Migracija aplikacija na računarski oblak

Dragan Bojić, član, IEEE

***Sadržaj* — Računarski oblaci su novi trend korišćenja informacione tehnologije. Postojeći informacioni sistemi u perspektivi će pretrpeti prilagođavanja konceptu softvera kao servisa. Rad daje pregled potencijala i rizika povezanih sa novom tehnologijom kao i zajedničkih tehničkih karakteristika postojećih rešenja vodećih aktera u oblasti računarskih oblaka. Razmatraju se opcije u vezi postojećih informacionih sistema i predlaže metodologija postepene migracije na novu računarsku platformu.**

***Ključne reči* — Metodologija, Migracija aplikacija, Računarski oblaci.**

I. Uvod

RAČUNARSKI oblaci predstavljaju novi trend u informaciono-komunikacionim tehnologijama. Pojam računarskog oblaka je teško jednoznačno definisati pošto još evoluira, ali definicija američkog nacionalnog instituta za standarde [1] obuhvata najvažnije aspekte:

Računarski oblak je model korišćenja računara koji omogućava (mrežni) pristup po potrebi deljenom skupu prilagodljivih računarskih resursa (mrežama, serverima, skladištima podataka, aplikacijama i servisima) koji se mogu brzo obezbediti i osloboditi uz minimum upravljačkog napora i interakcije sa provajderom.

Modeli servisa računarskog oblaka obuhvataju softver kao servis (SaaS), platformu kao servis (PaaS) i infrastrukturu kao servis (IaaS), o čemu će naknadno biti više reči.

Modeli raspoređivanja obuhvataju privatne, javne /komunalne i hibridne oblake. U slučaju javnih oblaka, resursi koji se nude u vidu servisa dele se od strane više organizacija (ili najšire publike), pod kontrolom eksternog provajdera usluga. U slučaju privatnog oblaka, IT resursi su posvećeni jednoj organizaciji i nude se po potrebi. Hibridni oblak je mešavina privatnog i javnog oblaka, kojima se upravlja kao jedinstvenim entitetom, da bi se kapacitet servisa povećao prema potrebi.

Potencijali računarskih oblaka leže u efikasnijem korišćenju računarskih resursa nego u klasičnim pristupima, manji ukupan trošak posedovanja minimizovanjem investiranja u IT infrastrukturu, bez potrebe da se brine o servisnim zakrpama, upgrade-ima itd. Bitno se pojednostavljuje dostavljanje i primena IT servisa, što omogućava organizacijama da brže reaguju na poslovne zahteve. Kontrola nad obimom i kvalitetom usluge obezbeđuje se ugovorima (engl. Service Level Agreements). Korisnici imaju slobodu izbora servisa od bilo kod provajdera i pristup u svako doba i sa svakog mesta [2].

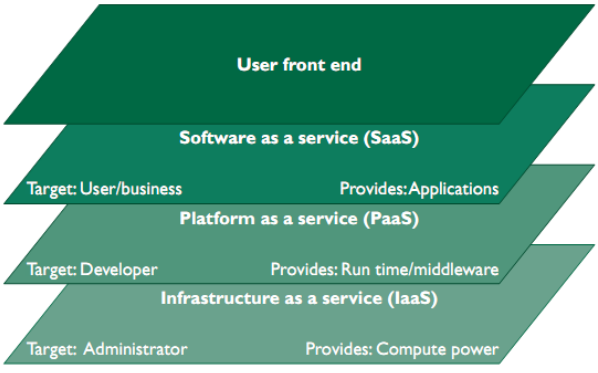
Rizici prelaska na oblak (na postojećem nivou razvoja) ogledaju se u tehničkim (migriranje postojećih složenih inf. sistema nije automatizovan niti jednostavan proces), pravni i administrativni (npr. u nekim oblicima poslovanja postoje ograničenja gde podaci mogu da se čuvaju i kome da budu dostupni), ekonomski (ukoliko se neracionalno koriste, oblaci mogu biti skuplji od klasičnog posedovanja inf. sistema), i vezani za sigurnost i dostupnost servisa (SLA ugovori tipično štite provajdera da u slučaju prekida usluge plaća penale samo u vrednosti usluge, a ne u vrednosti poslovne štete koja je time izazvana) [3].

U nastavku se prezentuju osnove arhitekture oblaka i aplikativnih interfejsa za implementaciju servisa. Predlaže se jedna konkretna višefazna strategija migracije postojećih informacionih sistema u novu arhitekturu, uzimajući u obzir poslovne potrebe i rizike vezane za dalje evoluiranje koncepta računarskih oblaka.

II. Arhitektura oblaka

Arhitektura oblaka je slojevita, od hardvera ka aplikacijama za krajnjeg korisnika. Prema nivou na kome se pružaju usluge korisnicima razlikuju se sledeći tipovi oblaka: infrastrukturni (IaaS – infrastructure as a service), platformski (PaaS) i aplikativno-softverski (SaaS), Sl.1.

Infrastrukturni servisi oblaka, poznati pod imenom "Infrastruktura kao servis" (IaaS), čine dostupnom računarsku infrastrukturu – tipično virtuelizovano okruženje, kao servis. Iz skupa servera, spoljne memorije i mrežnih komponenata uzimaju se resursi prema trenutnoj potrebi. Korisnik slobodno bira operativne sisteme i aplikacije koje će koristiti.



Sl. 1. Slojevita arhitektura računarskog oblaka.

Platformski oblaci uključuju pored hardvera i sloj softvera - integrisan operativni sistem, softver srednjeg nivoa (tipa aplikativnih servera) i razvojne alate. Korisnik može sam da bira i razvija aplikativni softver i konfiguraciju operativnog okruženja.

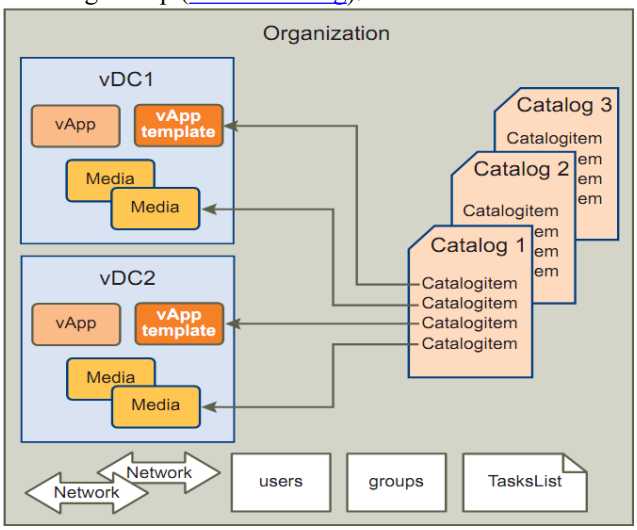
Dragan Bojić, Elektotehnički fakultet Univerziteta u Belgradu, Bul. kralja Aleksandra 73, 11120 Beograd, Srbija (tel: 381-11-3218346; email: bojic@etf.rs).

Softver kao servis omogućava da se kompletna aplikacija ponudi u vidu servisa po potrebi. Jedna instanca softverske aplikacije izvršava se na oblaku i opslužuje veći broj korisnika i klijentskih organizacija.

Jedna od bitnih karakteristika po kojoj se računarski oblaci razlikuju od postojećeg poslovnog računarstva je da je samu infrastrukturu moguće programirati. Umesto da direktno koriste fizičke servere, skladišni prostor i mrežne resurse za aplikacije, proizvođači softvera specifikuju kako se virtuelne komponente povezuju, uključujući smeštaj i upotebu slika virtuelnih mašina i podataka sa skladišnog prostora u oblaku. Specificiranje kada i kako se komponente upotrebljavaju radi se kroz aplikacioni programski interfejs (API) definisan od strane provajdera oblaka. Na žalost, API nije standardizovan, svaki provajder nudi sopstveni, što vezuje korisnika za određenog provajdera.

A. Programski interfejs

Programski model računarskog oblaka tipično definiše skup objekata sličnih onima prikazanim na slici 2, koja se odnosi na VMWare vCloud tehnologiju u klasi IaaS [4]. Postoje početni napori na standardizaciji programskih interfejsa oblaka, od strane DMTF Cloud Management Working Group (www.dmtf.org).



Sl. 2. Glavni koncepti upravljanja računarskim oblacima.

Virtuelni data centar (vDC) je alokacioni mehanizam za resurse kao što su mreža, skladišni prostor, procesori i memorija. Pored vDCova kao forme logičkog grupisanja resursa sa aspekta krajnjeg korisnika, često se uvode i zone kao vid grupisanja virtuelnih resursa sa aspekta provajdera oblaka (npr. resursi u jednoj zoni mogu biti koncentrisani na istoj geografskoj lokaciji sa određenim karakteristikama mrežnog propusnog opsega). Resursi su potpuno virtuelizovani i dodeljuju se na zahtev. Katalozi su kolekcije referenci na virtuelne sisteme i jedinice skladištenja, koji se stavljaju na raspolaganje za korišćenje, pod kontrolom administratora. Virtuelne aplikacije mogu obuhvatiti jednu ili više virtuelnih mašina. Nastaju procesom instancijacije odgovarajućih templejta, pri čemu se resursi opisani u templejtu mapiraju na resurse dostupne u data centru. U procesu instanciranja definišu se i operacioni parametri kao što su: način na koji su virtuelne mašine mrežno povezane među sobom i prema spoljnom svetu; ograničenje koliko resursa može da se angažuje; prava pristupa od strane korisnika.

Pod kontrolom virtuelne mašine tipično se izvršava operativni sistem (po izboru korisnika u IaaS varijanti, ili provajdera u PaaS varijanti), sa instaliranim softverom srednjeg sloja (web serveri, monitoring virtuelne mašine itd) i korisnička aplikacija koja obezbeđuje jednu ili više servisa. Servisi koji su dostupni upitom nad pomenutim objektima ili čitanjem kataloga tipično se implementiraju kao RESTful web servisi. Representational State Transfer (REST) je vrsta klijent-server arhitekture bez pamćenja stanja u kojoj je svaki servis identifikovan jedinstvenim URLom. URL se vraća korisniku na upit korišćenjem http protokola. Informacije o trenutnom stanju servisa i mogućim operacijama nad tim stanjem kodirane su u odgovoru nekim od Internet formata, kao što su html, xml ili JSON [5].

Na primer, ako korisnik Oracle računarskog oblaka želi da sazna informacije o dostupnim resursima (data centrima, zonama, aplikativnim templejtima), postaviće sledeći http GET upit na adresu oblaka koju definiše provajder [6]:

```
GET /
Host: cloudcompany.com
Authorization: Basic xxxxxxxxxx
Accept: application/vnd.com.oracle.cloud.Cloud+json
X-Cloud-Client-Specification-Version: 0.1
```

Mogući odgovor u JSON formi:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.com.oracle.cloud.Cloud+json
Content-Location: https://cloudcompany.com/
{
  "uri" : "https://cloudcompany.com/",
  "specification_version" : [ "0.8" ] ,
  "implementation_version" : "3.8RC2",
  "name" : "Oracle Cloud Service Provider",
  "description" : "Cloud services to the ABC industry…",
  "tags" : ["ABC", "Cloud", "Telecom", … ],/templates
  "zones" : {
     "uri" : "/123/zones",
     "type" : "Zone",
     "total" : "5" ,
     "elements" : [
 {"name" : "Europe West",
 "uri" : "/123/euzone"},
 … ]
 },
 "vdcs" : {
     "uri" : "/123/vdcs",
     "type" : "VDC",
     "total" : "1" ,
     "elements" : [
 {"name" : "Default Work Center",
 "uri" : "/123/vdcs/vdc232"}
 … ]
 },
 "service_templates" : {
     "uri" : "/templates/items/",
     "type" : "VMTemplate",
     "total" : "5" ,
     "elements" : [
 {"name" : "Oracle Peoplesoft Sales Demo",
 "uri" : "/templates/items/t123"},
 …]
 }
…
}
```

Provajderi infrastrukture oblaka nude različite programske biblioteke za pristup oblaku koje enkapsuliraju osnovni http protokol. Npr. Amazon EC2 nudi biblioteke za Javu, PHP, Python, Ruby i .NET programske jezike. Pored toga često postoje alati komandne linije i web interfejsi za upravljanje oblakom.

Važna karakteristika oblaka je da nudi automatsko balansiranje opterećenja i skaliranje korisnikovih servisa, često realizovano kroz merenje opterećenja i prema potrebi instanciranje novih primeraka servisa. To obično uvodi ograničenja u realizaciji poslovnih servisa, npr. da se ne pamti stanje klijenata koji se konektuju na servis u instanci, nego isključivo na skladišnom delu. Instance tipično međusobno mogu da komuniciraju i nekim protokolom slanja poruka osim web servisa.

B. Skladištenje podataka u oblacima

Virtuelne aplikacije normalno nemaju perzistentno skladište podataka, odnosno podaci se gube kada se izvršavanje instance okonča. Međutim, oblaci tipično nude više varijanti skladištenja, razmotrićemo to na primeru Amazon Web Servisa: Simple Storage Service (S3) je web servis kojem se pristupa RESTful http protokolom koji nudi smeštaj objekata od maksimum 5 gigabajta u kolekcije (buckets) i direktan pristup objektima (mogu dobiti i privremenu web adresu). S3 se odlikuje visokom pouzdanošću i skalabilnošću. Drugi smeštajni servis je Elastic Block Storage veličine do 1TB. Alocirani delovi se mogu asocirati sa jednom instancom virtuelne mašine u jednom trenutku i ponašaju se kao perzistentni diskovi. Na njima se mogu smeštati i klasične relacione baze (u tom slučaju, naravno, skalabilnost je ograničena). SimpleDB je skalabilno i pouzdano rešenje baze podataka nerelacionog tipa (leksikografsko indeksiranje) i sa odloženom konzistencijom (podaci se prosleđuju na više računara što ima konačno vremensko trajanje) [7].

III. METODOLOGIJA MIGRACIJE

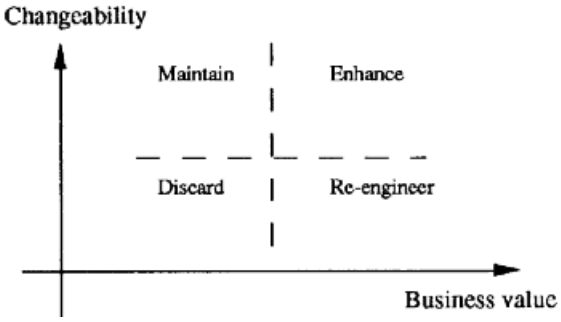
Ocena zastarelosti postojećeg informacionog sistema zavisi od perspektive posmatranja. Sa aspekta proizvođača softvera, u tu kategoriju spadaju sistemi koji nisu pogodni za održavanje i dorade. Sa aspekta tehnološkog razvoja, i sasvim funkcionalan i dobro održavan sistem može se smatrati tehnološki zastarelim u nekom trenutku. Sa aspekta poslovanja, zastareo je onaj sistem koji ne može da se prilagodi tempu promena u domenu poslovanja. Zastareli sistemi podložni su aktivnosti reinženjeringa tj. transformacije u novu formu. Ova aktivnost ima svoje ekonomske, organizacione, tehničke i pravne aspekte [8].

Reinženjering softvera često je povezan sa reinženjeringom poslovnog procesa tj. okruženja u kome se koristi softverski sistem. Radi se o ponovnom osmišljavanju i projektovanju poslovnog procesa u cilju poboljšanja kritičnih pokazatelja performansi kao što su troškovi, kvalitet, servisi i brzina.

A. Portfolio analiza

Opravdavanje reinženjeringa zahteva analizu postojećih aplikacija, procesa održavanja, i poslovne vrednosti aplikacija. Mora se osigurati povratak investicije – do kog nivoa će kvalitet softvera porasti, proces održavanja biti poboljšan, a poslovna vrednost povećana (analiza troškovi/dobit). Portfolio analiza služi za trijažu aplikacija kandidata za reinženjering prema njihovoj pogodnosti za reinženjering nasuprot poslovnoj vrednosti: nastavak održavanja, zamena novim, izmena, ili otpis (Sl.3).

U kontekstu migracije na oblak, nastavak održavanja bi mogao da znači da se aplikacija funkcionalno neizmenjena sa fizičke mašine (servera) prebaci na virtuelnu, koja se izvršava na fizičkom serveru korisnika ili kao deo privatnog oblaka. Zamena novom bi značila da se postojeća aplikacija izbacuje iz upotrebe, a da se umesto toga kupuje pravo na korišćenje novog servisa (prema SaaS konceptu), pri čemu treba rešiti pitanje migracije podataka.



Sl. 3. Portfolio analiza.

Ukoliko se donese odluka o reinženjeringu sistema, strateški se ova aktivnost može sprovesti nad svim komponentama sistema odjednom, ili u inkrementalnoj varijanti, gde se deo po deo sistema prebacuje na novo radno okruženje (oblak) a ostatak ostaje u postojećem radnom okruženju. Prednost migracije odjednom je u tome što se ceo sistem dovodi u novo operativno stanje u istom trenutku. Nedostatak se ogleda u visokom riziku povezanom sa reimplementacijom poslovnih funkcija i korektnošću rada novog sistema. Neke od postojećih tehnologija oblaka uzimaju u obzir potrebe inkrementalne migracije sistema, tako da obezbeđuje mostove za komunikaciju dela sistema na oblaku sa ostalim delovima van oblaka (na primer, Azure AppFabric Service Bus [9]). Deo aplikacije na staroj platformi trpi minimalne izmene. Na primer, ako se baza podataka prebacuje na oblak i pri tome ne trpi izmene strukture potrebno je samo u starom kodu koristiti odgovarajući (odbc ili sličan) konektor obezbeđen od strane provajdera. Ako se naprotiv, baza zadržava na postojećoj infrastrukturi, komunikacioni most automatski obezbeđuje pristup iz oblaka sa keširanjem podataka radi povećanja performansi.

Najviše napora zahteva reimplementacija postojećeg softvera za izvršavanje na oblaku. U tom scenariju mora se identifikovati i po potrebi restrukturirati arhitektura softvera. Ona opisuje softverski sistem kao kolekciju arhitekturnih komponenata. Arhitektura definiše globalnu kontrolnu strukturu, protokole komunikacije, sinhronizacije i razmene podataka među komponentama. Isto tako adresira pitanja dodele funkcionalnosti projektnim elementima, fizičke distribucije, skaliranja i performansi.

Zavisno od aspekta koji se posmatra, sistem se može dekomponovati na više načina u arhitekturne komponente: funkcionalna dekompozicija grupiše srodne funkcionalnosti u istu komponentu, strukturna dekompozicija ima za kriterijum minimalnu spregu između komponenata i maksimalnu spregu unutar komponente. Dekompozicija po slojevima najčešće identifikuje tri sloja:

- sloj sprege sa korisnikom
- sloj poslovnih pravila
- perzistentni sloj (sloj podataka)

Jeftinija i brža alternativa kompletnoj refaktorizaciji i reimplementaciji sistema je putem oblaganja (engl. wrapping) postojećeg softvera. Aplikacija je delimično ili kompletno obložena kada su neki ili svi njeni spoljni interfejsi prekriveni slojem softvera koji prevodi ili modifikuje interfejsne informacije da bi predstavio drugačiju sliku. Drugačija slika se odnosi na drugi protokol, format podataka ili skup komandi. Za primenu oblaganja, nije potrebno kompletno razumevanje strukture i ponašanja sistema, nego samo razumevanje korisničkog interfejsa (takozvano razumevanje tipa crne kutije).

Jedinica komponentizacije aplikacije u virtuelnom oblaku je instanca virtuelne mašine. Neki od principa koje treba slediti su: treba unifikovati operativno okruženje tako da ima što manje različitih templejta za instanciranje. Treba raspregnuti kod od podataka, raspregnuti funkcije koje će ići na različite virtuelne mašine, identifikovati kritične funkcije i posvetiti im posebnu pažnju (npr. redundansa i tolerantnost na otkaze). Ne treba praviti pretpostavke o fizičkoj lokaciji komponenata jer se instance VM mogu seliti sa jedne fizičke mašine na drugu. Treba koristiti kataloge, asinhrone poruke (nezavisnost komponente od stana rada druge komponente). Privatne IP adrese koriste se interno među komponentama, a komponente prednjeg kraja reaguju na mali broj javnih IP adresa čiji se zakup posebno plaća. Komponente treba da budu otporne na otkaze, da otkaz jedne ne povuče druge. Treba da budu bez internog stanja da bi u slučaju restarta mogle odmah da odgovore na zahteve. Kritične podatke treba odmah upisivati u perzistentu memoriju, ili imati dnevničke zapise. Periodični bekap svega potrebnog, od VM ili aplikativnog koda do podataka.

Skaliranje aplikacija na oblaku je relativno lako. Postiže se ili korišćenjem kapacitetnijih instanci VM (više memorije, procesorskih jezgara), ili korišćenjem većeg broja virtuelnih (ili fizičkih) mašina za određenu komponentu sa balansiranjem opterećenja. Skaliranje može biti i u suprotnom smeru, da se troši manje resursa kada ima manje zahteva. Automatski monitoring i skaliranje može se postići posebnim provajderskim servisima, ili se može napraviti posebna programska komponenta koja to radi prema preferencama autora programskog sistema.

Sigurnosti podataka treba posvetiti posebnu pažnju. Mere uključuju distribuiranje komponenata osetljivih podataka u više baza, enkripciju podataka, sigurne komunikacije (izolacija na mrežnom nivou, sigurni komunikacioni protokoli), osiguranje radnog okruženja (autorizovan pristup templejtima virtelnih aplikacija, upravljačkoj konzoli itd).

IV. Zaključak

Relativno mali broj objavljenih iskustava [10] iz migracije aplikacija na računarske oblake ukazuje da je tehnologija oblaka još u povoju. Zreliju fazu karakterisaće ukrupnjavanje provajdera tehnološke infrastukture oblaka i standardizacija na nivou osnovnih koncepata.

Literatura

[1] P. Mell, T. Grance, " The NIST Definition of Cloud Computing," Version 15, 10-07-09, National Institute of Standards and Technology (http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc).

[2] -, "WMware and Cloud Computing," Wmware Inc, 2010. (http://www.vmware.com/files/pdf/cloud/VMware-and-Cloud-Computing-BR-EN.pdf).

[3] B. Golden, "The Case Against Cloud Computing," Jan 22, 2009, CXO Media Inc (http://www.cio.com/article/477473/The_Case_Against_Cloud_Computing_Part_One).

[4] -, "vCloud API Programming Guide," EN-000180-00, Wmware Inc, 2010. (http://www.vmware.com/support/pubs).

[5] R. Fielding, R.N. Taylor, "Principled Design of the Modern Web Architecture, " *ACM Trans. Internet Technology* 2 (2), pp. 115–150.

[6] -, "Oracle Cloud Resource Model API," 2010-06-28, Oracle, (http://www.oracle.com/us/technologies/cloud/index.htm).

[7] J. Varia, "Architecting for the Cloud: Best Practices, " Amazon Web Services, Jan 2010, (http://jineshvaria.s3.amazonaws.com /public/cloudbestpractices-jvaria.pdf).

[8] D. Bojić, D. Velasević, "The Current State of Software Reengineering", *Proc. of the INFO-TECH '99*, Sep. 1999, pp. 29-37.

[9] D. Chappell, *Introducting the Windows Azure Platform.* White paper, Microsoft, Dec. 2009.

[10] A. Khajeh-Hosseini, D. Greenwood, I. Sommerville, "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS, " *IEEE 3rd International Conference on Cloud Computing*, 2010, pp.450-457.

Abstract

Cloud computing is an important new trend in IT. Existing enterprise applications will be adapted to the concept of software as a service. Given in the paper is an overview of potentials and risks associated with the new technology, as well as common technical features of current cloud solutions. Options regarding legacy applications are considered, and appropriate evolutive cloud migration strategy suggested.

CLOUD APPLICATION MIGRATION

Dragan Bojić