ELEKTROTEHNIŠKI VESTNIK 81(3): 101-106, 2014
IZVIRNI ZNANSTVENI ČLANEK

Programsko ogrodje za prenos inženirskih aplikacij v oblak

Jernej Južna, Vlado Stankovski

Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo, Jamova 2, 1000 Ljubljana, Slovenija
E-pošta: vlado.stankovski@fgg.uni-lj.si

Povzetek. Računalništvo v oblaku aplikacijam omogoča nove možnosti, kot so vseprisotnost, elastičnost in odpornost proti napakam. Inženirske aplikacije so po navadi računsko in podatkovno zahtevne, vendar še ne izkoriščajo možnosti, ki jih ponujajo tehnologije računalništva v oblaku. Z namenom, da bi olajšali prenos obstoječih inženirskih aplikacij v oblak, smo v okviru tega dela razvili učinkovito in splošno namensko programsko ogrodje. Pri delu smo se omejili na inženirske aplikacije, razvite v okolju MathWorks Matlab. Za prenos smo uporabili platformo mOSAIC kot storitev (angl. Platform as a Service, PaaS). Platforma mOSAIC se razlikuje od drugih ponudnikov PaaS, saj je odprtokodna in jo lahko namestimo na poljubni infrastrukturi kot storitev (angl. Infrastructure as a Service, IaaS). Razvito generično programsko ogrodje smo preverili tako, da smo razvili aplikacijo za oceno hitrosti izvajanja posameznih sklopov Matlaba, ki smo jo namestili na ponudnikih IaaS Amazon EC2 in Eucalyptus. Rezultati kažejo, da razvito programsko ogrodje omogoča uspešen in razmeroma preprost prenos obstoječih inženirskih aplikacij, pri tem pa sta zagotovljena pravilnost delovanja in primerljiva hitrost izvajanja.

Ključne besede: računalništvo v oblaku, inženirske aplikacije, Matlab, prenos

A Programming Framework for Porting Engineering Applications to the Cloud

Cloud computing technologies offer a lot of advanced possibilities to applications, such as elasticity, redundancy and fault tolerance. Many computationally and data-intensive engineering applications could benefit from being ported to the Cloud. The goal of the present work was to develop an effective and general-purpose programming framework that can be used by engineers to port their applications to the Cloud with minimal changes in their existing applications. This work focuses on applications developed in MathWorks' Matlab. To port the applications to the Cloud, the mOSAIC Platform as a Service (PaaS) was used. The mOSAIC PaaS distinguishes from other similar platforms by being Open Source and by its independence from the Infrastructure as a Service (IaaS) provider. The performance of the overall solution was tested against a benchmarking engineering application to estimate the Matlab performance for which two different Cloud providers - Amazon EC2 and Eucalyptus were used. The results indicate that the developed programming code assures successful porting of engineering applications to the Cloud.

1 UVOD

Računalništvo v oblaku (angl. cloud computing) [1] ponuja vrsto prednosti uporabnikom, razvijalcem in upravljavcem različnih računalniških aplikacij. Nekatere koristne lastnosti [2] takih aplikacij so na primer elastičnost, odpornost proti napakam, nadzor in vseprisotnost. Investicije v potrebno infrastrukturo ni treba vnaprej načrtovati, prav tako računalniških infrastruktur ni treba vzdrževati. Tako se razvijalci izognejo visokim začetnim stroškom, saj se v večini primerov plača le za čas, ko uporabljamo najeto računalniško infrastrukturo [3].

Razvoj novih aplikacij, ki bi delovale na najetih računalniških infrastrukturah, je pogosto dokaj zapleten in dostopen le ozkemu krogu strokovnjakov, saj je za ustrezno izkoriščenost treba poznati širok spekter zahtevnih tehnologij [4].

Še večji problem je prenos obstoječih aplikacij, saj uporaba oblačnih tehnologij zahteva temeljite posege v obstoječe aplikacije oziroma ponovni razvoj. Za inženirske aplikacije bi bila uporaba oblaka zelo privlačna, saj omogoča preprost dostop do zmogljivih računskih in pomnilniških virov po sprejemljivih cenah [3, 5]. Pri občasnih potrebah po kratkotrajnih, a intenzivnih izračunih, je tako smiselno namesto zahtevnega in dragega vzdrževanja lastne infrastrukture najeti potrebne dodatne računske vire le za čas opravljanja izračunov. Podobno se lahko aplikacija brez večjih stroškov odpre širši znanstveni skupnosti brez bojazni, kako bi morebitna povečana obremenitev vplivala na lastno infrastrukturo.

Prenos inženirskih aplikacij je še posebno zapleten, saj po navadi take aplikacije temeljijo na zahtevnih algoritmih, strogo vezanih na določeno programsko okolje, ki pa v oblaku velikokrat ni na voljo [5]. Poleg tega je lahko analiza rezultatov inženirskih aplikacij otežena, ker so podatki obsežni in pogosto porazdeljeni med več računalnikov v oblaku [6]. Obstoječe inženirske aplikacije v oblaku obravnavamo kot zastarele aplikacije (angl. legacy applications) [4], saj

Prejet 16. december, 2013
Odobren 9. maj, 2014

ne poznajo oziroma ne znajo uporabljati programskih vmesnikov okolja, nemalokrat pa vanj tudi ni mogoče preprosto posegati.

Selitev v oblak je torej cenovno dostopna opcija, vendar je zahtevnost prenosa eden glavnih razlogov, da inženirske aplikacije obtičijo v razvojnih oddelkih. Posamezni ponudniki računalništva v oblaku se težavnosti prenosa inženirskih in drugih aplikacij zavedajo in skušajo z lastnimi orodji postopek poenostaviti. To sicer lahko razvijalce vodi v odvisnost od posameznega ponudnika (angl. vendor lock-in) [7].

Namen našega dela je bil izdelati programsko ogrodje, ki se lahko uporablja za čim preprostejši prenos inženirskih aplikacij v oblak. V delu smo se osredinili na inženirske aplikacije, razvite v okolju Mathworks Matlab. Za razvoj programskega ogrodja bomo uporabili platformo kot storitev mOSAIC [8]. Razvito programsko ogrodje bomo ovrednotili na primeru prenosa generične aplikacije, namenjene testiranju zmogljivosti aplikacije, ki bo potekala na izbranih ponudnikih infrastruktur kot storitev.

2 TRENUTNO STANJE

Za prenos obstoječih inženirskih aplikacij v oblak je treba poznati možnosti, ki jih ponujajo obstoječe infrastrukture in platforme kot storitev. Oba pristopa na kratko obravnavamo v nadaljevanju.

2.1 Infrastruktura kot storitev

V okolju IaaS po navadi lahko obstoječe inženirske aplikacije prenesemo brez večjih naporov, saj imamo nadzor nad celotnim izvajalnim okoljem: lahko izberemo operacijski sistem, izvajalno okolje, namestimo vse potrebne podporne komponente, ki ustrezajo naši aplikaciji, brez potrebe po njihovem spreminjanju. Žal tako prenesena inženirska aplikacija ne izkoristi vseh prednosti, ki jih ponuja računalništvo v oblaku, saj ne zna samodejno upravljati virov. Za odpravo te pomanjkljivosti je treba razviti ločene mehanizme, ki uporabljajo API IaaS ponudnika za dodeljevanje in upravljanje virov. Uspešen prenos je odvisen tudi od same inženirske aplikacije oziroma od vgrajenega algoritma in možnosti za paralelizacijo.

V [9, 10, 11, 12] je narejen pregled uporabe in cen modela IaaS za potrebe inženirskih aplikacij in njihovih performančnih lastnosti. Rezultati teh meritev kažejo, da je računska moč v oblaku ustrezna, vendar obstaja tudi nekaj slabih lastnosti: latenca v komunikaciji med virtualnimi računalniki (angl. Virtual Machine, VM) je precejšnja, ker je zmogljivost prenosnih poti omejena z zmogljivostjo internetne povezave tako uporabnika kot ponudnika oblaka. Zato dosežejo trivialno paralelni problemi [13, 14] najvišje performance v oblaku.

2.2 Platforma kot storitev

Prenos inženirskih aplikacij z uporabo ponudnikov PaaS mora potekati nekoliko drugače, saj ponudniki predvidevajo, da bo aplikacija komunicirala z okoljem izključno prek nudenih API-jev. Uporaba posebej pripravljenih API-jev aplikacijam zagotavlja lastnosti, kot sta na primer elastičnost in odpornost proti napakam. Če niso uporabljeni API-ji, pa se različni ponudniki PaaS medsebojno zelo razlikujejo. Nekateri ponudniki, tako komercialni (GAE) kot odprtokodni (AppScale, CloudFoundry, OpenShift), prenosa obstoječih aplikacij sploh ne podpirajo. Pri drugih ponudnikih (npr. Stackato, Aneka) pa je mogoče doseči tak prenos, a končna rešitev po navadi nima vseh lastnosti aplikacij v oblaku. Pri našem delu se bomo oprli na platformo mOSAIC, ki je odprtokodna in neodvisna od ponudnikov IaaS.

3 ANALIZA ZAHTEV

Cilj našega dela je izdelava programskega ogrodja, s katerim lahko prenesemo obstoječe zahtevne inženirske aplikacije v oblak. Poudarek pri razvoju programskega ogrodja je, da omogoča čim bolj preprost in splošen prenos. Želimo si, da bo izdelano programsko ogrodje uporabno za inženirje, ki bi želeli prenesti svoje aplikacije v oblak z minimalnim trudom.

Pri delu smo se osredinili na prenos inženirskih aplikacij, razvitih v okolju Mathworks Matlab, saj le-to omogoča vrsto numeričnih algoritmov in je namenjeno inženirjem in znanstvenikom, ki ga uporabljajo na najrazličnejših področjih, med drugim v avtomobilski industriji, pri preizkušanju letalskih in vesoljskih modelov, prepoznavanju govora in slik, napovedovanju vremena, analizi borznih tečajev in pri drugih aplikacijah, ki zahtevajo uporabo različnih numeričnih algoritmov. Matlab ima svoj programski jezik, ki omogoča učinkovito in naravno opisovanje matrik in operacij nad njimi, hkrati pa omogoča tudi delo s spremenljivkami, kontrolne stavke, zanke, delo z vhodom/izhodom, definiranje funkcij itd. Izvrševanje inženirskih aplikacij temelji na interpretiranju stavkov, ki jih izvaja računski pogon Matlab (angl. Matlab engine).

Netehnične in tehnične zahteve za razvoj programskega ogrodja so tako naslednje:

- ogrodje mora biti primerno za prenos inženirskih aplikacij, napisanih v Matlabu,
- aplikacija, ki jo razvijemo s pomočjo ogrodja, mora biti dostopna na spletu v obliki programske opreme kot storitve (SaaS),
- izvajanje aplikacije mora biti mogoče brez uporabe katerekoli plačljive licence Matlab,
- aplikacija mora biti elastična – uporaba razpoložljivih virov naj se dinamično prilagaja potrebam aplikacije in
- uporaba ogrodja mora biti preprosta in vloženo delo za prenos obstoječih aplikacij mora biti minimalno.

4 ARHITEKTURA PLATFORME MOSAIC

Programsko ogrodje za prenos obstoječih inženirskih aplikacij v oblak smo razvili z uporabo odprtokodne platforme mOSAIC (Open Source API and Platform for Multiple Clouds – odprtokodni programski vmesnik in platforma za več oblakov) [15].

Platforma mOSAIC:

- je portabilni PaaS, zato programer lahko razvija aplikacije na višji stopnji abstrakcije kot pri IaaS,
- zagotovlja neodvisnost aplikacij od ponudnika IaaS,
- zagotavlja podporo aplikacijam, pisanim v jezikih Java, Node.js, Erlang in Python, ter
- ima vgrajene module za pogajanja s ponudniki IaaS.

mOSAIC omogoča določene prednosti pred drugimi rešitvami PaaS, ki niso portabilne, saj je neodvisen od programskega jezika in ponudnika IaaS. To aplikacijam zagotavlja določeno stopnjo avtonomije, ker se pri njihovem razvoju ni treba obremenjevati z vprašanjem, na kateri infastrukturi se bodo izvajale. Možnost proste izbire ponudnika IaaS v trenutku, ko je aplikacija že izdelana, prinaša prednost v smislu minimizacije stroškov, saj lahko v vsakem trenutku zamenjamo ponudnika IaaS s cenovno ugodnejšim.

Pri razvoju aplikacij je omogočen popoln nadzor nad programsko opremo (angl. enabling software), to je izbiro operacijskega sistema, podpornih knjižnic in programskega okolja, in tudi njeno spreminjanje za potrebe izvajanja aplikacij. Ta način programiranja nam omogoča tudi predelavo obstoječih (zastarelih) aplikacij brez bojazni, da bi izgubili prednosti modela PaaS.

Platforma mOSAIC vsebuje orodja in storitve, ki uporabnikom olajšajo njeno uporabo, načrtovanje, razvoj in izvajanje aplikacij, nadzor nad platformo, pomoč pri pridobivanju virov itd.

Arhitektura platforme mOSAIC je sestavljena iz nekaj plasti. Najnižja plast je namenjena podpori API-jev različnih ponudnikov IaaS. Podpora za nekaj znanih ponudnikov IaaS je že vključena in obsega Amazon EC2, Eucalyptus, OpenStack, Rackspace, FlexiScale in nekatere druge [16].

Naslednja plast je namenjena podpori pridobivanja potrebnih virov od enega ali več ponudnikov IaaS, kar je omogočeno z uporabo večagentnega sistema za posredovanje virov (angl. multi-agent brokering system).

Tretja plast zagotavlja glavne funkcionalnosti tipičnega okolja PaaS: pridobljene virtualne računalnike povezuje v celoto – gručo, ki zagotavlja računsko moč, nadzoruje izvajanje aplikacij, jim zagotavlja elastičnost ter razvršča komponente aplikacije po razpoložljivi infrastrukturi. Tretja plast skrbi tudi, da je vedno na voljo dovolj virtualnih računalnikov.

Četrta plast je namenjena programskim vmesnikom, predvsem vmesniku za medobratovalnost (angl. Interoperability API), ki v osnovi omogoča komunikacijo med komponentami aplikacije ne glede na to, kje v gruči se nahajajo in v katerem programskem jeziku so napisane.

Zadnja plast implementira programski vmesnik mOSAIC (angl. mOSAIC API) za posamezne programske jezike.

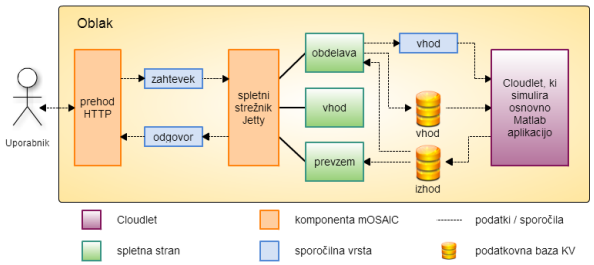
Aplikacija, razvita z uporabo programskega vmesnika mOSAIC, je sestavljena iz ohlapno povezanih sestavnih delov oziroma komponent. Komponente imajo natančno določene funkcionalnosti, ki jih definira razvijalec. Platforma mOSAIC kontrolira njihov celoten življenjski cikel. Med drugim je platforma odgovorna za njihov zagon, ustavitev ter medsebojno komunikacijo. Tako mOSAIC zagotavlja optimalno delovanje celotne aplikacije ob minimalni porabi virov.

Cloudleti so posebna vrsta komponent, katerim platforma mOSAIC zagotavlja dodatne lastnosti, kot so elastičnost, odpornost proti napakam in avtonomija. Da lahko zagotovi omenjene lastnosti, platforma mOSAIC ne razlikuje med posameznimi instancami istega Cloudleta. Želena komunikacija s Cloudletom je lahko preusmerjena povsem avtomatično h katerikoli njegovi instanci, zato mora biti Cloudlet napisan tako, da je njegovo izvajanje popolnoma neodvisno od trenutnega stanja in števila instanc.

Določene standardne komponente, kot npr. porazdeljeni datotečni sistem, porazdeljena podatkovna baza, sporočilne vrste itd., so v platformo že vključene in tako njihova razpoložljivost ni odvisna od morebitne podpore ponudnika IaaS.

5 IMPLEMENTACIJA

V nadaljevanju smo izdelali načrt generičnega programskega ogrodja, ki bi ga lahko inženirji uporabljali za prenos aplikacij v oblak. Načrt temelji na platformi mOSAIC, prenesene aplikacije pa bi lahko potekale pri poljubnem ponudniku IaaS.



Slika 1: Arhitektura programskega ogrodja za prenos inženirskih aplikacij (povzeto po: J. Južna [17])

Kot je razvidno iz slike 1, bo končni uporabnik do aplikacije dostopal prek spleta. Predvidene so najmanj tri spletne strani, in sicer stran **vhod,** prek katere uporabnik odda vhodne datoteke, stran **obdelava**, ki je namenjena prikazu poteka izračuna, ter stran **prevzem**, ki se uporablja za prevzem izhodne datoteke po končanem izračunu.

Načrtovana aplikacija bo delovala takole. Uporabnik bo obiskal URL spletne strani **vhod**. Zahtevek HTTP bo prevzela komponenta prehoda HTTP (HTTPg), ki ga bo posredovala komponenti strežnika spletnih strani (Jetty). Nato se sproži naslednja spletna stran, **obdelava**, ki deluje na podlagi mehanizma AJAX (angl. Asynchronous JavaScript and XML). Ta datoteko pregleda, preveri njeno veljavnost, jo zapiše v podatkovno bazo **vhod** (za storitev skrbi vgrajena komponenta shranjevanja ključ-vrednost) pod naključno generiranim ključem in pošlje sporočilo, ki ta ključ vsebuje, v ustrezno sporočilno vrsto (za storitev skrbi vgrajena komponenta sporočilnih vrst). Tam sporočilo prevzame Cloudlet, ki iz podatkovne baze **vhod** na podlagi ključa v sporočilu prevzame vhodne podatke in z njimi zažene (simulira) osnovno inženirsko aplikacijo (to je lahko npr. katera od operacij, ki so predstavljene v tabeli 1). Ob uspešni izvedbi računske naloge se izhodni podatki zapišejo v podatkovno bazo **izhod**. Spletna stran **obdelava** zazna ta zapis in uporabniku omogoči prevzem datoteke z rezultati. Če se uporabnik za to možnost odloči, se zažene spletna stran **prevzem**, ki podatke prevzame iz podatkovne baze **izhod** ter jih v obliki izhodne datoteke posreduje uporabniku.

V nadaljevanju smo razvili Cloudlet, ki simulira delovanje navadne inženirske aplikacije, razvite v okolju Matlab. Največja težava pri implementaciji je natančno posnemanje izvajanja Matlaba. Prepis v programski jezik Java je nepraktičen in zelo zahteven proces, saj Matlab pri določenih funkcijah notranje uporablja zahtevne, numerično nestabilne algoritme, ki navsezadnje niso javno objavljeni. Izpolnitev zahtev po pravilnosti delovanja osnovne inženirske aplikacije je tako mogoča le z uporabo MCR (Matlab Compiler Runtime). Le-ta zagotavlja, da se prevedene aplikacije izvajajo popolnoma enako kot v osnovnem okolju Matlab. Pri uporabi prevajalnika je na voljo več izhodnih oblik za različna programska okolja. Za naše potrebe smo uporabili dodatek, imenovan Builder JA (skupno ime MCJ), kjer izhod prevajanja dobimo v obliki arhiva JAR, in je torej izhod lahko neposredno uporaben v Cloudletu, ki je implementiran v programskem jeziku Java.

Glede na to, da pridobljeni JAR popolnoma opravi delo osnovne aplikacije, mora delovanje preostalega dela Cloudleta poskrbeti le še za ustrezno komunikacijo z okoljem (sprejemanje in obdelava sporočil, prevzem in zapis podatkov v podatkovno bazo) in posredovanje izračuna v obdelavo pridobljenemu arhivu JAR. Razvidno je, da je ta del delovanja neodvisen od osnovne inženirske aplikacije in celoten delovni tok Cloudleta je tako vnaprej natančno znan. Na tej podlagi smo lahko pripravili predlogo, ki vsebuje vso potrebno izvorno kodo za preprosto izdelavo poljubnih Cloudletov, ki rešujejo različne inženirske probleme.

Za testiranje smo v razviti Cloudlet vgradili vrsto sintetičnih operacij (povzetih po [18]), ki so prikazani v tabeli 1. Testi so razdeljeni v tri sklope po pet testov: prvi sklop obsega delo z matrikami, drugi izvaja funkcije nad matrikami, tretji pa testira hitrost izvajanja programske kode. Pri testiranju vsako od naštetih operacij ponovimo petkrat.

Tabela 1: Seznam opravljenih testov

Test	Operacija
1	gradnja/preoblikovanje matrik
2	potenciranje
3	sortiranje
4	vektorski produkt
5	linearna regresija
6	hitra Fourierjeva transformacija
7	lastni vektorji
8	determinanta
9	razcep Choleskega
10	obratna matrika
11	Fibonaccijeva števila
12	Hilbertova matrika
13	največji skupni delitelj
14	Toeplitzova matrika
15	RV koeficient

Vhod v generično inženirsko aplikacijo je parameter, ki definira število ponovitev posameznega testa. Izhod aplikacije sta izpis povprečnih časov izvajanja posameznega testa na standardni izhod in matrika, ki vsebuje enake podatke.

6 REZULTATI

Razvito aplikacijo smo izvajali na več različnih testnih infrastrukturah (glej tabelo 2) in vse teste smo ponovili petkrat (glej tabelo 3).

Prva infrastruktura je namizni računalnik, na katerega je nameščena programska oprema Matlab IDE. Druga infrastruktura je namizni računalnik, na katerega je nameščena testna virtualna gruča (angl. Portable Testing Cluster, PTC). Testna virtualna gruča je orodje, ki je zelo uporabno pri razvoju in testiranju aplikacij, saj simulira IaaS ponudnika na lastni delovni postaji. Testna virtualna gruča deluje tako, da ustvari določeno število virtualnih računalnikov, na katerih se zažene razvita aplikacija. Pri tem ni mogoče pričakovati pohitritve, saj je dejanska infrastruktura en sam fizični računalnik. Tretja infrastruktura pomeni fizično gručo IaaS ponudnika BUT, na katerega je nameščena prilagojena programska oprema Eucalyptus. Četrta infrastruktura je IaaS ponudnik Amazon EC2. mOS je okrnjen operacijski sistem, ki temelji na Linuxu.

Tabela 2: Seznam testnih infrastruktur

IaaS	CPE	RAM	OS
Matlab IDE namizni računalnik	2x 2640M@2.80 GHz	4 GB	Ubuntu 10.04
PTC	1x 2640M@2.80GHz	1.5 GB	mOS
BUT Eucalyptus	2x E5504@2.00GHz	2 GB	mOS
Amazon EC2	2x 2.5 ECU	1.7 GB	Ubuntu 10.04

Pri testiranju nas je zanimala razlika med hitrostjo izvajanja testov iz tabele 1 v okolju Matlab na namiznem računalniku ter hitrostjo izvajanja testov v obliki Cloudleta (glej poglavje 5) na treh različnih infrastrukturah: na namiznem računalniku, na katerega je nameščena testna virtualna gruča PTC, in na IaaS ponudnikih BUT Eucalyptus in Amazon EC2.

Rezultati testiranja so zbrani v tabeli 3 in nazorno prikazujejo prednosti platforme mOSAIC pred drugimi ponudniki PaaS. Aplikacije, razvite z uporabo platforme mOSAIC, se lahko izvajajo na poljubnem zasebnem ali javnem ponudniku IaaS in se tako izognemo problemu zaklepa na posameznega ponudnika (angl. vendor lock-in). Tako smo dosegli enega naših poglavitnih ciljev, da lahko razvito aplikacijo izvajamo na poljubnem ponudniku IaaS. V praksi bo to videti tako, da bomo za izvajanje aplikacije vedno izbrali ponudnika IaaS, ki ponuja najboljšo kakovost storitev (angl. Quality of Service). V našem primeru se je ponudnik IaaS Amazon EC2 izkazal za boljšega od BUT Eucalyptusa.

Tabela 3: Povprečni rezultati opravljenih testov (v s)

Test	Matlab IDE	PTC	BUT Eucalyptus	Amazon EC2
1	0.101	2.923	0.287	0.261
2	0.972	2.631	1.051	0.624
3	0.983	3.910	1.097	0.751
4	1.122	5.403	1.820	1.073
5	0.646	3.109	1.118	0.721
6	0.155	3.828	0.526	0.380
7	3.057	5.395	6.095	4.729
8	0.700	3.562	1.180	0.692
9	0.907	4.059	1.389	0.818
10	0.586	2.201	0.866	0.599
11	0.595	1.495	0.636	0.349
12	0.547	18.28	2.168	1.275
13	0.599	1.105	0.888	1.006
14	0.008	0.013	0.012	0.014
15	0.726	1.354	0.741	1.021
Σ	58.53	296.3	99.37	71.58

Predstavljeni rezultati kažejo tudi, da je zaradi dodanih operacij (glej poglavje 5) izvajanje testov prek Cloudleta nekoliko počasnejše na ponudnikih IaaS (BUT Eucalyptus in Amazon EC2) kot izvajanje testov v njihovi osnovni obliki na namiznem računalniku in v okolju Matlab. Razlike so v mejah sprejemljivega glede na prednosti, ki jih dosežemo s prenosom v oblak: večja dostopnost inženirskih aplikacij in prenosljivost. Glede na [16] so Cloudleti tudi elastični. Testna virtualna gruča PTC je bila najpočasnejša, kar je bilo pričakovati, saj gre za gručo virtualnih računalnikov, ki se izvajajo na enem samem namiznem računalniku, na katerem so bili izvajani testi.

7 SKLEP

Izdelali smo programsko ogrodje, s katerim lahko v oblak uspešno prenesemo zahtevne inženirske aplikacije, napisane v Matlabu. Za razvoj ogrodja smo uporabili platformo mOSAIC. Poudarka pri razvoju ogrodja sta bila preprostost in splošnost, tako da je razvito programsko ogrodje uporabno za inženirje, ki bi želeli prenesti svoje aplikacije v poljuben oblak IaaS z minimalnim trudom.

LITERATURA

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, zv. 25, st. 6, str. 599–616, 2009.

[2] I. Foster, Y. Zhao, I. Raicu, S. Lu. Cloud computing and grid computing 360-degree compared. V zborniku Grid Computing Environments Workshop, str. 1–10. IEEE, 2008.

[3] M. Armbrust, A. Fox, R. Grith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica in drugi. A view of cloud computing. Communications of the ACM, zv. 53, št. 4, str. 50–58, 2010.

[4] S. Panica, M. Neagul, C. Craciun, D. Petcu. Serving Legacy Distributed Applications by a Self-configuring Cloud Processing Platform. V zborniku International Conference on Intelligent Data Acquisition and Advanced Computing Systems, zv. 1, str. 139–144, 2011.

[5] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good. The cost of doing science on the cloud: The Montage example. V zborniku International Conference for High Performance Computing, Networking, Storage and Analysis, str. 1–12, 2008.

[6] I. Tomašić, A. Rashkovska, M. Depolli, R. Trobec. A comparison of Hadoop tools for analyzing tabular data. Informatica 37(1), 131–138, 2013.

[7] N. Leavitt. Is cloud computing really ready for prime time. Growth, zv. 27, st. 5, str. 15–20, 2009.

[8] B. Di Martino, D. Petcu, R. Cossu, P. Goncalves, T. Mahr, M. Loichate. Building a mOSAIC of Clouds. V zborniku Euro-Par 2010 Parallel Processing Workshops, str. 571–578. Springer, 2011.

[9] I. Foster. Designing and building parallel programs, zv. 95. Addison-Wesley Reading, MA, 1995.

[10] G. Juve, E. Deelman, G. B. Berriman, B. P. Berman, P. Maechling. An evaluation of the cost and performance of scientific workflows on Amazon EC2. Journal of Grid Computing, zv. 10, st. 1, str. 5–21, 2012.

[11] C. Vecchiola, S. Pandey, R. Buyya. High-performance cloud computing: A view of scientific applications. V zborniku International Symposium on Pervasive Systems, Algorithms, and Networks, str. 4–16. IEEE, 2009.

[12] J. Južna, P. Češarek, D. Petcu, V. Stankovski. Solving Solid and Fluid Mechanics Problems in the Cloud with mOSAIC. IEEE Computers in Science and Engineering, 2014 (in press).

[13] C. Evangelinos, C. Hill. Cloud Computing for parallel Scientic HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. ratio, zv. 2, st. 2.40, str. 2–34, 2008.

[14] K. Jorissen, F. D. Vila, J. J. Rehr. A high performance scientific cloud computing environment for materials simulations. Computer Physics Communications, zv. 183, st. 9, str. 1911–1919, 2012.

[15] mOSAIC: Open-Source API and Platform for Multiple Clouds. 2013. Dostopno na: http://www.mosaic-cloud.eu/.

[16] D. Petcu, B. Di Martino, S. Venticinque, M. Rak, T. Mahr, G. E. Lopez, F. Brito, R. Cossu, M. Stopar, V. Stankovski in drugi. Experiences in building a mOSAIC of clouds. Journal of Cloud Computing: Advances, Systems and Applications, zv. 2, st. 1, str. 12, 2013.

[17] J. Južna, Prenos inženirskih aplikacij v oblak s platformo mOSAIC. Magistrska naloga, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2013.

[18] P. Coey. Benchmarking the Amazon Elastic Compute Cloud (EC2). Doktorska disertacija, Worcester Polytechnic Institute, 2011.

Jernej Južna je diplomiral leta 2006 in magistriral leta 2013 na Fakulteti za računalništvo in informatiko v Ljubljani. Njegovo raziskovalno zanimanje sta e-izobraževanje in računalništvo v oblaku.

Vlado Stankovski je diplomiral leta 1995, magistriral leta 2000 in doktoriral leta 2009 na Fakulteti za računalništvo in informatiko v Ljubljani. Njegovi interesi so na področju distribuiranega računalništva, tehnologij grid in računalništva v oblaku.