor omrežja.

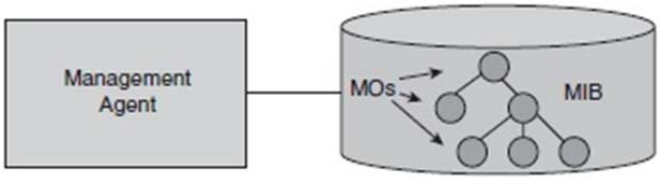
4.2.1 MIB

Kadar upravljalec in agent komunicirata, pogovor vedno teče o napravi, ki je nadzorovana. Vse, kar mora upravljalec vedeti o nadzorovani napravi, predstavlja upravljalska informacija (angl. *management information*). Osrednja naloga upravljalske informacije je, da vzpostavi skupni jezik, ki uporablja poenotene oznake in terminologijo, med upravljalcem in agentom. Deli upravljalske informacije, na katere se nanaša pogovor med upravljalcem in agentom, imenujemo objekti nadzora, ki sestavljajo bazo upravljalskih informacij – MIB nadzorovane naprave.

MIB si najlažje predstavljamo kot konceptualno shrambo podatkov. Upravljalec lahko dostopa do teh podatkov z zahtevki »get«, ki jih pošlje agentu. Lahko tudi spreminja podatke v MIB. To mu omogočajo zahtevki »set«, »create« ali »delete«. Kadar upravljalec pridobiva

informacije iz MIB, te predstavljajo nek parameter, ki ga nadziramo (npr. interni register, ki sledi številu prenesenih paketov prek vrat). Upravljalec lahko tudi spreminja informacije v MIB in s tem nastavi določen parameter nadzorovane naprave, s čimer se spremeni tudi delovanje naprave.

MIB torej administratorju omrežja omogoča nadzor in upravljanje z napravami. Vsebuje veliko individualnih informacij naprave. Te so fizične narave, kot so vrata, strojne komponente in logične narave, na primer protokoli, programska oprema in komunikacijske storitve. Dele, ki sestavljajo MIB, imenujemo objekti nadzora (angl. *managed objects – MO*) in so prikazana v hierarhični strukturi (Slika 6) [1].



Slika 6: MIB in MO [1]

MIB vsebuje več kategorij informacij nadzora, ki jih moramo ločiti, saj aplikacija za nadzor vsako izmed njih obravnava drugače in jo uporablja v druge namene.

- **informacija stanja** vsebuje podatke o trenutnem fizičnem in logičnem stanju naprave in njenih virih;
- **informacija o fizičnih nastavitvah** vsebuje podatke o tipu naprave, fizični konfiguraciji kartic in vratih, ki so na voljo, serijski številki in naslovu MAC. Teh nastavitev ne moramo spreminjati;
- **informacija o logičnih nastavitvah** vsebuje podatke o nastavitvah parametrov kot so naslov IP, telefonske številke in logični vmesniki. Te nastavite lahko administrator po potrebi spremeni;
- **zgodovinski podatki** vsebujejo zapise o preteklih dogodkih na napravi, kot so napake, spremembe konfiguracije, podatkov o povezavah itd.

4.2.2 Upravljalec

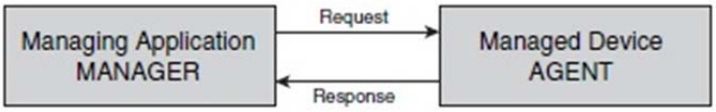
Upravljalec neprestano pošilja zahtevke agentom in dobiva njihove odzive, hrani dogodke in statistiko, ob izpadu pa sproži alarm. Aplikacija za nadzor običajno vključuje tudi grafično predstavitev meritev na nadzorovanih naprave, prikaz statistike in alarmov.

Slaba stran upravljalca je, da nadzoruje omrežje, od katerega je odvisno njegovo delovanje. Če upravljalec preneha delovati, bo omrežje še vedno delovalo brez posledic, le administrator ne bo več sposoben nadzorovati omrežja in njegovih morebitnih izpadov. Čas odprave napak se bi tako močno podaljšal, kakovost storitev bi začela padati, administrator ne bi mogel dodajati novih uporabnikov, storitev itd. Tako pridemo do dejstva, da je upravljalec nujen za

nemoteno delovanje omrežja. Da bi preprečili odpoved upravljalca lahko uvedemo distribucijo aplikacije za nadzor na več naprav, strežnikov. S tem postane sistem bolj robusten in redundanten, saj ob odpovedi enega strežnika sistem nadzora še vedno deluje na drugem.

Interakcije upravljalca

Najosnovnejša oblika interakcije med upravljalcem in agentom sestoji iz izmenjave zahtevkov in odzivov (Slika 7). upravljalec tvori zahtevek za pridobitev informacije za nadzor, spremembo nastavitev ali ukaza za zagon storitve. Agent se posledično odzove z odzivom, ki vsebuje obvestilo o tem ali se je zahtevek uspešno izvedel ali pa je prišlo do napake.



Slika 7: Interakcija med upravljalcem in agentom [1]

Zahtevek tipično vsebuje naslednje parametre:

- tip zahtevka,
- informacijo objekta nadzora na katero se zahtevek nanaša,
- dodatne informacije, kot so na primer podatki o avtentikaciji upravljalca, ki je poslal zahtevek.

Po sprejemu zahtevka agent najprej preveri njegovo veljavnost. Preveri, ali je zahtevek razumljiv in pregleda varnostne parametre, ki zajemajo avtentikacijo upravljalca. Če zahtevek ni veljaven, agent javi napako, drugače pa obdela zahtevek in sestavi odziv, ki vsebuje:

- odzivno kodo, ki sporoča, ali je bil zahtevek uspešen ali ne, in v primeru, da ni bil uspešen, razlog zakaj,
- rezultat zahtevka (npr. zahtevano informacijo),
- dodatne informacije, kot je identifikator, da lahko upravljalec odziv poveže z zahtevkom.

V postopku izvajanja določene naloge nadzora si upravljalec in agent izmenjata več zahtevkov in odzivov. V splošnem morata biti število in frekvenca take izmenjave čim manjša, ne da bi s tem škodovali funkcionalnosti in odzivnosti nadzorne aplikacije. Najbolj pogosti zahtevki upravljalca so zahtevki po informaciji o stanju, konfiguracijski zahtevki, zahtevki o akcijah in zgodovini informacij [1].

4.2.3 Agent

Agent predstavlja kratek program, ki je nameščen v nadzorovanih sistemih, kot so strežniki, usmerjevalniki, delovne postaje, itd. Namestitev agenta je preprosta in običajno neodvisna od operacijskega sistema. Takoj po zagonu začne nabirati informacije o strojni in programski

opremi sistema ter jih po potrebi posreduje upravljalcu. Protokol, po katerem teče komunikacija med agentom in upravljalcem, se nahaja na aplikacijski plasti modela OSI in je specifičen od proizvajalca do proizvajalca. Agenti poročajo upravljalcu odzive na njegove zahtevke in ga proaktivno obveščajo o nepričakovanih dogodkih v napravi. Primer komunikacije lahko vidimo na Sliki 7. Da lahko nadzorovana naprava oziroma aplikacija komunicira z upravljalcem, mora imeti implementirano programsko opremo agenta. Agent temelji na treh glavnih komponenetah: vmesnik za upravljanje, proces agenta in MIB.

Vmesnik za upravljanje omogoča komunikacijo in definira protokol oziroma pravila izmenjave podatkov med upravljalcem in agentom. Omogoča vzpostavitev in prekinitev seje za nadzor naprave.

MIB je objektno orientirana podatkovna zbirka o vrednostih parametrov, ki jih v nadzorovani napravi nadziramo. Upravljalec izvaja poizvedbe o podatkih o nadzorovanih objetih v MIB, lahko pa jih tudi dodaja, spreminja in briše. Sama operacija med MIB in dejansko napravo pa prevaja proces agenta. Zahtevek upravljalca najprej potuje skozi vmesnik za upravljanje do MIB, kjer se ta prevede v operacijo za pridobitev registra, ki razkriva vrednost nadzorovanega objekta.

Interakcije agenta

Drugo veliko kategorijo komunikacije med upravljalcem in agentom predstavljajo dogodki. V tem primeru je agent tisti, ki pošlje sporočilo upravljalcu in ga s tem obvesti o določenem pojavu na nadzorovani napravi. To so lahko sporočila o napakah, doseženem pragu, spremembi konfiguracije, večkratni neuspešni prijavi uporabnika, kar lahko kaže na napad. Sporočila o dogodkih v kontekstu s protokolom SNMP, ki ga bomo obdelali v nadaljevanju, imenujemo tudi pasti (angl. *traps*).

V nasprotju z odzivom, ki ga agent pošlje po prejemu zahtevka, agent sam ugotovi, kdaj mora poslati sporočilo o dogodku. Dogodki so spontana, nenaročena sporočila, niso pa nepričakovana, saj agent dogodke pošilja točno določenemu upravljalcu, ki se na te dogodke »naroči«.

Obstaja več kategorij obvestil o dogodkih, ki so odvisni od vzroka pojavitve dogodka. Najpogostejše kategorije so:

- **alarmi** – nepričakovani dogodki, ki kažejo na stanje zaradi česar je potrebna upravljalčeva pozornost;
- **dogodki o spremembi konfiguracije** – dogodki, ki upravljalca obvestijo, da je prišlo do spremembe konfiguracije na napravi;

- **alarm o preseženem pragu** – obvestilo, ki sporoča, da je naprava dosegla predhodnje določeno vrednost nadzorovanega parametra (npr. visoka obremenitev CPE) in potrebuje pozornost upravljalca, da ne bi prišlo do resnejših problemov;
- **zapisovanje operativnih dogodkov** – običajni dogodki, ki se pojavijo ob delovanju naprave in sporočajo kaj se trenutno na njej dogaja. Ti dogodki se uporabijo pri analizi in statistiki delovanja naprave. Zapisujejo se dogodki operaterjeve aktivnosti (ti so pomembni z vidika varnosti saj omogočajo vpogled v zgodovino sprememb), dogodki sistemskih aktivnosti (te se uporabljajo ob morebitnem iskanju napak v delovanju sistema) in dogodki na omrežju in storitvah;
- **informativni dogodki** – vsi drugi dogodki.

Da je sporočilo o dogodku uporabno, mora vsebovati vsaj naslednje informacije:

- sistem, s katerega je bil poslan,
- čas pojavitve dogodka,
- tip dogodka zaradi katerega se je pojavil,
- podrobnejše informacije o dogodku.

4.3 Protokol SNMP za nadzor omrežja

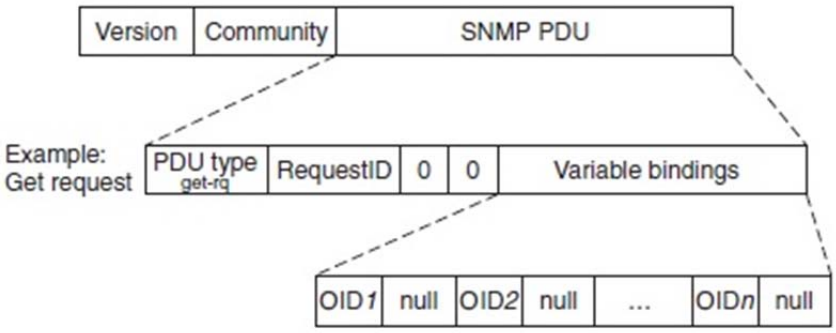
Protokol SMNP (angl. *Simple Network Management Protocol*) je najbolj razširjen protokol za nadzor omrežij. Poleg samega protokola zajema tudi specifikacije MIB, strukturo upravljalskih informacija (angl. *Structure of Management Information – SMI*) in arhitekturo in implementacijo agentov. Poznamo tri različice protokola SMNP, in sicer SMNPv1, SMNPv2c in SMNPv3. Razlike med različicami niso velike, zato je originalni SNMP še vedno zelo razširjen. Že samo ime protokola napeljuje na to, da je bil namenjen za preprost nadzor omrežja. Omogoča lahko implementacijo agenta na nadzorovani napravi, kompleksnost logike upravljalca pa je, zaradi preprostosti agenta, toliko večja. Ta preprostost implementacije agenta je povzročila širok sprejem protokola SNMP med proizvajalci, razvijalci programske opreme in uporabniki [1].

4.3.1 Struktura sporočila SNMP

Operacije SMNP med agenti in upravljalcem potekajo s pomočjo sporočil SNMP (Slika 8), ki sestojijo iz treh delov:

- različica protokola SNMP;
- niz skupnosti (angl. *Commnity string*), ki predstavlja nabor znakov, ki so nastavljeni na agentu in se morajo ujemati, da je zahteva sprejeta. Deluje kot neke vrste geslo, ki se prenaša nešifrirano, zato se SNMPv1 smatra kot varnostno pomanjkljiv protokol (problem varnosti je odpravljen s SNMPv2 in SNMPv3);

- podatkovna enota SNMP (SNMP PDU), to je kodirana operacija samega SNMP, ki vsebuje identificiran tip operacije in druge parametre (OID, vrednosti).



Slika 8: Struktura sporočila SNMP [1]

4.3.2 SNMPv2 in SNMPv3

Med tem, ko je protokol SNMP pridobil široko priljubljenost se je izkazalo, da ima tudi nekaj pomanjkljivosti. Je namreč zelo neučinkovit pri pridobivanju velikih količin upravljalskih informacij naenkrat. Druga pomanjkljivost pa je njegova šibka varnost, saj se vsi podatki prenašajo v tekstovni obliki brez šifriranja, kar ga naredi ranljivega. Zato se je uporabljal izključno za nadzor in ne toliko za upravljanje in oskrbovanje aplikacij.

SNMPv2 nam tako predstavi dve novi upravljalski operaciji ***get-bulk*** in ***inform***. Operacija *get-bulk*, ki je podobna zahtevi *get-next*, je novost v SNMPv2 in omogoča upravljalcu, da z enim zahtevkom prejme večji del tabele naenkrat. Zahtevek *get-bulk* (Slika 9) mora vsebovati tudi parametra, ki povesta koliko skalarjev in največ koliko vrednosti za posamezen stebrast objekt v zahtevi naj zahtevek vrne.

```
$ snmpbulkget -v2c -B 1 3 linux.ora.com public sysDescr ifInOctets ifOutOctets
system.sysDescr.0 = "Linux linux 2.2.5-15 #3 Thu May 27 19:33:18 EDT 1999 i686"
interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
```

Slika 9: Zahtevek get-bulk [13]

Ker je *get-bulk* SNMPv2 operacija moramo v parametru podati zahtevo, da uporabimo SMNPv2 PDU. To naredimo s parametrom *–v2c*. Parameter *B 1 3* nastavi, da prejmemo en skalar (v našem primeru *sysDescr*) in tri vrednosti iz tabele posameznega stebrastega objekta (*ifInOctets* in *ifOutOctets*).

Zahtevek *inform* se obnaša podobno kot *get* in *get-response*, le da zagotavlja mehanizem potrjevanja prejema odziva. Če na primer agent upravljalcu pošlje sporočilo *trap*, bo upravljalec agentu potrdil prejem sporočila *trap*.

SMNPv3 pa uvede najpomembnejšo novost, ki protokol SMNP naredi varen. Omogoča namreč enkripcijo upravljalskih sporočil in možnost avtentikacije pošiljatelja. Sedaj lahko agent, ki sprejme zahtevek, preveri avtentičnost upravljalca, ki je zahtevek poslal. Tako je protokol SNMP ponovno postal zanesljiv za nadziranje in upravljanje naprav in aplikacij.

5. Nadzor in primerjava primerkov strežnikov v oblaku

Zaradi velike rasti računalništva v oblaku v zadnjih letih je veliko število strežnikov nameščenih v podatkovnih centrih, ki ponujajo navidezne vire velikemu številu uporabnikov. Namesto nakupa strojne in programske opreme za poganjanje aplikacij se podjetja odločajo za najem navideznih virov v internetnem oblaku. Tako se je razvilo tudi več tehnik in platform za virtualizacijo, kot so VMware, Xen, KVM in VirtualBox. Za boljšo kvaliteto storitev morajo biti sistemski administratorji v podatkovnih centrih vedno obveščeni o trenutnem stanju njihovih strežnikov, porabe virov (obremenitev CPE, poraba diskovnega prostora, omrežnih virov,...) in delovanja aplikacij. Za zanesljivo upravljanje in nadzor nad heterogeno navidezno infrastrukturo (navideznimi računalniki v podatkovnem centru) je tako potreben standardiziran vmesnik [14].

Zaradi rasti priljubljenosti računalništva v oblaku se na trgu pojavlja vedno več ponudnikov teh storitev. Ker se ti ponudniki med seboj razlikujejo po uporabljeni infrastrukturi in načinu virtualizacije to privede do različno zmogljivih primerkov strežnikov v oblaku. Zato bomo v tej diplomski nalogi s pomočjo orodja CloudCmp tudi primerjali posamezne primerke strežnikov najbolj priljubljenih ponudnikov storitev v oblaku.

5.1 Orodje za nadzor omrežja

Na trgu lahko najdemo kar nekaj odprtokodnih aplikacij za nadzor omrežij. Za potrebe diplomske naloge smo se na podlagi raznih člankov na spletu odločili, da uporabimo orodje Zenoss.

Zenoss je vodilno odprtokodno orodje za nadzor omrežja. Preko preprostega spletnega vmesnika omogoča pregleden nadzor naše omrežne infrastrukture. Ko aplikacija odkrije novo napravo avtomatsko prične z nadzorom njenega stanja in delovanja.

Aplikacija kot privzeti protokol za nadzor uporablja običajen »brez-agentni« (angl. *agent-less*) protokol SNMP. Glavne značilnosti orodja Zenoss, ki so združene v enotnem in modernem spletnem vmesniku, so:

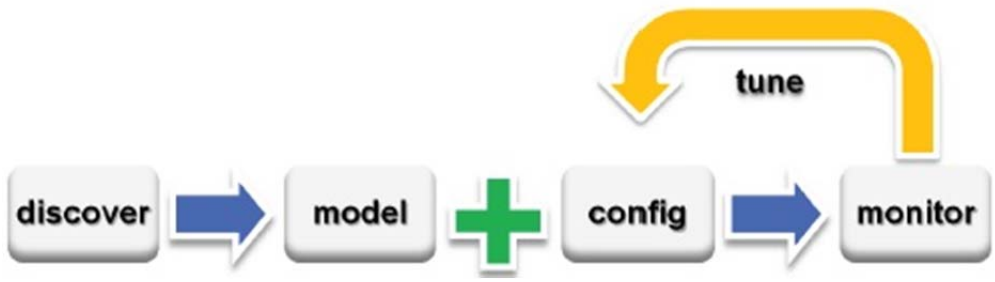
- odkrivanje in konfiguracija naprav,
- nadzor nad delovanjem in dostopnostjo naprav,
- upravljanje napak in dogodkov,
- opozarjanje na napake in njihovo odpravljanje,
- poročanje o delovanju naprav,
- brez-agentno zbiranje podatkov,

- normalizacija podatkov,
- podpora za več platform,
- razširljivost.

Aplikacija Zenoss Core je sestavljena iz štirih plasti:

- **uporabniška plast:**
  Predstavlja spletni uporabniški vmesnik, ki je zgrajen s pomočjo aplikacijskega spletnega okolja Zope. Preko spletnega vmesnika lahko nadziramo naše omrežje in naprave v njem, preverjamo status naprav. Vmesnik omogoča njihovo upravljanje, nadzor nad dogodki ter odziv nanje, upravljanje uporabnikov nadzornega sistema in izdelavo poročil o delovanju našega omrežja;
- **podatkovna plast:**
  Tu so v treh različnih podatkovnih bazah shranjeni podatki o konfiguraciji in podatki o dogodkih našega omrežja;
- **procesna plast:**
  Upravlja komunikacijo med podatkovno in zbirno plastjo. Izvaja periodična opravila sistema, kot tudi opravila, ki jih je zahteval uporabnik preko spletnega vmesnika;
- **zbirna plast:**
  Vključuje storitve, ki zajemajo podatke o nadzorovanem omrežju in z njimi oskrbuje podatkovno plast. Aplikacija za zajem podatkov iz naprav uporablja protokole SNMP, SSH in WMI. S pomočjo procesov v ozadju preverja status, dostopnost in zmogljivost nadzorovane infrastrukture.

Zenoss za nadzor infrastrukture IT uporablja modelno-usmerjen pristop. Odkrivanje naprav skupaj z dodeljevanjem modela nadzora omogoča avtomatski nadzor naprav v omrežju in s tem zmanjšuje probleme z vzdrževanjem sistema. Kot prikazuje Slika 10, se modelno usmerjen pristop nadzora prične z odkrivanjem, ki mu sledita dodeljevanje in konfiguracija modela nadzora, vse skupaj pa vodi k začetku nadzora naprave. Sam nadzor naprave se lahko med njenim delovanjem tudi nastavlja in prilagaja [15].



Slika 10: Modelno-usmerjen pristop nadzora [15].

5.2 Orodje za primerjavo oblakov

Primerjavo ponudnikov računalništva v javnem oblaku bomo naredili s pomočjo orodja CloudCmp [17], ki je bil razvit na univerzi Duke v ZDA. To orodje sistematično primerja zmogljivosti in ceno ponudnika računalništva v oblaku. CloudCmp meri elastičnost in zmogljivost računalništva v oblaku, nenehnega shranjevanja podatkov in mrežnega povezovanja storitev, ki jih oblak pounja, saj vse to neposredno vpliva na delovanje uporabnikovih aplikacij v oblaku [18].

5.3 Upravljanje z oblakom

Zasebni oblak bomo zgradili s pomočjo orodja OpenStack. To je odprtokodna platforma računalništva v oblaku, s pomočjo katere lahko ustvarimo in upravljamo javne in zasebne primerke strežnikov v oblaku. Platforma je nastala kot projekt NASE in Rackspace Cloud Hosting, ki sta želela ponuditi rešitve vseh vrst računalništva v oblaku širši javnosti na način, ki je preprost za implementacijo, zelo razširljiv, funkcijsko bogat in zmožen delovati na standardni strojni opremi. V našem primeru smo uporabili preprosto rešitev »vse v enem« (angl. *all-in-one*), ki je strojno manj zahtevna, saj imamo na voljo le en testni računalnik, ki ni primeren za resnejšo realizacijo zasebnega oblaka [5].

Za realizacijo javnega oblaka se bomo obrnili na tri najbolj priljubljene ponudnike računalništva v oblaku in njihove ponudbe IaaS:

- **Amazon Elastic Compute Cloud (EC2)** je centralni del Amazonove platforme računalništva v oblaku, Amazon Web Services (AWS). EC2 omogoča uporabnikom najem navideznih računalnikov, na katerih lahko poganjajo svoje aplikacije. EC2 preko spletnega portala omogoča uporabnikom veliko prednastavljenih primerkov navideznih strežnikov, ki jih uporabnik preprosto zažene in namesti željene aplikacije. Preko spletnega portala lahko uporabnik ustvari svoj primerek strežnika, ga zažene in po potrebi ustavi, zaradi česar se je uveljavil izraz elastičen (angl. *elastic*) [5].
- **Rackspace Cloud Servers** je storitev, ki ponuja infrastrukturo v oblaku. Podobno kot Amazon EC2 uporabniku omogoča namestitev več naprednih, visoko razpoložljivih navideznih strežnikov v nekaj minutah. Tudi Rackspace Cloude Servers omogoča upravljanje s primerki strežnikov preko spletnega portala [5].
- **Windows Azure** je Microsoftova platforma računalništva v oblaku, ki storitev PaaS in je skupaj s ponudbo storitve SaaS – Microsoft Online Services, del Microsoftove ponudbe računalništva v oblaku. Tako kot predhodnja ponudnika, tudi Windows Azure ponuja različne primerke navideznih srtežnikov, ki jih je moč upravljati preko spletnega portala [5].

5.4 Testno okolje

Testno okolje obsega osebni računalnik, ki do interneta dostopa preko usmerjevalnika za domačo rabo.

Naprave v testnem okolju:

- brezžični usmerjevalnik Thomson ST780,
- osebni prenosni računalnik HP Pavilion dm4, procesor Intel i5 2,53 GHz, 8 GB RAM.

Da lahko namestimo orodje za nadzor omrežja Zenoss in orodje za upravljanje oblaka OpenStack, potrebujemo operacijski sistem (OS) Linux. Pri tem smo si s pomočjo programa VirtualBox pomagali z virtualizacijo, ki nam omogoča, da delimo vire fizičnega računalnika med več različnih navideznih primerkov. Tako smo na osebnem prenosnem računalniku ustvarili 2 nova navidezna računalnika. Prvemu smo dodelili 2 CPE, 1,5 GB RAM, 20 GB prostora na disku. Nanj smo namestili OS Ubuntu Server 10.04 in aplikacijo za nadzor Zenoss. Poimenovali smo ga »ZenossSRV«. Drugemu smo dodelili 2 CPE, 2 GB RAM in 20 GB prostora na disku. Nanj smo namestili OS Ubuntu Server 12.04 in orodje za upravljanje oblaka OpenStack. Poimenovali smo ga »Essex«.

S pomočjo orodja OpenStack bomo izdelali primerek zasebnega oblaka z 512MB RAM in 1 navidezno CPE, ki smo ga poimenovali »*OpenStack Linux Server*«, na katerem bo nameščen OS Ubuntu Server 12.04 LTS.

Poleg primerka Linux v zasebnem oblaku bomo s pomočjo Amazon EC2, Windows Azure in Rackspace Hosting ustvarili primerke strežnikov v javnem oblaku. Po dva primerka pri vsakemu ponudniku, enega z OS Microsoft Windows Server 2008 R2 SP1 64-bit in enega z OS Linux Ubuntu Server 12.04 LTS. Ustvarili bomo naslednje primerke navideznih strežnikov v oblaku:

- Amazon EC2:
  - mikro (*t1.micro*) primerek Ubuntu Server 12.04 LTS poimenovan »*Amazon EC2 Linux*«, z 613MB RAM in z do 2 CPE,
  - mikro (*t1.micro*) primerek Microsoft Windows Server 2008 R2 SP1 poimenovan »*Amazon EC2 Windows*«, z 613MB RAM in z do 2 CPE;
- Microsof Azure Virtual Machine:
  - najmanjši (*ExtraSmall*) primerek Ubuntu Server 12.04 LTS poimenovan »*MS Azure Linux*«, z 768MB RAM in s skupnio (ang. *shared*) CPE,
  - najmanjši (*ExtraSmall*) primerek Microsoft Windows Server 2008 R2 SP1 poimenovan »*MS Azure Windows*«, z 768MB RAM in s skupno (ang. *shared*) CPE;

- Rackspace Hosting:
  - za OS Ubuntu Server 12.04 LTS smo uporabili primerek z 512MB RAM in 1 navidezno CPE, poimenovali smo ga *»Rackspace Linux«*,
  - za OS Windows 2008 R2 SP1 64-bit smo uporabili primerek z 1GB RAM in 1 navidezno CPE, poimenovali smo ga *»Rackspace Windows«*.

Na vse primerke strežnikov v oblaku (javnem in zasebenm) bomo namestili javanski spletni strežnik Apache Tomcat 6, s pomočjo katerega bomo poganjali orodje CloudCmp za ocenjevanje zmogljivosi strežnika v oblaku.

5.5 Potek preizkusa

5.5.1 Priprava strežniških primerkov

Da smo lahko primerke ustvarjenih strežnikov dodali v naš nadzorni sistem Zenoss in na njih konfigurirali spletni strežnik Apache Tomcat, smo morali, odvisno od OS, namestiti določene dodatke ter izvesti nekaj dodatnih nastavitev.

Za pripravo **primerkov strežnikov z OS Linux** smo morali z ukazom `apt-get install snmp snmpd` namestiti paket *snmp* za nadzor strežnika preko protokola SNMP in agenta *snmpd*, ki se odziva na zahteve SNMP, ki jih nadzorni sistem pošilja strežniku. Za pravilno delovanje smo morali nastaviti vsebino datoteke `/etc/default/snmpd`, tako da smo dovolili dostop do strežnika tudi iz drugih naprav. Dodati smo morali vrstico s parametri `SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid'`. Prav tako smo v vsebino datoteke `/etc/snmp/snmpd.conf` vnesli parametre, ki jih sistem Zenoss zahteva za komunikacijo z nadzorovanim strežnikom (Slika 11) :

```
#       sec.name source          community
com2sec notConfigUser  default   public

#       groupName       securityModel   securtyName
group   notConfigGroup  v1              notConfigUser
group   notConfigGroup  v2              notConfigUser

# Make at least snmpwalk -v 1 localhost -c public system fast again.
#       name    incl/excl       subtree mask(optional)
view    systemview      included        .1

#       group           context sec.model sec.level prefix read write notify
access  notConfigGroup  ""      any       noauth    exact  systemview none none
```

Slika 11: Parametri datoteke snmpd.conf [19]

Za namestitev paketa *cloudcmp.war*, ki smo ga ustvarili pred tem in ga bomo opisali v nadaljevanju, potrebujemo spletni stežnik Apacke Tomcat in dodatne pakete strežnika. Te

namestimo s pomočjo ukaza `apt-get install tomcat6 tomcat6-admin tomcat6-common tomcat6-user`. Po namestitvi potrebnih paketov moramo na strežniku Tomcat ustvariti tudi administratorskega uporabnika, s katerim bomo namestili paket cloudcmp.war. To storimo tako, da prekonfiguriramo datoteko `/etc/tomcat6/tomcat-users.xml` kamor vnesemo parametre, ki ustvarijo administratorja z geslom in določenimi vlogami (Slika 12) :

```
<role rolename="manager"/>
<role rolename="admin"/>
<user name="admin" password="secret_password" roles="manager,admin"/>
```

Slika 12: Parametri za konfiguriranje administratorskega uporabnika na spletnem strežniku Tomcat

S tem smo pripravili naše primerke strežnikov z OS Linux za nadaljevanje preizkusa.

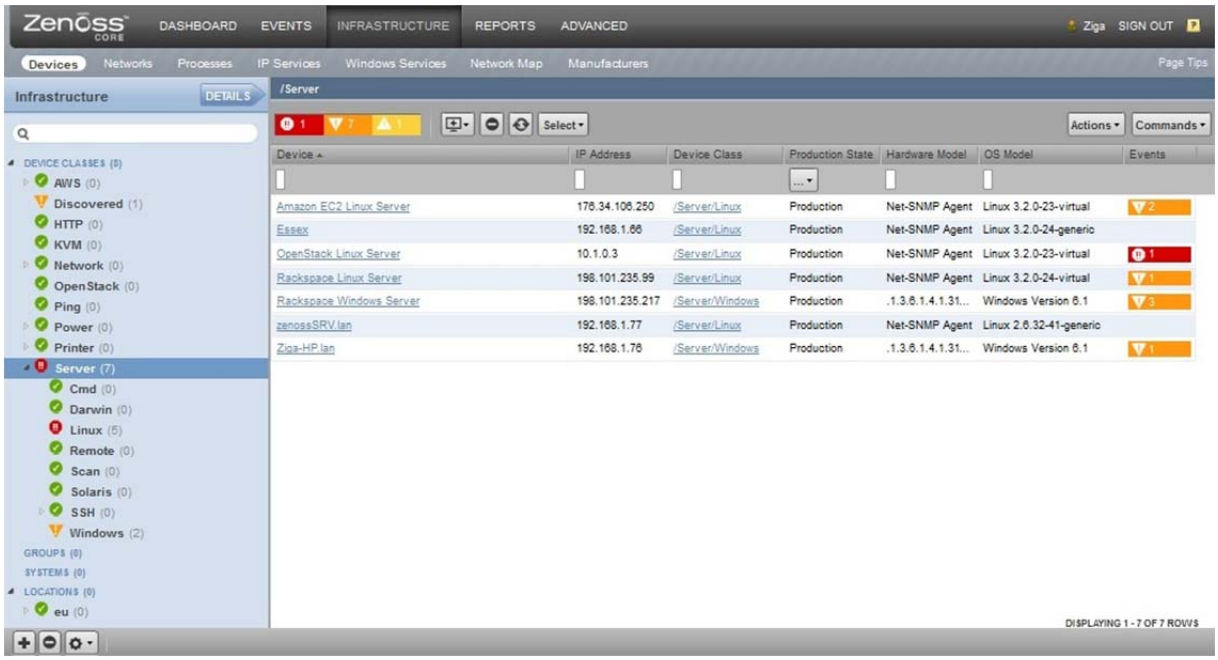
Primerki strežnikov z OS Windows od nas zahtevjo malce drugačno nastavljanje. Da omogočimo nadzor strežnika preko protokola SNMP, moramo z uporabo čarovnika za dodajanje Windows funkcij (angl. *Add Feature Wizard*) namestiti storitev SNMP. Ko opravimo namestitev, moramo v Upravitelju strežnika (angl. *Server Manager*) prekonfigurirati *SNMP Service*. To storimo tako, da ga poiščemo med servisi sistema in odpremo njegove nastavitve. V nastavitvah pod zavihkom Varnost (angl. *Security*) dodamo javno skupnost (angl. *public community*), ki ji dodelimo pravice samo za branje (angl. *read-only*) in označimo možnost sprejema paketov SNMP iz vseh gostiteljev (angl. *Accept SNMP packets from any hosts*). Poleg tega moramo v Windows požarni pregradi omogočiti vrata UDP 160, preko katerih komunicira protokol SNMP.

Za potrebe orodja CloudCmp pa moramo namestiti javansko okolje JRE in strežnik Apache Tomcat. Nastavitev administratorja za strežnik Tomcat v okolju Windows opravimo med postopkom namestitve.

Tako smo tudi naše primerke strežnikov z OS Windows pripravili za nadaljevanje našega preizkusa.

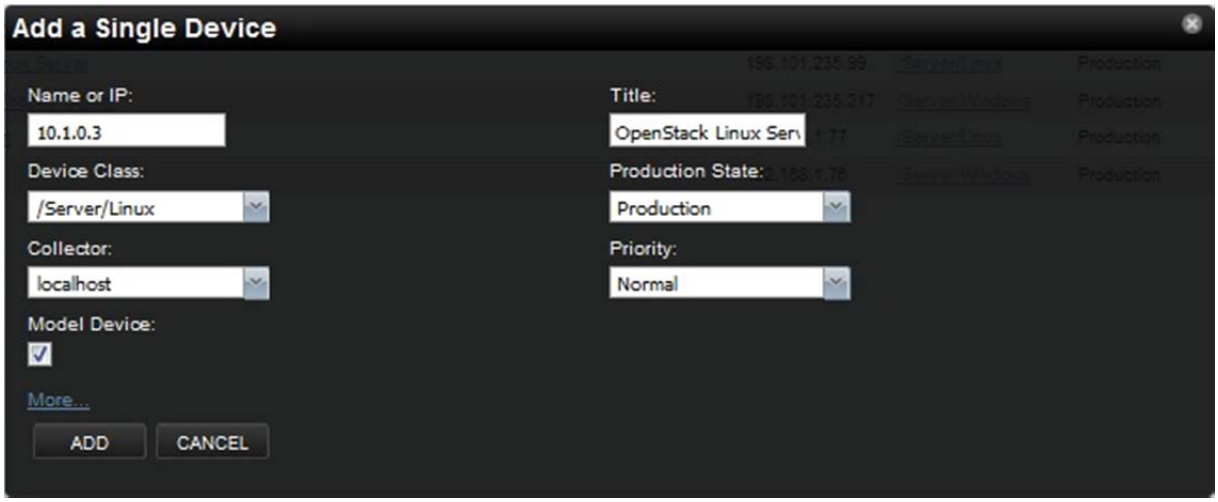
5.5.2 Dodajanje strežnikov v nadzorni sistem Zenoss

Da lahko nadziramo ustvarjene primerke strežnikov v oblaku, jih moramo dodati v nadzorni sistem Zenoss. Zenoss nam za upravljanje s sistemom za nadzor ponuja preprost uporabniški vmesnik preko spletnega brskalnika. V podmeniju *»Infrastructure«* (Slika 13) imamo seznam naših nadzorovanih naprav.



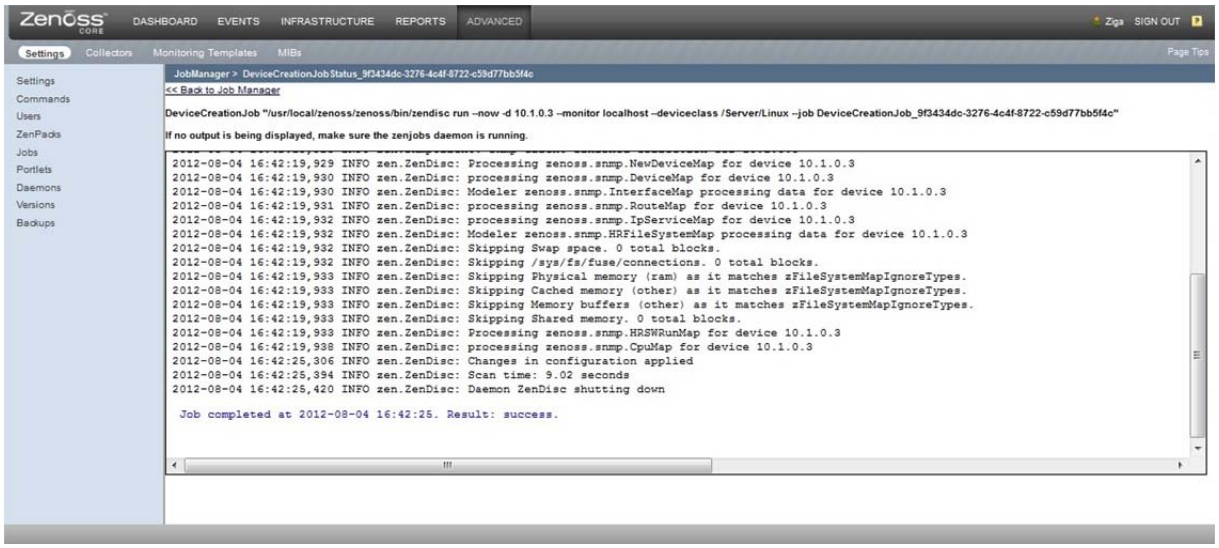
Slika 13: Seznam nadzorovanih naprav v sistemu za nadzor Zenoss

V tem podmeniju preko preprostega čarovnika dodamo željeno napravo (Slika 14). Vnesemo številko IP naprave, ime naprave in razred oz. tip naprave (angl. *Device Class*), ki določi prednastavljen vzorec (angl. *template*) parametrov, ki se nadzirajo.



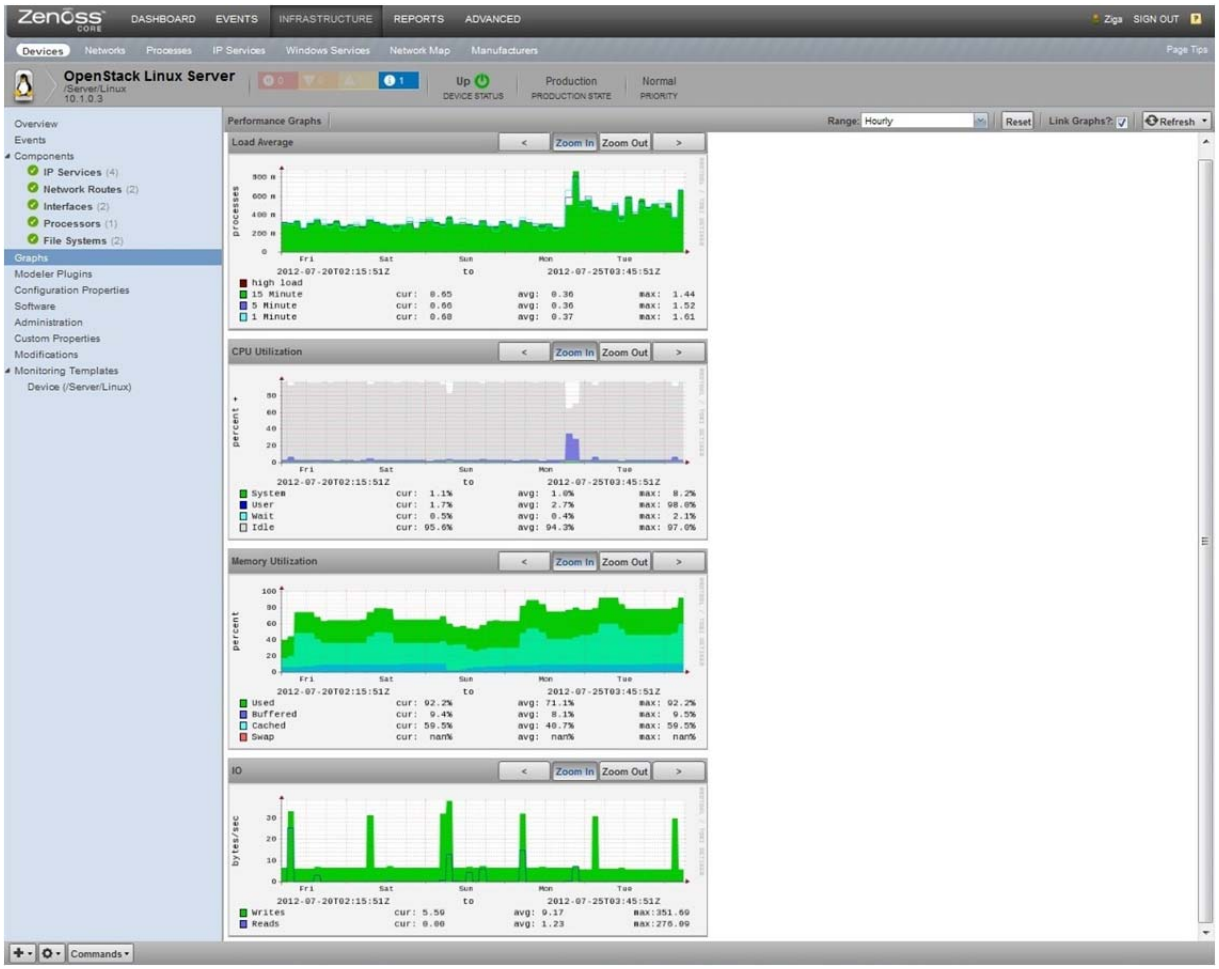
Slika 14: Dodajanje naprave v sistem za nadzor Zenoss.

Potek dodajanja naprave lahko spremljamo v dnevniku opravila (Slika 15).



Slika 15: Dnevnik opravila dodajanja naprave v sistemu za nadzor Zenoss.

Ko smo napravo uspešno dodali, jo lahko v spletnem vmesniku orodja Zenoss nadziramo, spremljamo stanje naprave preko grafov (Slika 16), pregledujemo opozorila, nastavljamo pravila za posamezne dogodke, itd.



Slika 16: Prikaz nadzorovane naprave v sistemu za nadzor Zenoss.

5.5.3 Primerjava zmogljivosti z orodjem CloudCmp

Preden smo namestili paket CloudCmp v spletni strežnik Tomcat na naše primerke strežnikov v oblaku, smo morali ustvariti paket *WAR* (angl. *Web application archive*). To je javanska arhivska datoteka, ki se jo uporablja za distribucijo zbirke javanskih strežniških datotek, javanskih servletov, datotek XML itd., ki skupaj tvorijo spletno aplikacijo [5].

Ta paket smo ustvarili na računalniku z OS Linux Ubuntu 12.04 LTS ob sledenju navodil, ki smo jih dobili na domači strani orodja *CloudCmp* [17]. Da smo lahko ustvarili paket *cloudcmp.war*, smo morali najprej namestiti Javo 1.6, spletni strežnik Apache Tomcat 6, orodje za ocenjevanje SPECjvm2008 in programski paket Apache Ant, ki je z ukazom `ant war` poskrbel za izdelavo paketa *cloudcmp.war*. Ta paket smo za tem preko spletnega portala *Tomcat Web Application Manager* namestili na vsakega izmed naših strežnikov.

S pomojčjo orodja CloudCmp bomo izvajali dve različni meritvi. Ena meritev, *crypto.rsa*, izvaja operacije, ki obremenjuje CPE in je hitrost izvedbe operacije odvisna od procesorske moči strežnika. Druga meritev, *memory.large*, pa izvaja opravilo, katerega hitrost izvedbe je odvisna od tega, kako hitro lahko pomnilnik dostavlja podatke procesorju. Obe meritvi med izvajanjem opravila merita čas, ki ga strežnik porabi za eno izvedbo opravila meritve.

Crypto.rsa med svojim izvajanjem opravlja kriptiranje in dekriptiranje podatkov s ključem dolžine 1024 bitov ter preverja enakost teh podatkov. Da so podatki o izvajanju relativni, in da se opravilo izvaja dovolj časa, v enkratni izvedbi opravila večkrat izvaja kriptiranje in dekriptiranje. Ob koncu izvajanja vrne čas, potreben za izvedbo.

Memory.large je preprost bralno-pisalni test pomnilnika. Ta alocira tabelo long[] z 1250000 elementi, kamor kopira zaporedne številke (kopiranje v spomin). Takoj za tem te podatke zopet prebere iz tabele. To pisanje in branje opravi stokrat, med tem si zapomni čas izvajanja, ki ga ob koncu izvajanja tudi izpiše.

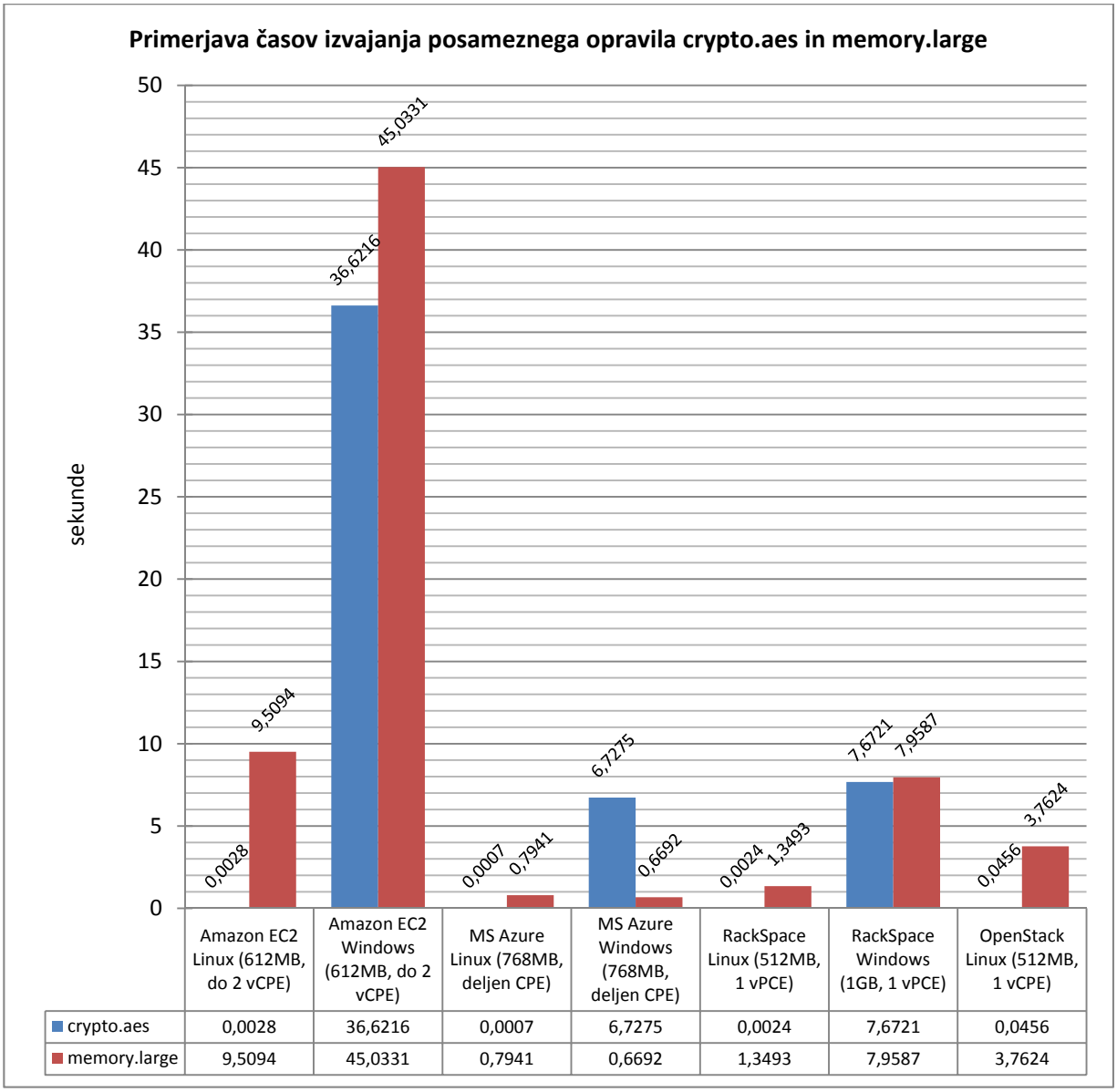
Na vseh strežniških primerkih smo s pomočjo zgoraj opisanega orodja *CloudCmp* opravili tisoč meritev za vsako izmed zgornjih opravil. Izvajanje meritev smo zagnali preko spletnega brskalnika. V naslovno vrstico smo vpisali IP naslov spletnega strežnika in vrata, na katerih strežnik posluša vhodni promet. Tem podatkom so sledili specifični parametri za zagon opravila.

Rezultati meritev in primerjava ponudnikov

V tem razdelku bomo predstavili rezultate, opravljene na strežniških primerkih v oblaku pri zgoraj omenjenih ponudnikih storitev IaaS (angl. *Infrastructure as a Service*): Amazon, Microsoft in Rackspace ter na primerku strežnika v zasebnem oblaku, ustvarjenim s pomočjo orodja OpenStack. Kot smo že omenili, smo z orodjem OpenStack ustvarili stežniški primerek z OS Linux Ubuntu Server 12.04 LTS, pri ponudnikih IaaS pa smo ustvarili po dva primerka

strežnikov. Enega z OS Linux Ubuntu Server 12.04 LTS, drugega pa z OS Windows Server 2008 R2. Na vseh primerkih smo opravili tisoč meritev izvajanja opravila, ki obremeni CPE (opravilo crypto.aes) in tisoč meritev izvajanja opravila, ki obremeni pomnilnik (opravilo memory.large) ter ob izvajanju teh meritev, merili čas izvajanja posameznega opravila. V spodnjih odstavkih bomo predstavili rezultate naših meritev, ki jih lahko vidimo na Grafu 1.

Kot vidimo na grafu, so primerki strežnikov z OS Linux za izvajanje posameznega opravila potrebovali občutno manj časa kot primerljiv primerek istega ponudnika IaaS z OS Windows. To lahko pripišemo manjši porabi strojnih virov sistema Linux v primerjavi s sistemom Windows.



Graf 1: Primerjava časov izvajanja posameznega opravila crypto.aes in memory.large.

V kolikor primerjamo samo primerke strežnikov z OS Linux, lahkko vidimo, da prav vsi primerki porabijo občutno manj časa za izvajanje opravila crypto.aes, ki obremenjuje CPE, kot za izvajanje opravila memory.large, katerega hitrost izvajanja je odvisna od hitrosti in kapacitete pomnilnika. Iz rezultatov je lepo razvidno, da je najbolj učinkovit primerek strežnika z OS Linux, primerek ustvarjen s pomočjo Microsoft Windows Azure. Ta primerek ima tudi največ pomnilnika med vsemi Linux strežniškimi primerki, vendar nima dodeljene svoje lastne navidezne CPE, temveč si jo deli z drugimi primerki te velikosti (extra small) pri ponudniku IaaS. Kot vidimo, so primerki strežnikov z OS Linux za izvajanje opravila crypto.aes potrebovali zelo malo časa, v povprečju nekaj manj kot 1,3 stotinke sekunde. Na drugi strani pa so za izvedbo opravila memory.large potrebovale veliko več. Najpočasnejši med vsemi je bil primerek strežnika, ustvarjen pri ponudniku Amazon, ki je za eno izvedbo opravila memory.large potreboval kar 9,5 sekunde, kar je skoraj 9 sekund počasneje od primerka, ki je bil ustvarjen pri Microsoftu, ki je s tem opravilom opravil najhitreje.

Ob pogledu na rezultate meritev, opravljenih na primerkih strežnikov z OS Windows, opazimo, da rezultati niso taki, da bi jih posplošili na vse ponudnike. Kot vidimo pri primerkih ustvarjenih pri Amazonu, je rezultat meritev podoben tistim, ki smo jih dobili pri primerkih z OS Linux. Izvajanje opravila crypto.aes je namreč trajalo dobrih 8 sekund manj kot izvajanje opravila memory.large. Obenem pa je bil ta primerek občutno najpočasnejši v primerjavi z ostalimi primerki strežnikov z OS Windows, vendar je imel dodeljenega najmanj pomnilnika med primerki z OS Windows. Ta razlika v hitrosti je ob primerjavi s primerki strežnikov z OS Linux še bolj očitna. Rezultati meritev na primerku, ustvarjenem s pomočjo Microsoft Windows Azure, so povsem drugačni od ostalih primerkov, saj je izvajanje opravila memory.large vzelo dobrih 6 sekund manj kot običajno hitrejše izvedeno oravilo crypto.aes, med tem ko je izvajanje teh dveh opravil pri primerku strežnika RackSpace z OS Windows povsem primerljivo a je bilo opravilo crypto.aes vseeno izvedeno slabe 3 stotinke sekunde hitreje kot opravilo memory.large. Tako lahko iz naših meritev zopret vidimo, da je primerek strežnika ustvarjen pri Microsoftu najučinkovitejši, med tem ko smo bili nad počasnostjo primerka z OS Windows ustvarjenim pri Amazonu presenečeni in hkrati razočarani, saj smo od pionirja računalništva v oblaku pričakovali zmogljivejši primerek navideznega strežnika.

Statistična primerjava rezultatov

Statistično bomo podobnost rezultatov opravljenih meritev preverili s pomočjo t-testa, ki primerja dve različni množici podatkov in z verjetnostjo (p) pove ali, sta si ti dve množici podobni ali ne. Verjetnost se pri t-testu giblje med 0 (malo verjetno) in 1 (zagotovo). Večja kot je verjetnost, bolj je verjetno, da sta si množici podobni in obratno. V našem primeru bomo za kritično vrednost podobnosti vzeli vrednost 0,05 (5%). To pomeni, da če bo vrednost verjetnosti večja ali enaka 0,05 sta si množici statistično enaki, oziroma, če bo vrednost verjetnosti manjša od 0,05 bosta množici statistično različni.

V spodnjih tabelah smo opravili t-test nad rezultati naših meritev crypto.aes in memory.large. Iz Tabele 2 lahko razberemo, da sta si pri izvajanju opravila crypto.aes statistično primerljiva samo primerka Amazon EC2 Linux in Rackspace Linux. Tudi čas, ki sta ga ta dva primerka potrebovala za enkratno izvedbo opravila crypto.aes, je zelo podoben. Iz Grafa 1 je že na prvi pogled razvidna podobnost, saj je primerek Amazon EC2 Linux za enkratno izvedbo opravila potreboval 0,0028 s, primerek Rackspace Linux pa 0,0024 s. Kljub na prvi pogled podobnemu času enkrane izvedbe opravila crypto.aes (Graf 1) med primerkoma MS Azure Windows (6,7275 s) in Rackspace Windows (7,6721 s), si ta dva primerka statistično nista podobna.

	Amazon EC2 Linux	Amazon EC2 Windows	MS Azure Linux	MS Azure Windows	RackSpace Linux	RackSpace Windows	OpenStack Linux
Amazon EC2 Linux		<0,001	<0,001	<0,001	0,139	<0,001	<0,001
Amazon EC2 Windows	<0,001		<0,001	<0,001	<0,001	<0,001	<0,001
MS Azure Linux	<0,001	<0,001		<0,001	<0,001	<0,001	<0,001
MS Azure Windows	<0,001	<0,001	<0,001		<0,001	<0,001	<0,001
RackSpace Linux	0,139	<0,001	<0,001	<0,001		<0,001	<0,001
RackSpace Windows	<0,001	<0,001	<0,001	<0,001	<0,001		<0,001
OpenStack Linux	<0,001	<0,001	<0,001	<0,001	<0,001	<0,001	

Tabela 2: t-test nad rezultati meritev crypto.aes.

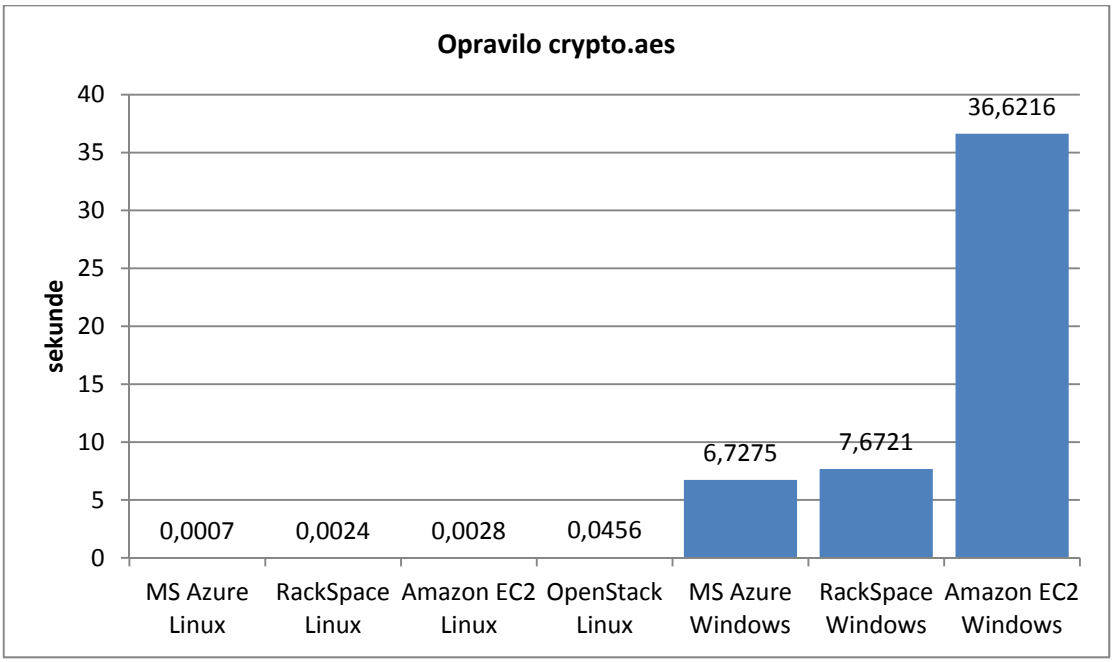
Kot lahko razberemo iz Tabele 3, ne moremo dokazati statistične podobnosti med primerki ob merjenu hitrosti enkratne izvedbe opravila memory.large. Čeprav bi lahko iz Grafa 1 na prvi pogled sklepali, da sta si primerljiva primerka Amazon EC2 Linux (9,5 s) in Rackspace Windows (7,9 s) ter primerki MS Azure Linux (0,79 s), MS Azure Windows (0,67 s) in RackSpace Linux (1,35 s).

	Amazon EC2 Linux	Amazon EC2 Windows	MS Azure Linux	MS Azure Windows	RackSpace Linux	RackSpace Windows	OpenStack Linux
Amazon EC2 Linux		<0,001	<0,001	<0,001	<0,001	0,004	<0,001
Amazon EC2 Windows	<0,001		<0,001	<0,001	<0,001	<0,001	<0,001
MS Azure Linux	<0,001	<0,001		<0,001	<0,001	<0,001	<0,001
MS Azure Windows	<0,001	<0,001	<0,001		<0,001	<0,001	<0,001
RackSpace Linux	<0,001	<0,001	<0,001	<0,001		<0,001	<0,001
RackSpace Windows	0,004	<0,001	<0,001	<0,001	<0,001		<0,001
OpenStack Linux	<0,001	<0,001	<0,001	<0,001	<0,001	<0,001	

Tabela 3: t-test nad rezultati meritev memory.large.

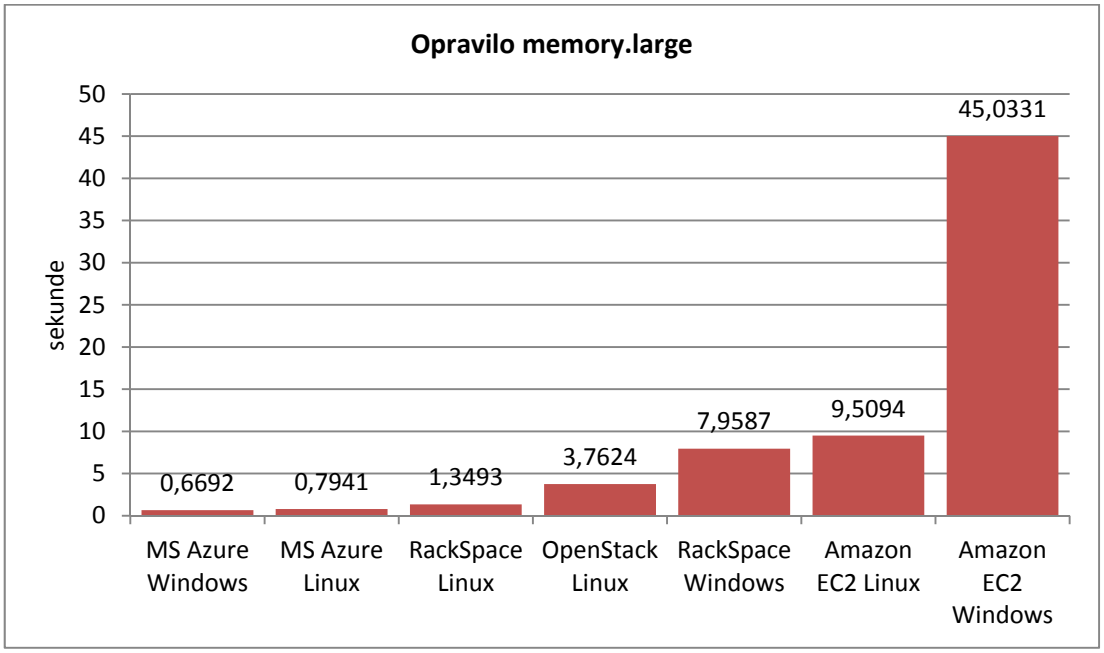
Vrednotenje rezultatov

Rezultate primerjav smo povzeli v spodnjih grafih. Na Grafu 2 smo razvrstili primerke od najhitrejšega do najpočasnejšega ob izvajanju opravila crypto.aes, ki obremeni CPE. Kot vidimo, so primerki z OS Linux občutno hitrejši od tistih z OS Windows. Najbolje se je odrezal primerek MS Azure Linux, najslabše pa primerek Amazon EC2 Windows, ki je bil od najhitrejšega počasnejši za več kot 36 sekund. Z rdečo barva sta označena primerka Racksapce Linux in Amazon EC2 Linux, ki sta po hitrosti izvajanja opravila crypto.aes tudi statistično enaka, kar smo dokazali s pomočjo t-testa.



Graf 2: Čas izvajanja crypto.aes – primerki so razvrščeni od najhitrejšega do najpočasnejšega.

Podobno smo na Grafu 3 razvrstili primerke od najhitrejšega do najpočasnejšega ob izvajanju opravila memory.large, ki je odvisen od hitrosti delovanja pomnilika. Statistično si pri izvajanju opravila memory.large nista podobna nobena primerka.



Graf 3: Čas izvajanja memory.large – primerki so razvrščeni od najhitrejšega do najpočasnejšega.

Rezultate meritev in primerjave časov izvajanja smo ovrednotili v Tabeli 4, ki prikazuje seštevek točk, ki smo jih dodelili posameznim primerkom glede na hitrost izvedbe opravila. Točkovali smo hitrost izvedbe opravila, tako da smo najhitrejšemu dodelili 7 točk, najpočasnejšemu pa 1 točko. Tako smo na podlagi zgornjih dveh grafov ovrednotili

posamezne primerke glede na čas potreben za izvedbo enkratnega opravila crypto.aes in memory.large, sešteli dodeljene točke in jih razvrstili po doseženem skupnem številu točk. Po točkovanju se je najbolje odrezal primerek MS Azure Linux, ki je skupaj dosegel 13 točk. Nad 10 točk pa sta dosegla tudi primerka Rackspace Linux (11 točk) in MS Azure Windows, ki je dosegel točno 10 točk.

V Tabeli 5 smo na podlagi doseženih točk primerkov posameznega ponudnika ovrednotili tudi ponudnike IaaS. Tako lepo vidimo, da se je pri našem testiranju zmogljivosti primerkov strežnikov v oblaku z OS Linux in Windows najbolje odrezal ponudnik Microsoft s storitvijo Windows Azure, ki je dosegel 23 točk. Primerki, ustvarjeni pri tem ponudniku so bili najbolj konsistentni glede na izvajanje posameznega opravila ne glede na nameščen OS. Slabše, s 16 točkami, se je odrezal ponudnik Rackspace Hosting, še slabše, s samo 9 točkami pa se je odrezal Amazon. Primerki ustvarjeni s pomočjo Amazon EC2 sta imela tudi največjo razliko v času izvajanja posameznega opravila na priemerku z OS Linux v primerjavi s primerkom, na katerem je bil nameščen OS Windows, kar je tudo lepo vidno na Grafu 1.

Instanca \ Opravilo	crypto.aes(točke)	memory.large(točke)	Skupaj
MS Azure Linux	7	6	13
RackSpace Linux	6	5	11
MS Azure Windows	3	7	10
OpenStack Linux	4	4	8
Amazon EC2 Linux	5	2	7
RackSpace Windows	2	3	5
Amazon EC2 Windows	1	1	2

Tabela 4: Vrednotenje primerkov glede na hitrost izvedbe opravil.

Ponudnik \ OS	Linux	Windows	Skupaj
MS Azure	13	10	23
RackSpace	11	5	16
Amazon EC2	7	2	9

Tabela 5: Vrednotenje ponudnikov IaaS.

5.5.4 Odziv nadzornega sistema Zenoss na obremenitev strežniških primerkov

Nadzorni sistem Zenoss prične z nadzorom dodane naprave takoj, ko smo napravo preko naslova IP dodali v sistem in jo pravilno konfigurirali za komunikacijo preko protokola SNMP z nadzornim sistemom Zenoss. Preformančne podatke o nadzorovani napravi, ki so prikazani preko grafov, beleži v podatkovni bazi Round Robin (angl. *Round Robin Database*) v datotekah RRD, kamor v specifičnem formatu in določenem časovnem intervalu, zapisuje trenutno stanje nadzorovanih virov, kot so obremenitev CPE, pomnilnika, trdih diskov itd. Preden lahko razberemo določene podatke iz teh datotek, jih moramo pretvoriti v nam berljiv format, v našem primeru format XML, s pomočjo orodja RRDtool. To je visoko zmogljivo

odprtokodno orodje za beleženje in grafično vizualizacijo podatkov, ki se beležijo v časovnem intervalu, ki je zaradi svoje popularnosti preraslo v industrijski standard in ga uporabljajo skoraj vsi nadzorni sistemi podbni našemu, uporabljenem za potrebe diplomske naloge [5].

Datoteke RRD nadzorovanih naprav se shranjujejo in posodabljajo na strežniku, na katerem je je nameščen nadzorni sistem Zenoss. Najdemo jih v direktoriju `/usr/local/zenoss/zenoss/perf/Devices`. V tem direktoriju se za vsako novo dodano napravo ustvari svoja mapa z imenom naprave, tja pa se zapisujejo zapisi RRD za vsak vir naprave, ki se nadzoruje in grafično prikazuje v sistemu Zenoss.

Odziv nadzornega sistema Zenoss na naše obremenitve primerkov strežnikov med izvajanjem zgoraj opisanih opravil bomo predstavili na primeru primerkov z Rackspace Linux in Rackspace Windows.

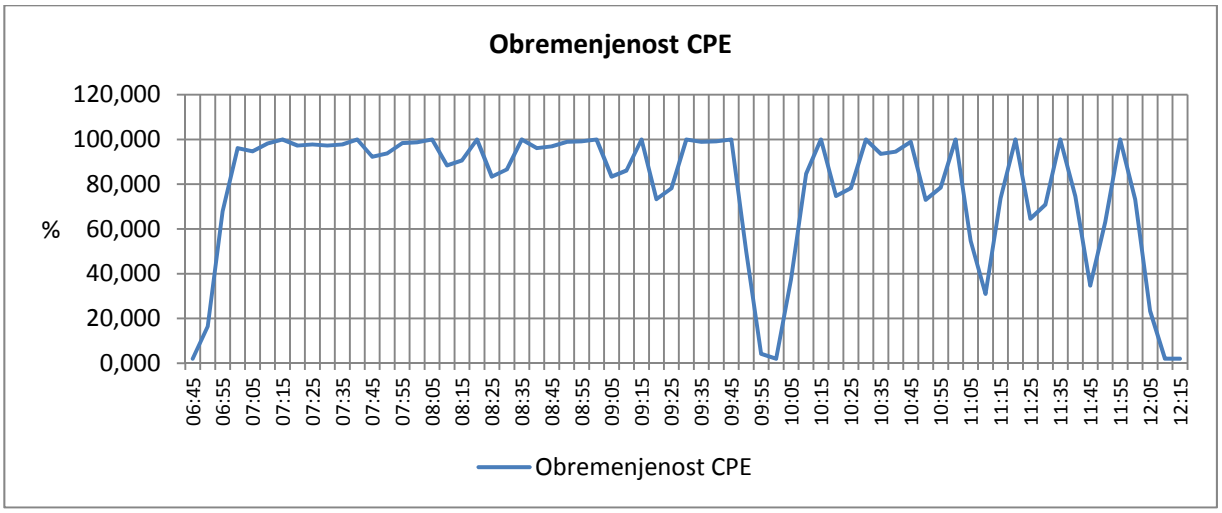
Najprej smo morali s pomočjo orodja RRDtool, datoteke RRD pretvoriti v nam berljiv format XML. Datoteke RRD smo našli na strežniku, na katerem je nameščen nadzorni sistem Zenoss, in sicer v spodnjih direktorijih:

- `/usr/local/zenoss/zenoss/perf/Devices/198.101.235.217` za primerek strežnika z OS Windows in,
- `/usr/local/zenoss/zenoss/perf/Devices/198.101.235.99` za primerek strežnika z OS Linux.

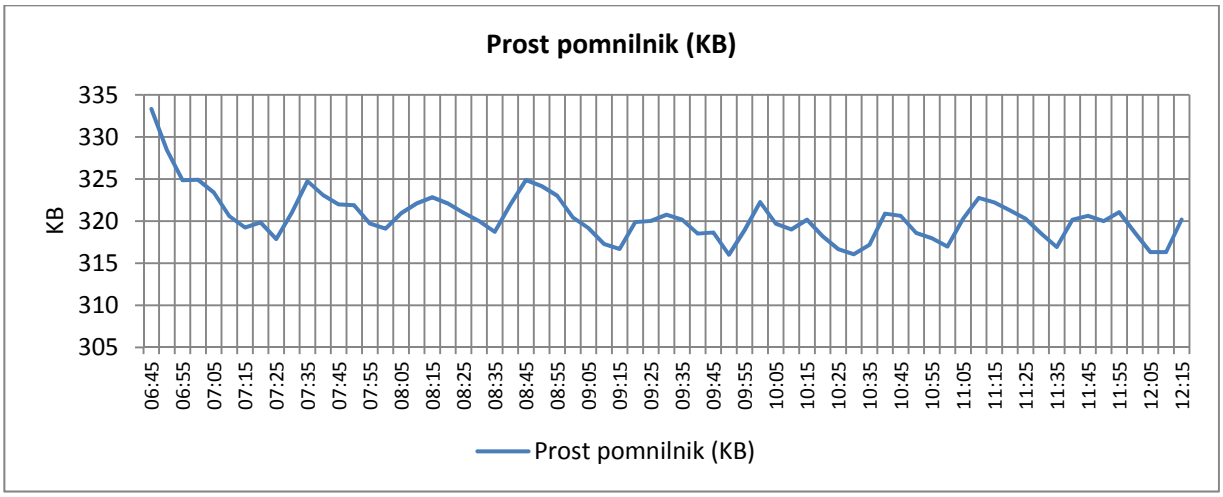
S pomočjo ukaza `rrdtool dump` smo vse zapise RRD pretvorili v datoteke XML, iz katerih smo lahko razbrali vrednosti posameznih virov nadzorovanih strežniških primerkov, ki jih bomo predstavili v nadaljevanju diplomske naloge. V našem primeru nas zanimajo zapisi, ki prikazujejo obremenjenost CPE in pomnilnika.

Strežniški primerek z OS Windows

Nadzorni sistem Zenoss podatke o obremenjenosti CPE shranjuje v datoteki `cpuPercentProcessorTime_cpuPercentProcessorTime.rrd`, o porabljenosti pomnilnika pa v datoteki `memoryAvailableKBytes_memoryAvailableKBytes.rrd`, ki jih najdemo v zgoraj omenjenem direktoriju. Ti dve datoteki smo pretvorili v nam berljivo XML obliko. Izsek, ki prikazuje zapise obremenitev CPE in pomnilnika v času našega izvajanja meritev, to je 2. avgusta 2012 med 6:45 in 12:15, vidimo v Prilogi 1 in 2. Vrednosti zapisov smo pretvorili v desetiški zapis in dobili grafični prikaz obremenitev kot ga vidimo na Grafu 4 in 5.



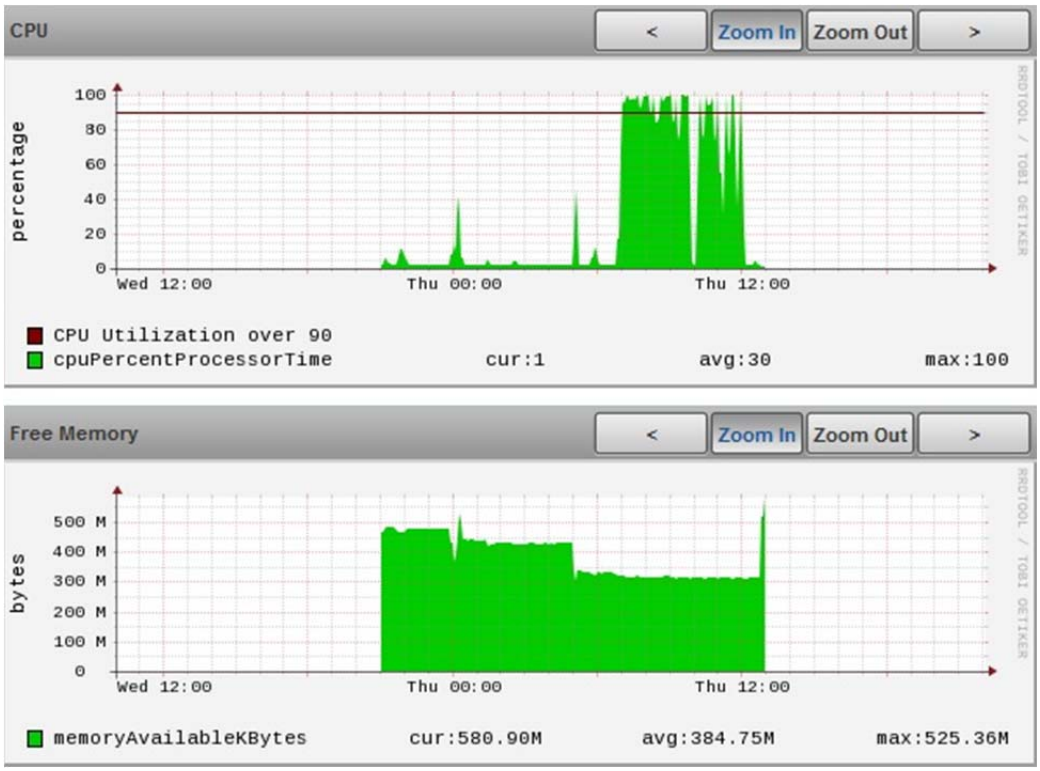
Graf 4: Graf obremenjenosti CPE primerka Rackspace Windows.



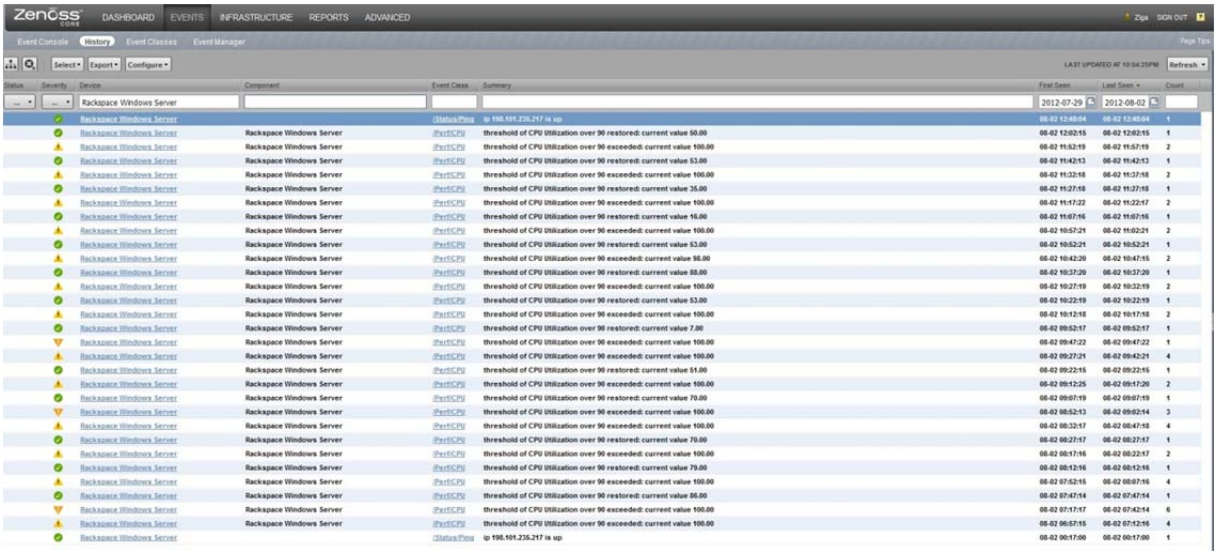
Graf 5: Graf porabljenosti pomnilnika primerka Rackspace Windows.

Podobna zgornjima grafoma, ki smo jih ustvarili s pomočjo vrednosti iz datoteke RRD, sta tudi grafa, ki ju generira nadzorni sistem Zenoss (Slika 17). Kot vidimo, sta CPE in pomnilnik obremenjena tekom celotnega izvajanja meritev, ne glede na to ali izvajamo crypto.aes, ki obremeni CPE ali memory.large, ki je odvisen od pomnilnika.

Ob izvajanju meritev na primerku strežnika Rackspace Windows smo v nadzornem sistemu Zenoss preizkusili tudi delovanje obvestil (angl. *alerts*). Prag sprožitve obvestila o prekomerni obremenjenosti CPE smo nastavili na mejo 90% obremenitve CPE. Mejno vrednost za opozorila lahko vidimo na zgornjem grafu Slike 17 v obliki rdeče črte. Vsakič, ko je obremenitev presegla mejo 90%, je sistem sprožil opozorilo o preseženi vrednosti (opozorilo s klicajem v rumenem trikotniku na Sliki 18), kasneje pa ob normalizaciji obremenjenosti CPE sistem zopet javi obvestilo (opozorilo s kljukico v zelenem krogu na Sliki 18), da se je stanje normaliziralo.



Slika 17: Grafični prikaz obremenitve primerka Rackspace Windows v sistemu Zenoss.



Slika 18: Prikaz obvestil o delovanju primerka Rackspace Windows v sistemu Zenoss.

Strežniški primerek z OS Linux

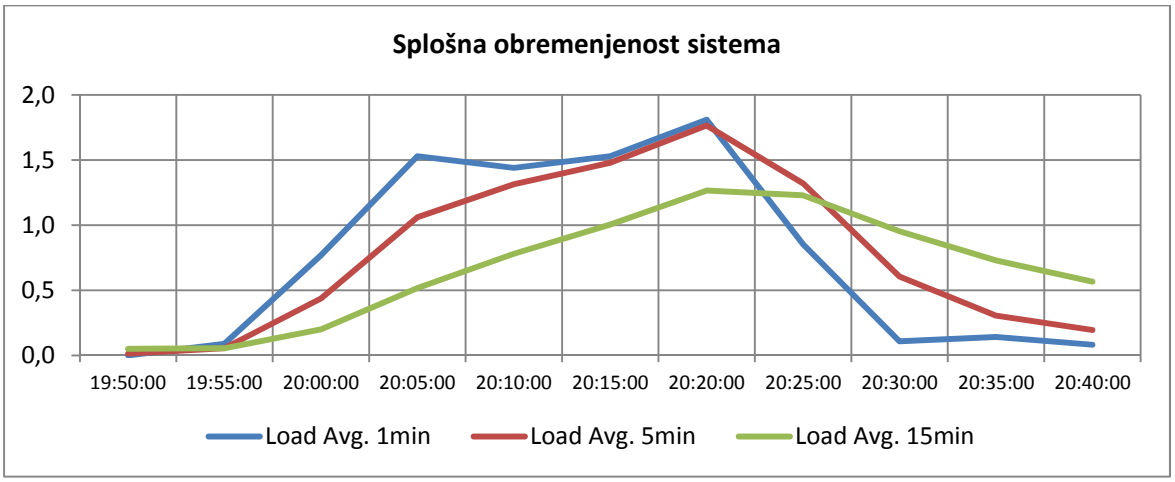
Zapisi o porabljenosti virov primerka Rackspace Linux shranjuje v zgornjem direktoriju. Za nas so bili zanimivi naslednji zapisi v datotekah RRD:

- zapisi o splošni obremenjenost sistema:
  - `laLoadInt1_laLoadInt1.rrd` (Priloga 3),
  - `laLoadInt5_laLoadInt5.rrd` (Priloga 4),
  - `laLoadInt15_laLoadInt15.rrd` (Priloga 5);

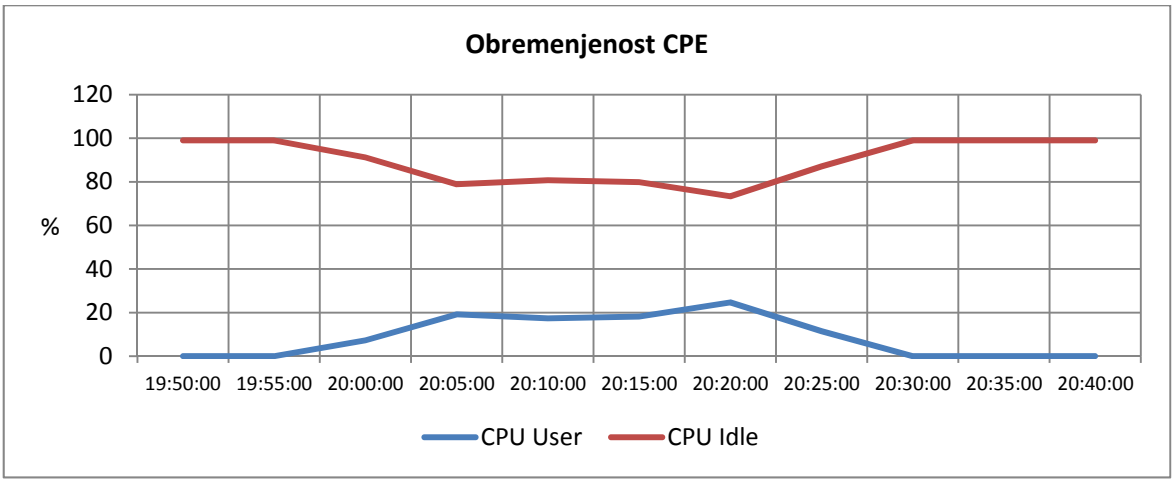
- zapisi o obremenjenosti CPE:
  - `ssCpuUser_ssCpuUser.rrd` (Priloga 6),
  - `ssCpuIdle_ssCpuIdle.rrd` (Priloga 7);
- zapisi o porabljenem pomnilniku:
  - `memAvailReal_memAvailReal.rrd` (Priloga 8).

Tudi te datoteke smo s pomočjo orodja RRDtool pretvorili v nam berljive datoteke XML, katerih izseke, ki prikazujejo obremenitev med našimi meritvami, lahko vidimo v zgoraj navedenih prilogah. Te vrednosti smo prav tako pretvorili v desetiški zapis in grafično prikazali obremenitev sistema na spodnjih grafih.

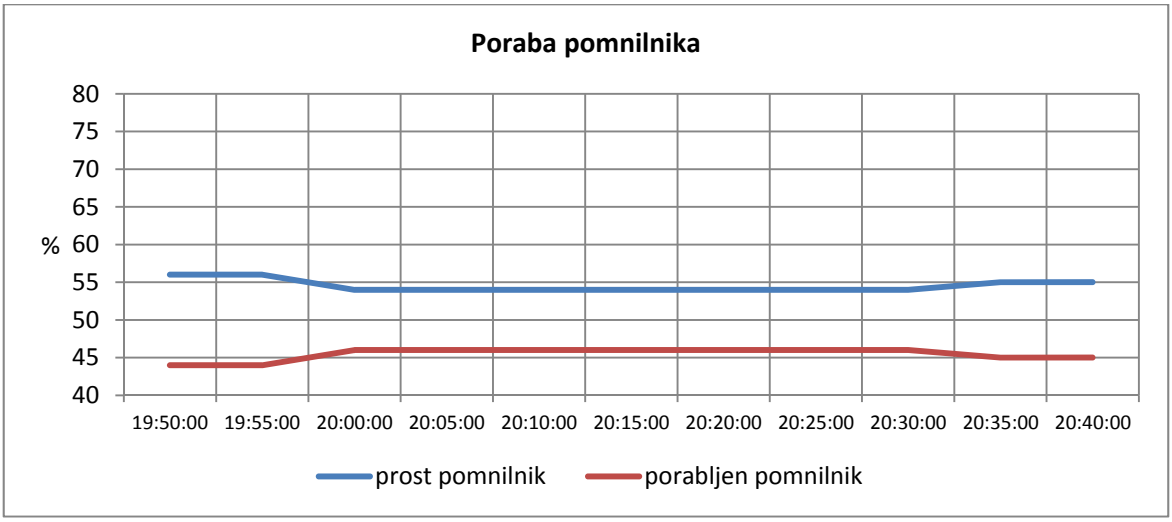
Na Grafu 6 se lepo vidi povečano obremenitev sistema s pričetkom izvanjanja meritev ob 19:55 (1.8.2012) in postopno zmanjševanje obremenitve od 20:20 naprej, ko smo z našimi meritvami zaključili. Tudi pri primerku z OS Linux na Grafu 6 opazimo, da je CPE obremenjen tekom celotnega izvajanja meritev, ne glede na to katero izmed opravil izvajamo, med tem ko se porabljenost pomnilnika praktično ne spremeni (Graf 7).



Graf 6: Graf splošne obremenjenosti primerka Rackspace Linux.

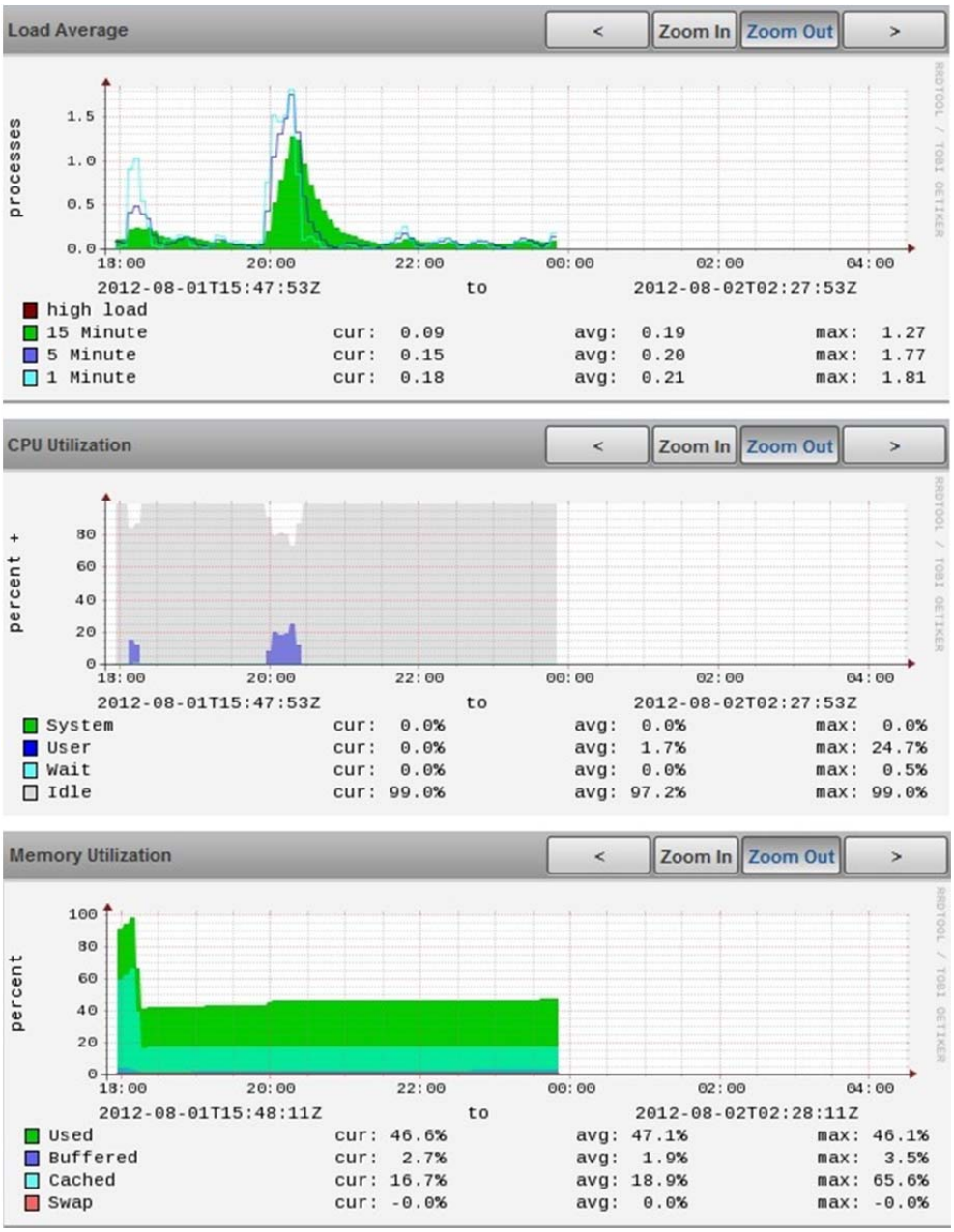


Graf 7: Graf obremenjenosti CPE primerka Rackspace Linux.



Graf 8: Graf porabe pomnilnika primerka Rackspace Linux.

Na Sliki 19 vidimo grafične prikaze, ki jih generira nadzorni sistem Zenoss in so praktično identični zgornjim grafom, izdelanih s pomočjo vrednosti, ki smo jih razbrali iz datotek RRD.



Slika 19: Grafični prikaz obremenitve primerka Rackspace Linux v sistemu Zenoss.

5.6 Omejitve preizkusa

V našem preizkusu smo uporabili prenosni računalnik, namenjen osebni uporabi, ki je bil v omrežje povezan preko preprostega domačega usmerjevalnika. Strojni viri, ki so nam bili na voljo s tem prenosnikom in domačim omrežjem, so po specifikaciji veliko šibkejši od tistih v resnih poslovnih omrežjih. Poleg tega smo te vire razdelili še med dva navidezna strežnika, enega za poganjanje nadzornega sistema Zenoss in drugega za poganjanje sistema OpenStack, s katerim smo ustvarili zasebni oblak.

Prav tako smo pri ponudnikih storitev v oblaku uporabili majhne in nezmogljive (z nizkimi strojnimi specifikacijami) strežniške primerke zaradi uporabe začasnih, preizkusnih in

brezplačnih računov, saj si nismo želeli nakopati visokih stroškov za najem navideznih strežnikov v oblaku. Tako Amazon kot Windows Azure ponujata časovno in strojno omejeno brezplačno obdobje uporabe njihovih storitev. Rackspace Hosting ne ponuja brezplačnega preizkusnega računa, vendar smo zaradi primerljivosti strežniških primerkov prav tako posegli po najmanj zmogljivih primerkih strežnikov, ki smo jih lahko uporabili za posamezen OS.

S tem, ko smo uporabljali majhne navidezne primerke in na njih poganjali preprosto aplikacije tudi nismo mogli preizkusiti vseh prednosti, ki jih ponujajo večji strežniki v oblaku, kot je na primer elastičnost. Prav tako smo se z našim preizkusom oprli samo na eno izmed storitev ponudnikov računalništva v oblaku. Pri ponudnikih smo najeli storitev IaaS – primerek navideznega strežnika, kamor smo namestili OS in našo aplikacijo za primerjavo zmogljivosti strežnikov v oblaku ter ostale potrebne dodatke za izvedbo našega preizkusa in primerjave.

Tako se moramo zavedati, da računalništvo v oblaku v večjih in resnejših okoljih ponuja še veliko več, kot smo imeli možnost preizkusiti med našim preizkusom, kjer smo zajeli le majhen del ponudbe, ki je bil za potrebe našega preizkus povsem zadosten.

6. Zaključek

Računalništvo v oblaku, je kot smo že večkrat omenili, vedno bolj priljubljena rešitev, ki se jo podjetja tudi vedno bolj poslužujejo. Namesto dragega nakupa, postavitve in nadaljnjega vzdrževanja lastnih strežnikov se raje odločijo za zakup strežnika v oblaku pri enem izmed mnogih ponudnikov. Ob vedno večjem številu teh ponudnikov se pojavi vprašanje, kateri izmed njih bi najlažje zadovoljil njihovim potrebam, saj vsak izmed njih ponuja različno zmogljive primerke po različnih cenah. Zato smo v diplomskem delu predstavili orodje CloudCmp, ki nam s svojimi meritvami pomaga odgovoriti na to vprašanje. Samo orodje ponuja še veliko več meritev, kot smo jih uporabili v diplomski nalogi, saj smo se oprli samo na preformančne meritve dveh izmed najpomembnejših lastnosti strežnika, to sta procesorska moč in hitrost pomnilnika. Kljub temu, da smo uporabili zelo majhne primerke strežnikov v oblaku, smo lahko na podlagi naših meritev izbrali ponudnika, ki je najbolje opravil naš preiskus. To je bil v našem primeru Microsoftov Windows Azure. Razočarani smo bili nad rezultati, ki smo jih dobili ob testiranju primerkov, ustvarjenih pri pionirju računalništva v oblaku, Amazon s storitvijo EC2, ki se je odrezal izredno slabo.

Poleg izbora najprimernejšega ponudnika storitev v oblaku je za administratorje infrastrukture IT zelo pomembno, da strežnike in aplikacije, ki jih prenesemo v oblak, lahko tudi nemoteno nadzira in spremlja njihovo delovanje, obremenjenost virov, itd. V diplomski nalogi smo tako pokazali, da se z nadzornim sistemom, v našem primeru je bil to odprtokodni nadzorni sistem Zenoss, da nadzirati primerke navideznih strežnikov v oblaku tako kot običajne strežnike, fizične ali virtualizirane, v običajnih sistemskih sobah podjetij. Zenoss na primer, v svoji odprtokodni skupnosti, ponuja možnost za razširitev svojega nadzornega sistema, ki je posebeno prilagojena za nadzor strežnikov v oblaku ustvarjenih s pomočjo Amazon EC2, kjer omogoča specifičan nadzor teh strežnikov.

Kot vse kaže, je prihodnost zagotovo v računalništvu v oblaku, kamor bodo podjetja zagotovo selila vse več strežnikov in aplikacij podpornih procesov, ki ne vsebujejo občutljivih poslovnih podatkov. Prav zaradi teh tajnih poslovnih in osebnih podatkov, za katere veljajo stroga varnostna pravila, pa se ne bomo povsem izognili lastnim strežnikom v sistemskih sobah podjetij. Zato menimo, da se bodo podjetja najpogosteje posluževala hibridnega modela oblaka, ki jih bodo IT administratorji želeli nadzirati kot običajne strežnike in aplikacije. S tem bo informacijska infrastruktura postala obsežnejša in kompleksnejša ter jo bo težje nadzirati, vendar menimo, da se bodo proizvajalci aplikacij za nadzor IT tehnologije vedno bolje prilagodili za nadzor strežnikov in aplikacij v oblaku, tako kot so se pred tem, s prihodom virtualizacije, prilagodili za nadzor navideznih strežnikov.

Literatura in viri

[1] A. Clemm, »Network Management Fundamentals«, Indianapolis: Cisco Press, 2007, /1. poglavje

[2] W. Stallings, »SNMP, SNMPv2, SNMPv3, and RMON1 and 2, - 3rd edition«, Massachusetts: Addison Wesley Longman, 1999, /1. poglavje

[3] D. Reynolds, E. Wright, »Practical TCP/IP and Enthernet Networking«, Burlington: Elsevier, /2. poglavje

[4] (2011) Internet in serija protokolov TCP/IP. Dostopno na: http://www.gimvic.org/predmeti/informatika/gradiva/internetni_protokoli.doc/

[5] (2012) Spletna enciklopedija *Wikipedia.* Iskalni nizi: Cloud computing, Network management model, Računalništvo v oblaku, Network management, FCAPS, SNMP, WAR, Amazon EC2, Rackspace, Windows Azure. Dostopno na: http://www.wikipedia.org/

[6] (2012) Spletno mesto »ISACA.org«, »*Essential characteristics of Cloud Computing*«. Dostopno na: http://www.isaca.org/Groups/Professional-English/cloud-computing/GroupDocuments/Essential%20characteristics%20of%20Cloud%20Computing.pdf

[7] (2012) D.Reeves, »*Cloud computing: Transforming IT*«, Burton Group. Dostopno na: http://net.educause.edu/ir/library/pdf/ECR0901.pdf

[8] (2012) A. Leites, »*Switching to the cloud? Advantages and disadvantages.*«, International computer concepts blog. Dostopno na: http://blog.icc-usa.com/2011/08/31/switching-to-the-cloud-advantages-and-disadvantages/

[9] (2012) Spletno mesto »InformIT«, M. Miller, »*Cloud Computing Pros and Cons for End User*«. Dostopno na: http://www.informit.com/articles/article.aspx?p=1324280&seqNum=1

[10] (2012) Spletno mesto »CloudDirectory«, »*Disadvantages of Cloud Computing*« in »*Advantages of Cloud computing*«. Dostopno na: http://www.clouddir.com/guides/disadvantages_of_cloud_computing.aspx http://www.clouddir.com/guides/advantages_of_cloud_computing.aspx

[11] (2012) Spletno mesto »*NetworkWorld*«, »*Staying ahead of cloud complexity*«. Dostopno na: http://www.networkworld.com/supp/2011/enterprise4/080811-ecs-cloud.html

[12] (2012) Spletno mesto »*NetworkWorld*«, »*Managing the Cloud*«. Dostopno na: http://www.networkworld.com/pdf/nw-gated/2011/0611-cloud-management.pdf

[13] (2012) Spletno mesto »*O'Reilly*«, »*Essential SNMP*«. Dostopno na: http://docstore.mik.ua/orelly/networking_2ndEd/snmp/index.htm

[14] (2012) »*SNMP-Based Monitoring of Heterogenous Virtual Infrastucture in Clouds*«. Dostopno na: http://www.copag.msu.ac.th/files/06077010.pdf

[15] (2012) Dokumentacija Zenoss Core. Dostopno na: http://community.zenoss.org/community/documentation/official_documentation/zenoss-guide/3.0-v03

[16] (2012) Spletno mesto *»ReadWriteWeb / Cloud«*, *»4 Tools for Assessing Cloud Performance«*. Dostopno na: http://www.readwriteweb.com/cloud/2010/08/tools-for-assessing-cloud-perf.php

[17] (2012) Spletno mesto *»CloudCMP.net«*. Dostopno na: http://cloudcmp.net/

[18] (2012) A. Li, X. Yang, S. Kandula, M. Zhang, »*CloudCmp: Comparing Public Cloud Providers*«. Dostopno na: http://www.cs.duke.edu/~angl/papers/imc10-cloudcmp.pdf

[19]: (2012) Spletno mesto *»Zenoss.org«*, *»Configuration of NetSNMP for use with Zenoss«*. Dostopno na: http://community.zenoss.org/docs/DOC-2502

Priloge

PRILOGA 1 - cpuPercentProcessorTime

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343905033 </lastupdate> <!-- 2012-08-02 12:57:13 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 1.0 </last_ds>
      <value> 1.3367573500e+02 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 5.5012127667e-01 </primary_value>
         <secondary_value> 4.4543619667e-01 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-02 06:50:00 CEST / 1343883000 --> <row><v> 1.6423055227e+01 </v></row>
         <!-- 2012-08-02 06:55:00 CEST / 1343883300 --> <row><v> 6.7874546080e+01 </v></row>
         <!-- 2012-08-02 07:00:00 CEST / 1343883600 --> <row><v> 9.6120299280e+01 </v></row>
         <!-- 2012-08-02 07:05:00 CEST / 1343883900 --> <row><v> 9.4663145350e+01 </v></row>
         <!-- 2012-08-02 07:10:00 CEST / 1343884200 --> <row><v> 9.8214775067e+01 </v></row>
         <!-- 2012-08-02 07:15:00 CEST / 1343884500 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 07:20:00 CEST / 1343884800 --> <row><v> 9.7227896617e+01 </v></row>
         <!-- 2012-08-02 07:25:00 CEST / 1343885100 --> <row><v> 9.7767041950e+01 </v></row>
         <!-- 2012-08-02 07:30:00 CEST / 1343885400 --> <row><v> 9.7230219167e+01 </v></row>
         <!-- 2012-08-02 07:35:00 CEST / 1343885700 --> <row><v> 9.7766718167e+01 </v></row>
         <!-- 2012-08-02 07:40:00 CEST / 1343886000 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 07:45:00 CEST / 1343886300 --> <row><v> 9.2254873147e+01 </v></row>
         <!-- 2012-08-02 07:50:00 CEST / 1343886600 --> <row><v> 9.3756058333e+01 </v></row>
         <!-- 2012-08-02 07:55:00 CEST / 1343886900 --> <row><v> 9.8338113800e+01 </v></row>
         <!-- 2012-08-02 08:00:00 CEST / 1343887200 --> <row><v> 9.8661743880e+01 </v></row>
         <!-- 2012-08-02 08:05:00 CEST / 1343887500 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 08:10:00 CEST / 1343887800 --> <row><v> 8.8367347920e+01 </v></row>
         <!-- 2012-08-02 08:15:00 CEST / 1343888100 --> <row><v> 9.0634149940e+01 </v></row>
         <!-- 2012-08-02 08:20:00 CEST / 1343888400 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 08:25:00 CEST / 1343888700 --> <row><v> 8.3361732800e+01 </v></row>
         <!-- 2012-08-02 08:30:00 CEST / 1343889000 --> <row><v> 8.6628291100e+01 </v></row>
         <!-- 2012-08-02 08:35:00 CEST / 1343889300 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 08:40:00 CEST / 1343889600 --> <row><v> 9.6117408793e+01 </v></row>
         <!-- 2012-08-02 08:45:00 CEST / 1343889900 --> <row><v> 9.6877108340e+01 </v></row>
         <!-- 2012-08-02 08:50:00 CEST / 1343890200 --> <row><v> 9.8892337607e+01 </v></row>
         <!-- 2012-08-02 08:55:00 CEST / 1343890500 --> <row><v> 9.9107264933e+01 </v></row>
         <!-- 2012-08-02 09:00:00 CEST / 1343890800 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 09:05:00 CEST / 1343891100 --> <row><v> 8.3378718000e+01 </v></row>
         <!-- 2012-08-02 09:10:00 CEST / 1343891400 --> <row><v> 8.6128907800e+01 </v></row>
         <!-- 2012-08-02 09:15:00 CEST / 1343891700 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 09:20:00 CEST / 1343892000 --> <row><v> 7.3324103877e+01 </v></row>
         <!-- 2012-08-02 09:25:00 CEST / 1343892300 --> <row><v> 7.8158062003e+01 </v></row>
         <!-- 2012-08-02 09:30:00 CEST / 1343892600 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 09:35:00 CEST / 1343892900 --> <row><v> 9.8911735507e+01 </v></row>
         <!-- 2012-08-02 09:40:00 CEST / 1343893200 --> <row><v> 9.9082867600e+01 </v></row>
         <!-- 2012-08-02 09:45:00 CEST / 1343893500 --> <row><v> 1.0000000000e+02 </v></row>
         <!-- 2012-08-02 09:50:00 CEST / 1343893800 --> <row><v> 4.9887992530e+01 </v></row>
         <!-- 2012-08-02 09:55:00 CEST / 1343894100 --> <row><v> 4.2259550667e+00 </v></row>
         <!-- 2012-08-02 10:00:00 CEST / 1343894400 --> <row><v> 2.0000000000e+00 </v></row>
         <!-- 2012-08-02 10:05:00 CEST / 1343894700 --> <row><v> 3.7444701013e+01 </v></row>
```

```
            <!-- 2012-08-02 10:10:00 CEST / 1343895000 --> <row><v> 8.4492960627e+01 </v></row>
            <!-- 2012-08-02 10:15:00 CEST / 1343895300 --> <row><v> 1.0000000000e+02 </v></row>
            <!-- 2012-08-02 10:20:00 CEST / 1343895600 --> <row><v> 7.4761615230e+01 </v></row>
            <!-- 2012-08-02 10:25:00 CEST / 1343895900 --> <row><v> 7.8207447803e+01 </v></row>
            <!-- 2012-08-02 10:30:00 CEST / 1343896200 --> <row><v> 1.0000000000e+02 </v></row>
            <!-- 2012-08-02 10:35:00 CEST / 1343896500 --> <row><v> 9.3571439680e+01 </v></row>
            <!-- 2012-08-02 10:40:00 CEST / 1343896800 --> <row><v> 9.4503628480e+01 </v></row>
            <!-- 2012-08-02 10:45:00 CEST / 1343897100 --> <row><v> 9.8909381233e+01 </v></row>
            <!-- 2012-08-02 10:50:00 CEST / 1343897400 --> <row><v> 7.3043223500e+01 </v></row>
            <!-- 2012-08-02 10:55:00 CEST / 1343897700 --> <row><v> 7.8502288517e+01 </v></row>
            <!-- 2012-08-02 11:00:00 CEST / 1343898000 --> <row><v> 1.0000000000e+02 </v></row>
            <!-- 2012-08-02 11:05:00 CEST / 1343898300 --> <row><v> 5.4717161560e+01 </v></row>
            <!-- 2012-08-02 11:10:00 CEST / 1343898600 --> <row><v> 3.0976871740e+01 </v></row>
            <!-- 2012-08-02 11:15:00 CEST / 1343898900 --> <row><v> 7.3665973590e+01 </v></row>
            <!-- 2012-08-02 11:20:00 CEST / 1343899200 --> <row><v> 1.0000000000e+02 </v></row>
            <!-- 2012-08-02 11:25:00 CEST / 1343899500 --> <row><v> 6.4601210167e+01 </v></row>
            <!-- 2012-08-02 11:30:00 CEST / 1343899800 --> <row><v> 7.0887531550e+01 </v></row>
            <!-- 2012-08-02 11:35:00 CEST / 1343900100 --> <row><v> 1.0000000000e+02 </v></row>
            <!-- 2012-08-02 11:40:00 CEST / 1343900400 --> <row><v> 7.4576169523e+01 </v></row>
            <!-- 2012-08-02 11:45:00 CEST / 1343900700 --> <row><v> 3.4704551290e+01 </v></row>
            <!-- 2012-08-02 11:50:00 CEST / 1343901000 --> <row><v> 6.3170568800e+01 </v></row>
            <!-- 2012-08-02 11:55:00 CEST / 1343901300 --> <row><v> 1.0000000000e+02 </v></row>
            <!-- 2012-08-02 12:00:00 CEST / 1343901600 --> <row><v> 7.3144093000e+01 </v></row>
            <!-- 2012-08-02 12:05:00 CEST / 1343901900 --> <row><v> 2.3399315040e+01 </v></row>
            <!-- 2012-08-02 12:10:00 CEST / 1343902200 --> <row><v> 2.0000000000e+00 </v></row>
            ...
            ...
            ...
        </database>
    </rra>
    ...
    ...
    ...
</rrd>
```

PRILOGA 2 - memoryAvailableKBytes

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343905033 </lastupdate> <!-- 2012-08-02 12:57:13 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 600028.0 </last_ds>
      <value> 8.0209351328e+07 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 5.9484178747e+05 </primary_value>
         <secondary_value> 5.2821274219e+05 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-02 06:45:00 CEST / 1343882700 --> <row><v> 3.3332435141e+05 </v></row>
         <!-- 2012-08-02 06:50:00 CEST / 1343883000 --> <row><v> 3.2844600933e+05 </v></row>
         <!-- 2012-08-02 06:55:00 CEST / 1343883300 --> <row><v> 3.2484957664e+05 </v></row>
         <!-- 2012-08-02 07:00:00 CEST / 1343883600 --> <row><v> 3.2489754417e+05 </v></row>
         <!-- 2012-08-02 07:05:00 CEST / 1343883900 --> <row><v> 3.2340009068e+05 </v></row>
         <!-- 2012-08-02 07:10:00 CEST / 1343884200 --> <row><v> 3.2058764773e+05 </v></row>
         <!-- 2012-08-02 07:15:00 CEST / 1343884500 --> <row><v> 3.1924363134e+05 </v></row>
         <!-- 2012-08-02 07:20:00 CEST / 1343884800 --> <row><v> 3.1985333239e+05 </v></row>
         <!-- 2012-08-02 07:25:00 CEST / 1343885100 --> <row><v> 3.1788077934e+05 </v></row>
         <!-- 2012-08-02 07:30:00 CEST / 1343885400 --> <row><v> 3.2098734759e+05 </v></row>
         <!-- 2012-08-02 07:35:00 CEST / 1343885700 --> <row><v> 3.2475562455e+05 </v></row>
         <!-- 2012-08-02 07:40:00 CEST / 1343886000 --> <row><v> 3.2310201928e+05 </v></row>
         <!-- 2012-08-02 07:45:00 CEST / 1343886300 --> <row><v> 3.2198290189e+05 </v></row>
         <!-- 2012-08-02 07:50:00 CEST / 1343886600 --> <row><v> 3.2187753485e+05 </v></row>
         <!-- 2012-08-02 07:55:00 CEST / 1343886900 --> <row><v> 3.1971768797e+05 </v></row>
         <!-- 2012-08-02 08:00:00 CEST / 1343887200 --> <row><v> 3.1909845643e+05 </v></row>
         <!-- 2012-08-02 08:05:00 CEST / 1343887500 --> <row><v> 3.2092587057e+05 </v></row>
         <!-- 2012-08-02 08:10:00 CEST / 1343887800 --> <row><v> 3.2210663489e+05 </v></row>
         <!-- 2012-08-02 08:15:00 CEST / 1343888100 --> <row><v> 3.2284087675e+05 </v></row>
         <!-- 2012-08-02 08:20:00 CEST / 1343888400 --> <row><v> 3.2209421448e+05 </v></row>
         <!-- 2012-08-02 08:25:00 CEST / 1343888700 --> <row><v> 3.2099143357e+05 </v></row>
         <!-- 2012-08-02 08:30:00 CEST / 1343889000 --> <row><v> 3.2000637024e+05 </v></row>
         <!-- 2012-08-02 08:35:00 CEST / 1343889300 --> <row><v> 3.1871750183e+05 </v></row>
         <!-- 2012-08-02 08:40:00 CEST / 1343889600 --> <row><v> 3.2197833386e+05 </v></row>
         <!-- 2012-08-02 08:45:00 CEST / 1343889900 --> <row><v> 3.2488965669e+05 </v></row>
         <!-- 2012-08-02 08:50:00 CEST / 1343890200 --> <row><v> 3.2415491129e+05 </v></row>
         <!-- 2012-08-02 08:55:00 CEST / 1343890500 --> <row><v> 3.2301857660e+05 </v></row>
         <!-- 2012-08-02 09:00:00 CEST / 1343890800 --> <row><v> 3.2043151906e+05 </v></row>
         <!-- 2012-08-02 09:05:00 CEST / 1343891100 --> <row><v> 3.1918944937e+05 </v></row>
         <!-- 2012-08-02 09:10:00 CEST / 1343891400 --> <row><v> 3.1728995982e+05 </v></row>
         <!-- 2012-08-02 09:15:00 CEST / 1343891700 --> <row><v> 3.1670256578e+05 </v></row>
         <!-- 2012-08-02 09:20:00 CEST / 1343892000 --> <row><v> 3.1988887596e+05 </v></row>
         <!-- 2012-08-02 09:25:00 CEST / 1343892300 --> <row><v> 3.2001476543e+05 </v></row>
         <!-- 2012-08-02 09:30:00 CEST / 1343892600 --> <row><v> 3.2076128348e+05 </v></row>
         <!-- 2012-08-02 09:35:00 CEST / 1343892900 --> <row><v> 3.2016912156e+05 </v></row>
         <!-- 2012-08-02 09:40:00 CEST / 1343893200 --> <row><v> 3.1850956730e+05 </v></row>
         <!-- 2012-08-02 09:45:00 CEST / 1343893500 --> <row><v> 3.1865860846e+05 </v></row>
         <!-- 2012-08-02 09:50:00 CEST / 1343893800 --> <row><v> 3.1601418117e+05 </v></row>
         <!-- 2012-08-02 09:55:00 CEST / 1343894100 --> <row><v> 3.1893705553e+05 </v></row>
         <!-- 2012-08-02 10:00:00 CEST / 1343894400 --> <row><v> 3.2225746356e+05 </v></row>
```

```
            <!-- 2012-08-02 10:05:00 CEST / 1343894700 --> <row><v> 3.1970549720e+05 </v></row>
            <!-- 2012-08-02 10:10:00 CEST / 1343895000 --> <row><v> 3.1900926971e+05 </v></row>
            <!-- 2012-08-02 10:15:00 CEST / 1343895300 --> <row><v> 3.2016256163e+05 </v></row>
            <!-- 2012-08-02 10:20:00 CEST / 1343895600 --> <row><v> 3.1820888990e+05 </v></row>
            <!-- 2012-08-02 10:25:00 CEST / 1343895900 --> <row><v> 3.1667469466e+05 </v></row>
            <!-- 2012-08-02 10:30:00 CEST / 1343896200 --> <row><v> 3.1606682390e+05 </v></row>
            <!-- 2012-08-02 10:35:00 CEST / 1343896500 --> <row><v> 3.1718542011e+05 </v></row>
            <!-- 2012-08-02 10:40:00 CEST / 1343896800 --> <row><v> 3.2089276293e+05 </v></row>
            <!-- 2012-08-02 10:45:00 CEST / 1343897100 --> <row><v> 3.2063179151e+05 </v></row>
            <!-- 2012-08-02 10:50:00 CEST / 1343897400 --> <row><v> 3.1858698038e+05 </v></row>
            <!-- 2012-08-02 10:55:00 CEST / 1343897700 --> <row><v> 3.1796035886e+05 </v></row>
            <!-- 2012-08-02 11:00:00 CEST / 1343898000 --> <row><v> 3.1697490277e+05 </v></row>
            <!-- 2012-08-02 11:05:00 CEST / 1343898300 --> <row><v> 3.2027775474e+05 </v></row>
            <!-- 2012-08-02 11:10:00 CEST / 1343898600 --> <row><v> 3.2275604625e+05 </v></row>
            <!-- 2012-08-02 11:15:00 CEST / 1343898900 --> <row><v> 3.2222507145e+05 </v></row>
            <!-- 2012-08-02 11:20:00 CEST / 1343899200 --> <row><v> 3.2127081680e+05 </v></row>
            <!-- 2012-08-02 11:25:00 CEST / 1343899500 --> <row><v> 3.2027932429e+05 </v></row>
            <!-- 2012-08-02 11:30:00 CEST / 1343899800 --> <row><v> 3.1851394307e+05 </v></row>
            <!-- 2012-08-02 11:35:00 CEST / 1343900100 --> <row><v> 3.1691598875e+05 </v></row>
            <!-- 2012-08-02 11:40:00 CEST / 1343900400 --> <row><v> 3.2017901356e+05 </v></row>
            <!-- 2012-08-02 11:45:00 CEST / 1343900700 --> <row><v> 3.2062908926e+05 </v></row>
            <!-- 2012-08-02 11:50:00 CEST / 1343901000 --> <row><v> 3.2000939513e+05 </v></row>
            <!-- 2012-08-02 11:55:00 CEST / 1343901300 --> <row><v> 3.2107148268e+05 </v></row>
            <!-- 2012-08-02 12:00:00 CEST / 1343901600 --> <row><v> 3.1862868555e+05 </v></row>
            <!-- 2012-08-02 12:05:00 CEST / 1343901900 --> <row><v> 3.1630382711e+05 </v></row>
            <!-- 2012-08-02 12:10:00 CEST / 1343902200 --> <row><v> 3.1630914511e+05 </v></row>
            <!-- 2012-08-02 12:15:00 CEST / 1343902500 --> <row><v> 3.2018255949e+05 </v></row>
            ...
            ...
            ...
        </database>
    </rra>
    ...
    ...
    ...
</rrd>
```

PRILOGA 3 - laLoadInt1

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343857632 </lastupdate> <!-- 2012-08-01 23:47:12 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 7.0 </last_ds>
      <value> 9.2671122600e+02 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 4.3521815600e+00 </primary_value>
         <secondary_value> 0.0000000000e+00 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-01 19:50:00 CEST / 1343843400 --> <row><v> 0.0000000000e+00 </v></row>
         <!-- 2012-08-01 19:55:00 CEST / 1343843700 --> <row><v> 8.9465550933e+00 </v></row>
         <!-- 2012-08-01 20:00:00 CEST / 1343844000 --> <row><v> 7.6389131320e+01 </v></row>
         <!-- 2012-08-01 20:05:00 CEST / 1343844300 --> <row><v> 1.5306697751e+02 </v></row>
         <!-- 2012-08-01 20:10:00 CEST / 1343844600 --> <row><v> 1.4468768232e+02 </v></row>
         <!-- 2012-08-01 20:15:00 CEST / 1343844900 --> <row><v> 1.5299063352e+02 </v></row>
         <!-- 2012-08-01 20:20:00 CEST / 1343845200 --> <row><v> 1.8123663857e+02 </v></row>
         <!-- 2012-08-01 20:25:00 CEST / 1343845500 --> <row><v> 8.5154278833e+01 </v></row>
         <!-- 2012-08-01 20:30:00 CEST / 1343845800 --> <row><v> 1.0795557483e+01 </v></row>
         <!-- 2012-08-01 20:35:00 CEST / 1343846100 --> <row><v> 1.4117154047e+01 </v></row>
         <!-- 2012-08-01 20:40:00 CEST / 1343846400 --> <row><v> 8.2911834000e+00 </v></row>
         ...
         ...
         ...
      </database>
   </rra>
   ...
   ...
   ...
</rrd>
```

PRILOGA 4 - laLoadInt5

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343857632 </lastupdate> <!-- 2012-08-01 23:47:12 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 9.0 </last_ds>
      <value> 1.1914885440e+03 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 6.7934807500e+00 </primary_value>
         <secondary_value> 0.0000000000e+00 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-01 19:50:00 CEST / 1343843400 --> <row><v> 1.4408583000e+00 </v></row>
         <!-- 2012-08-01 19:55:00 CEST / 1343843700 --> <row><v> 5.4732734667e+00 </v></row>
         <!-- 2012-08-01 20:00:00 CEST / 1343844000 --> <row><v> 4.3667773267e+01 </v></row>
         <!-- 2012-08-01 20:05:00 CEST / 1343844300 --> <row><v> 1.0565674822e+02 </v></row>
         <!-- 2012-08-01 20:10:00 CEST / 1343844600 --> <row><v> 1.3132255618e+02 </v></row>
         <!-- 2012-08-01 20:15:00 CEST / 1343844900 --> <row><v> 1.4789321248e+02 </v></row>
         <!-- 2012-08-01 20:20:00 CEST / 1343845200 --> <row><v> 1.7653813721e+02 </v></row>
         <!-- 2012-08-01 20:25:00 CEST / 1343845500 --> <row><v> 1.3208820833e+02 </v></row>
         <!-- 2012-08-01 20:30:00 CEST / 1343845800 --> <row><v> 6.0603574230e+01 </v></row>
         <!-- 2012-08-01 20:35:00 CEST / 1343846100 --> <row><v> 3.0621356350e+01 </v></row>
         <!-- 2012-08-01 20:40:00 CEST / 1343846400 --> <row><v> 1.9527466667e+01 </v></row>
         ...
         ...
         ...
      </database>
   </rra>
   ...
   ...
   ...
</rrd>
```

PRILOGA 5 - laLoadInt15

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343857632 </lastupdate> <!-- 2012-08-01 23:47:12 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 6.0 </last_ds>
      <value> 7.9432844400e+02 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 6.0000000000e+00 </primary_value>
         <secondary_value> 0.0000000000e+00 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-01 19:50:00 CEST / 1343843400 --> <row><v> 5.0000000000e+00 </v></row>
         <!-- 2012-08-01 19:55:00 CEST / 1343843700 --> <row><v> 5.5591586667e+00 </v></row>
         <!-- 2012-08-01 20:00:00 CEST / 1343844000 --> <row><v> 1.9978921000e+01 </v></row>
         <!-- 2012-08-01 20:05:00 CEST / 1343844300 --> <row><v> 5.1682221520e+01 </v></row>
         <!-- 2012-08-01 20:10:00 CEST / 1343844600 --> <row><v> 7.8064643480e+01 </v></row>
         <!-- 2012-08-01 20:15:00 CEST / 1343844900 --> <row><v> 1.0053821980e+02 </v></row>
         <!-- 2012-08-01 20:20:00 CEST / 1343845200 --> <row><v> 1.2653811771e+02 </v></row>
         <!-- 2012-08-01 20:25:00 CEST / 1343845500 --> <row><v> 1.2290383119e+02 </v></row>
         <!-- 2012-08-01 20:30:00 CEST / 1343845800 --> <row><v> 9.5344929427e+01 </v></row>
         <!-- 2012-08-01 20:35:00 CEST / 1343846100 --> <row><v> 7.2945651440e+01 </v></row>
         <!-- 2012-08-01 20:40:00 CEST / 1343846400 --> <row><v> 5.6614024000e+01 </v></row>
         <!-- 2012-08-01 20:45:00 CEST / 1343846700 --> <row><v> 4.2732103303e+01 </v></row>
         ...
         ...
         ...
      </database>
   </rra>
   ...
   ...
   ...
</rrd>
```

PRILOGA 6 - ssCpuUser

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343857632 </lastupdate> <!-- 2012-08-01 23:47:12 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 0.0 </last_ds>
      <value> 0.0000000000e+00 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 0.0000000000e+00 </primary_value>
         <secondary_value> 0.0000000000e+00 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-01 19:50:00 CEST / 1343843400 --> <row><v> 0.0000000000e+00 </v></row>
         <!-- 2012-08-01 19:55:00 CEST / 1343843700 --> <row><v> 0.0000000000e+00 </v></row>
         <!-- 2012-08-01 20:00:00 CEST / 1343844000 --> <row><v> 7.2691537967e+00 </v></row>
         <!-- 2012-08-01 20:05:00 CEST / 1343844300 --> <row><v> 1.9148832690e+01 </v></row>
         <!-- 2012-08-01 20:10:00 CEST / 1343844600 --> <row><v> 1.7290145280e+01 </v></row>
         <!-- 2012-08-01 20:15:00 CEST / 1343844900 --> <row><v> 1.8150841987e+01 </v></row>
         <!-- 2012-08-01 20:20:00 CEST / 1343845200 --> <row><v> 2.4677493500e+01 </v></row>
         <!-- 2012-08-01 20:25:00 CEST / 1343845500 --> <row><v> 1.1462828953e+01 </v></row>
         <!-- 2012-08-01 20:30:00 CEST / 1343845800 --> <row><v> 0.0000000000e+00 </v></row>
         <!-- 2012-08-01 20:35:00 CEST / 1343846100 --> <row><v> 0.0000000000e+00 </v></row>
         <!-- 2012-08-01 20:40:00 CEST / 1343846400 --> <row><v> 0.0000000000e+00 </v></row>
         ...
         ...
         ...
      </database>
   </rra>
   ...
   ...
   ...
</rrd>
```

PRILOGA 7 - ssCpuIdle

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343857632 </lastupdate> <!-- 2012-08-01 23:47:12 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 99.0 </last_ds>
      <value> 1.3105992933e+04 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 9.9000000000e+01 </primary_value>
         <secondary_value> 0.0000000000e+00 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-01 19:50:00 CEST / 1343843400 --> <row><v> 9.9000000000e+01 </v></row>
         <!-- 2012-08-01 19:55:00 CEST / 1343843700 --> <row><v> 9.9000000000e+01 </v></row>
         <!-- 2012-08-01 20:00:00 CEST / 1343844000 --> <row><v> 9.1171627327e+01 </v></row>
         <!-- 2012-08-01 20:05:00 CEST / 1343844300 --> <row><v> 7.8851145090e+01 </v></row>
         <!-- 2012-08-01 20:10:00 CEST / 1343844600 --> <row><v> 8.0709889720e+01 </v></row>
         <!-- 2012-08-01 20:15:00 CEST / 1343844900 --> <row><v> 7.9849133043e+01 </v></row>
         <!-- 2012-08-01 20:20:00 CEST / 1343845200 --> <row><v> 7.3322498650e+01 </v></row>
         <!-- 2012-08-01 20:25:00 CEST / 1343845500 --> <row><v> 8.7096341520e+01 </v></row>
         <!-- 2012-08-01 20:30:00 CEST / 1343845800 --> <row><v> 9.9000000000e+01 </v></row>
         <!-- 2012-08-01 20:35:00 CEST / 1343846100 --> <row><v> 9.9000000000e+01 </v></row>
         <!-- 2012-08-01 20:40:00 CEST / 1343846400 --> <row><v> 9.9000000000e+01 </v></row>
         ...
         ...
         ...
      </database>
   </rra>
   ...
   ...
   ...
</rrd>
```

PRILOGA 8 - memAvailReal

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
<!-- Round Robin Database Dump --><rrd>    <version> 0003 </version>
   <step> 300 </step> <!-- Seconds -->
   <lastupdate> 1343857632 </lastupdate> <!-- 2012-08-01 23:47:12 CEST -->

   <ds>
      <name> ds0 </name>
      <type> GAUGE </type>
      <minimal_heartbeat> 900 </minimal_heartbeat>
      <min> NaN </min>
      <max> NaN </max>

      <!-- PDP Status -->
      <last_ds> 267708.0 </last_ds>
      <value> 3.5440924874e+07 </value>
      <unknown_sec> 0 </unknown_sec>
   </ds>

<!-- Round Robin Archives -->    <rra>
      <cf> AVERAGE </cf>
      <pdp_per_row> 1 </pdp_per_row> <!-- 300 seconds -->

      <params>
      <xff> 5.0000000000e-01 </xff>
      </params>
      <cdp_prep>
         <ds>
         <primary_value> 2.6776272129e+05 </primary_value>
         <secondary_value> 0.0000000000e+00 </secondary_value>
         <value> NaN </value>
         <unknown_datapoints> 0 </unknown_datapoints>
         </ds>
      </cdp_prep>
      <database>
         ...
         ...
         ...
         <!-- 2012-08-01 19:50:00 CEST / 1343843400 --> <row><v> 2.8646017276e+05 </v></row>
         <!-- 2012-08-01 19:55:00 CEST / 1343843700 --> <row><v> 2.8646666405e+05 </v></row>
         <!-- 2012-08-01 20:00:00 CEST / 1343844000 --> <row><v> 2.7769580072e+05 </v></row>
         <!-- 2012-08-01 20:05:00 CEST / 1343844300 --> <row><v> 2.7095815558e+05 </v></row>
         <!-- 2012-08-01 20:10:00 CEST / 1343844600 --> <row><v> 2.7085599609e+05 </v></row>
         <!-- 2012-08-01 20:15:00 CEST / 1343844900 --> <row><v> 2.7048399086e+05 </v></row>
         <!-- 2012-08-01 20:20:00 CEST / 1343845200 --> <row><v> 2.7018132805e+05 </v></row>
         <!-- 2012-08-01 20:25:00 CEST / 1343845500 --> <row><v> 2.7021513458e+05 </v></row>
         <!-- 2012-08-01 20:30:00 CEST / 1343845800 --> <row><v> 2.7227818862e+05 </v></row>
         <!-- 2012-08-01 20:35:00 CEST / 1343846100 --> <row><v> 2.7374673612e+05 </v></row>
         <!-- 2012-08-01 20:40:00 CEST / 1343846400 --> <row><v> 2.7355335059e+05 </v></row>
         ...
         ...
         ...
      </database>
   </rra>
   ...
   ...
   ...
</rrd>
```