Razvoj obremenitvenih testov z odprtokodnim orodjem JMeter

[1]Primož Kastelic, [2]Mirjana Kljajić Borštnar, [3]Robert Leskovar
[1]3GEN, d. o. o.
[2,3]Univerza v Mariboru, Fakulteta za organizacijske vede, Kidričeva cesta 55a, 4000 Kranj
primoz.kastelic@3gen.si; mirjana.kljajic@fov.uni-mb.si; robert.leskovar@fov.uni-mb.si

Izvleček

V prispevku je prikazano obremenitveno testiranje informacijskega sistema z odprtokodnim orodjem JMeter v distribuirani testni arhitekturi. Objekt testiranja je obsegal skupek štirinivojskih aplikacij, pri čemer smo uporabili pet virtualnih strojev, dva virtualna stroja za IBM Websphere Portal, dva za aplikacijski strežnik IBM Websphere Application Server in enega za delilnik bremena HAProxy. Definirali smo tri testirne scenarije in tri testirne primere za vsak scenarij. Posamezni testirni primeri predvidevajo različno število generiranih uporabnikov (50, 75, 100, 200, 300 in 500), različno dinamiko generiranja uporabnikov (v 10 s, 60 s, 120 s, 240 s in 360 s) ter čase izvajanja testov med 57 in 728 s. Rezultati izvedenega testiranja omogočajo, da skupek testiranih spletnih aplikacij prenesemo v produkcijsko okolje.
Ključne besede: obremenitveno testiranje, Apache JMeter, upravljanje informacijskih sistemov, IEEE 829-2008.

Abstract

Load test development via the JMeter open-source software

The article outlines the IT system load testing using the Apache JMeter open source software tool in a distributed test architecture. The testing object consisted of a set of 4-level applications. In testing, we used 5 virtual machines, two IBM Websphere Portal virtual machines, two IBM Websphere Application Server virtual machines and one HAProxy load balancer virtual machine. Furthermore, for each scenario, we defined three test scenarios and three test cases. Individual test cases provide for different numbers of generated users (50, 75, 100, 200, 300 and 500), different user generation dynamics (10s, 60s, 120s, 240s) and running test times between 57s and 728s. All test cases have shown that the response time was appropriate and memory consumption will not exceed the allocated limits. Based on such results of testing, we can deploy the set of applications in a production environment.
Keywords: load testing, Apache JMeter, information systems management, IEEE 829-2008.

1 UVOD

V zadnjih letih se je uporaba računalništva v oblaku razširila na različna področja. To je pripeljalo tudi v povečanje števila aplikacij, ki delno ali popolnoma tečejo v oblaku (Gomez Saez, Andrikopoulos, Leymann in Strauch, 2015). V Sloveniji smo decembra 2015 vzpostavili sistem državnega računalništva v oblaku. Državni računalniški oblak (DRO) je namenska računalniška infrastruktura v lasti in upravljanju države, ki omogoča državnim institucijam (neposrednim proračunskim uporabnikom), da z uporabo koncepta računalništva v oblaku hitro in poceni dosežejo svoje poslovne cilje. Ponuja računske, shranjevalne, razvojne, poslovne in druge zmogljivosti v obliki storitev (Državni računalniški oblak, MJU, 2015).

Carinski informacijski sistem je ravno v fazi prenove, aplikacije se selijo na višje verzije aplikativne in sistemske programske opreme, istočasno pa poteka tudi selitev strežnikov z IBM 2097 z10 (IBM System z10 EC, 2013) na virtualizirano arhitekturo Intel/Linux. Eden prvih uporabnikov DRO bosta ravno Finančna uprava Republike Slovenije in Slovenski carinski informacijski sistem (SICIS).

Ker selitev na arhitekturo Intel/Linux že poteka in se bodo kasneje v DRO prenesli že delujoči virtualni stroji, poznamo prepustnost, zanesljivost in razpoložljivost že pred prenosom. Spoznavanje lastnosti sistema je temeljnega pomena za razumevanje

njegovega delovanja ter izbiro ustreznih metod za preučevanje njegovega obnašanja (Kljajić, 1994).

Informacijski sistem je slovenska carinska administracija poimenovala Slovenski carinski informacijski sistem – portal SICIS (angl. Slovenian customs information system). Znotraj portala SICIS sedaj deluje več sistemov. Carinska Uprava RS je že od samega začetka stremela k temu, da bi bil informacijski sistem čim bolj učinkovit. Posledično so se začeli razvijati različni podsistemi, ki povečujejo njegovo funkcionalnost. Takšni podsistemi so (Informacijski sistem Carinske uprave RS, 2015):

- sistem ECS (angl. export control system), ki je podsistem sistema SIAES ter služi dokazovanju izstopa blaga iz Skupnosti;
- sistem poslovnih pravil, ki skrbi in opozarja na pravilnost izpolnjevanja deklaracij skladno s Carinskim zakonikom Skupnosti, Zakonom o upravnem postopku ipd.;
- sistem analize tveganja, katerega naloga je, da identificira tvegane pošiljke, ki jih pooblaščene uradne osebe ustrezno kontrolirajo;
- sistem EORI (angl. economic operator identification number), ki preverja EORI številke gospodarskih subjektov.
- sistem GCUKOD, ki preverja zneske garancij gospodarskih subjektov.

Ogrodje SICIS sestavljajo štiri ključne arhitekturne komponente:

- IBM Websphere Portal,
- IBM Websphere Application Server,
- Oracle DB,
- IBM Websphere Message Broker in IBM Websphere Message Queue.

Testni sistem smo zgradili z namenom selitve obstoječega informacijskega sistema na nove in boljše tehnologije. Da bi testni sistem deloval po pričakovanjih, brez odpovedi in z optimalno porabljenimi viri, smo poskušali omiliti morebitna odstopanja s cikličnim ponavljanjem testiranja, analiziranja in ponovnega nastavljanja sistema. Nad novim testnim sistemom smo izvajali različne tehnike testiranja, ki so podlaga za zbiranje podatkov z opazovanjem in preverjanjem aplikacijskih dnevnikov, strežniških dnevnikov in odzivnih časov na delilniku bremena ter z opazovanjem in preverjanjem dnevnikov ter poročil orodij za izvajanje testov.

2 METODOLOGIJA DELA

Cilji raziskave so:

- predstaviti odprtokodno orodje Apache JMeter,
- izdelati testirno dokumentacijo v skladu s standardom testiranja IEEE 829-2008,
- testirati informacijski sistem SICIS z obremenitvenimi testi,
- dokumentirati izvedbo in zaključek testiranja in
- oceniti, ali je informacijski sistem SICIS zrel za uporabo v produkcijskem okolju.

V članku se ukvarjamo z raziskovalnim področjem obremenitvenega in stresnega testiranja. Teoretični okvir testiranja smo obdelali s poglobljenim preučevanjem tuje in domače literature ter preučevanjem že delujočih sistemov. Na podlagi pridobljenega znanja smo izdelali tudi testirno dokumentacijo v skladu s standardom IEEE 829-2008 (angl. Standard for software and system test documentation) in izvedli obremenitveno testiranje informacijskega sistema SICIS.

Obremenitveno testiranje je učinkovita metoda za izboljšanje zanesljivosti nekega produkta. Slabosti, ki jih pri proizvodnji ni mogoče zaznati, lahko kasneje povzročijo odpoved. Stresno testiranje nam omogoči, da nevidne napake postanejo vidne in jih lahko opazimo in popravimo (Chan, 1995).

Obremenitveno testiranje je posebna vrsta testiranja zmogljivosti, pri katerem ugotavljamo (merimo) robustnost, razpoložljivost, zanesljivost, odzivnost in podobne karakteristike. Uporabljamo ga za ugotavljanje obnašanja nekega sistema pri normalni in pri povečani obremenitvi. Povečana uporaba je vnaprej določena tako z volumnom (velikostjo) in hitrostjo zahtev, kot tudi s časovno dinamiko. Obremenitveno testiranje pomaga ugotoviti: a) kakšna je maksimalna prepustnost, b) kje so ozka grla in c) kateri element sistema jo povzroči. Ko se obremenitev poveča nad meje normalnega delovanja, govorimo o stresnem testiranju (Erinle, 2013).

Priljubljenost obremenitvenega testiranja spletnih aplikacij v svetovnem merilu narašča, kar dokazujejo tudi številni tuji članki, ki opisujejo to problematiko. V nadaljevanju so predstavljene izbrane raziskave.

Kiran, Mohaparta in Swamy (2015) so v svoji študiji uporabili orodje JMeter kot učinkovito orodje za izvajanje obremenitve določenemu sistemu. Uporabili so tehniko distribuiranega testiranja. Teste so ciklično ponavljali, vmes so popravljali število

uporabnikov, izstrelne čase ter parametre sistema. Rezultate so analizirali z orodjem JMeter.

Avtorji Jing, Lan, Hongyuan, Yuqiang in Guizhen (2010) v svojem delu obravnavajo simulacijo staranja računalniškega sistema. Kot orodje za obremenjevanje so uporabili JMeter v distribuirani arhitekturi, s katerim so poganjali dva scenarija. Pri prvem scenariju so močno obremenjevali statične strani, pri drugem strežnik J2EE Tomcat in MYSQL kot bazo podatkov. Kazalnike obremenjenih sistemov so dobili neposredno s področja, katerega vrednosti so shranjevali na dve sekundi.

Wu in Wang (2010) v prispevku govorita, kako strm je trend naraščanja aplikacij J2EE (v to kategorijo spadajo tudi aplikacije IBM WebSphere). Avtorja navajata, kateri sistemski parametri lahko vplivajo na boljše delovanje takih aplikacij. Sem spada predvsem velikost javanske kopice. Odsvetujeta aplikacijski strežnik in bazni strežnik na istem stroju. Kakor že pri zgoraj omenjenih delih sta tudi Wu in Wang za testiranje uporabila orodje JMeter v distribuirani arhitekturi in nad sistemom izvajala vrsto testov, katerim sta spreminjala parametre. Članek jasno pokaže pozitivni učinek apliciranja predlaganih sprememb nad ciljnim sistemom J2EE.

Avtorji v svojih delih obremenitveno testiranje uporabijo kot metodo, s katero ugotavljajo, da njihov predmet testiranja pod močnimi obremenitvami res počne to, za kar je narejen.

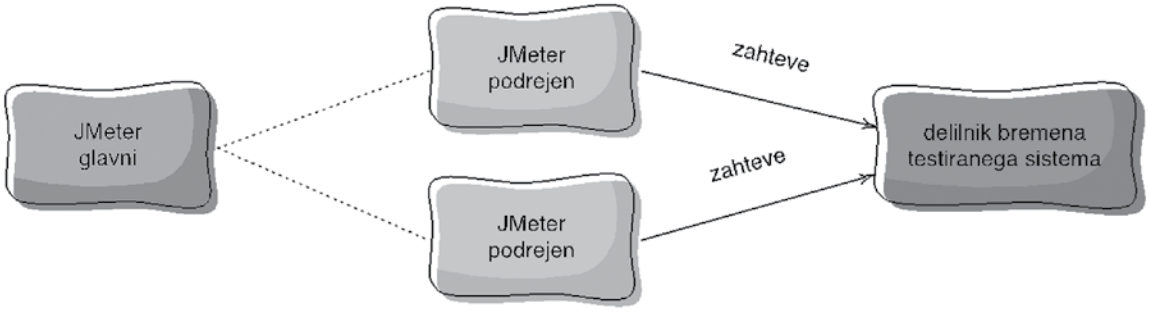
2.1 Orodje za testiranje in način testiranja

Kot orodje za poganjanje testov smo uporabili odprtokodni in prosto dostopni Apache JMeter (Apache JMeter, 2015). Apache JMeter lahko uporabljamo za testiranje učinka tako statičnih kot dinamičnih virov. Uporabljamo ga lahko za simuliranje močne obremenitve strežnika ter mrežnih virov, na podlagi česar analiziramo delovanje sistema pod različnimi pogoji. To orodje je namenjeno določenim testiranjem sistema (npr. stresno testiranje) in testiranju učinkovitosti (število obdelanih zahtev na časovno enoto) (Halili, 2008). Korake, ki so potrebni za izvajanje testiranja, lahko strnemo v naslednje točke:

- identificiramo testno okolje,
- identificiramo kriterij(e) uspešnosti,
- načrtujemo in kreiramo testirne načrte oz. testirne primere,
- kreiramo testirno okolje,
- izvedemo testirne načrte oz. testirne primere,
- analiziramo rezultate,
- posodobimo testirne načrte/primere in spremenimo parametre opazovanega sistema oz. programa.

Objekt testiranja je SICIS. Primere testiranja večnivojske spletne aplikacije so obravnavali številni avtorji (Iyer, Gupta in Johri, 2005). Pri poganjanju testov je treba paziti, da ozko grlo ne postane računalnik, na katerem testiramo. Pri našem delu smo se temu izognili tako, da smo uporabili način distribuiranega testiranja. Slika 1 prikazuje našo arhitekturo testiranja. Na tri računalnike smo namestili orodje JMeter, dva identična računalnika sta delovala kot podrejena (angl. slave), prenosnik pa smo namenili za nadzor izvajanja testov in prikaz rezultatov, imel pa je tudi vlogo gospodarja (angl. master). Karakteristike računalnikov so:

- enkrat OSX 10.11.2, MacBook Pro (Retina 13-inch), 2,5 GHz Intel Core i5, 8GB RAM,
- dvakrat Intel Linux 64, Lenovo ThinkCenter, 3,2 GHz Intel Core i5, 8GB RAM.



Slika 1: **Distribuirana testna arhitektura z orodjem JMeter**

2.2 Testirni sistem

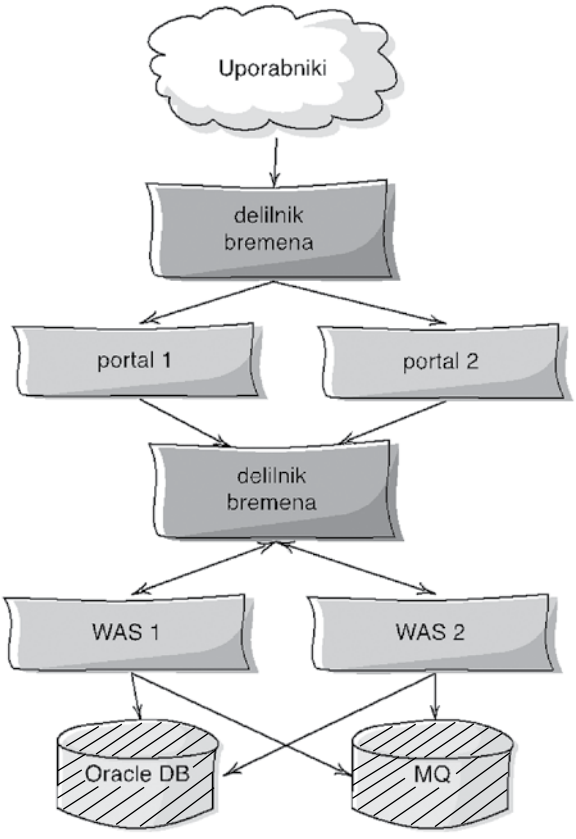
Testirali smo skupek aplikacij, ki so zgrajene štirinivojsko (uporabnik, predstavitveni nivo, poslovni nivo in podatkovni nivo). Za implementacijo testirne arhitekture smo uporabili pet virtualnih strojev, osnovanih na Linuxovem jedrnem modulu (KVM, angl. Karnel-based Virtual Maschine), z naslednjimi karakteristikami:

- dvakrat Intel Linux 64 + IBM Websphere Portal 8.5.5, KVM 8GB RAM, CPU 4 socket, 1 core per socket, Intel Xeon CPU E5-2670, 2.60 GHz,
- dvakrat Intel Linux 64 + IBM Websphere Application Server 8.5.6, KVM 8GB RAM, CPU 4 socket, 1 core per socket, Intel Xeon CPU E5-2670, 2.60 GHz,
- enkrat Intel Linux 64 + HAProxy 1.6.2 load balancer, KVM 8GB RAM, CPU 4 socket, 1 core per socket, Intel Xeon CPU E5-2670, 2.60 GHz.

Testirna arhitektura je prikazana na sliki 2. Podatkovna zbirka Oracle 12c in strežnik IBM Websphere Message Queue (MQ) nista bila predmet testiranja, zato smo ju na sliki prikazali črtkano. Delilnik bremena je zaradi preglednosti sheme prikazan dvakrat, fizično pa je to en virtualni stroj, ki deli zahtevke, namenjene portalnim aplikacijam na IBM Websphere Portal, in zahtevke, namenjene zalednim servisom, ki tečejo na strežnikih IBM Websphere Application Server (WAS).

Kot delilnik bremena smo uporabili odprtokodno rešitev HAProxy. HAProxy omogoča pršenje sedmega, t. i. aplikacijskega sloja (The OSI Model's …, 2015). V večini primerov administratorji uporabijo HAProxy in HTTP pršenje povsod tam, kjer je zahtevana visoka razpoložljivost kot nuja zaradi neprekinjenega poslovanja (Levine, 2015).

Pri postavitvi testnega okolja smo se oprli tudi na študijo redundančnega postavljanja podatkovnih centrov (Redundancy …, 2015). Za testno okolje smo izbrali postavitev 2N, pri čemer je N sklop strežnikov, ki so potrebni za normalno obratovanje, in 2N podvojeni, redundančno postavljeni sistem.



Slika 2: **Shematski prikaz testirnega okolja SICIS**

2.3 Testirni scenariji

Pod pojmom scenarij si predstavljamo delo enega uporabnika, ki se prijavi v portal, nato izvaja operacije od manj zahtevne do zelo zahtevne, na koncu pa se tudi odjavi iz portala. Pri pisanju testnih scenarijev smo upoštevali raznolikost obremenitve. Scenarije smo razdelili na tri razrede z oznakami SC1, SC2 in SC3. Vsak razred nam predstavlja en testni scenarij. Razred smo nato delili naprej na število uporabnikov in na čas razporejenosti prihodov do strežnega mesta (angl. ramp up time). V tabeli 1 so prikazani scenariji od SC1 do SC3 s številom uporabnikov in z izstrelnim časom uporabnika ter časom izvajanja testirnega primera.

Tabela 1: **Prikaz testirnih scenarijev in testirnih primerov**

Scenarij	Testirni primer	Število uporabnikov	Izstrelni čas (s)	Čas izvajanja testirnega primera (s)
SC1	SC1U50RT10T98	50	10	98
	SC1U200RT60T57	200	60	57
	SC1U500RT120T131	500	120	131
SC2	SC2U50RT60T112	50	60	112
	SC2U200RT120T387	200	120	387
	SC2U300RT240T613	300	240	613
SC3	SC3U50RT120T392	50	120	392
	SC3U75RT240T540	75	240	540
	SC3U100RT360T728	100	360	728

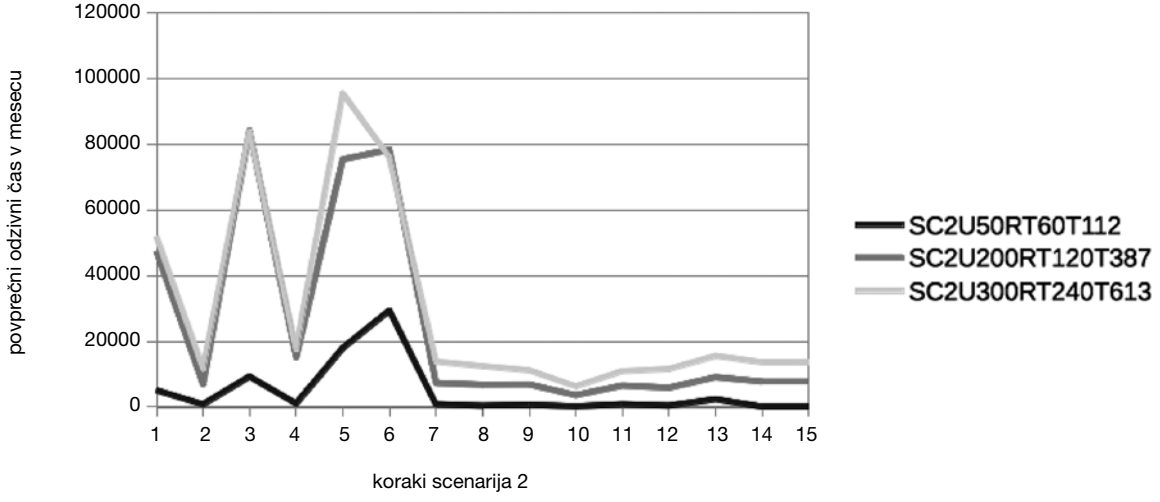
Ustreznost in potrditev scenarija smo preverili tako, da smo izvedli preizkus testa z enim uporabnikom. Scenarij smo imeli za uspešen le takrat, ko je bil odstotek napak enak točno 0. Teste smo izvajali večkrat zapored, vsako izvajanje testa smo označili z identifikacijsko številko. Pazili smo, da smo teste vedno izvajali pod enakimi pogoji in z enakimi vhodnimi parametri. Rezultate smo lahko grafično spremljali že med samim delovanjem, vsi rezultati testiranj so se shranjevali in uredili glede na datum in identifikacijsko številko testiranja. Rezultate testiranja smo shranjevali v SVN (Apache Subversion, 2015), od koder smo kasneje črpali podatke za kasnejše analize.

2.4 Analiza rezultatov testiranja

Gradivo za analizo rezultatov lahko razdelimo na naslednje kategorije:

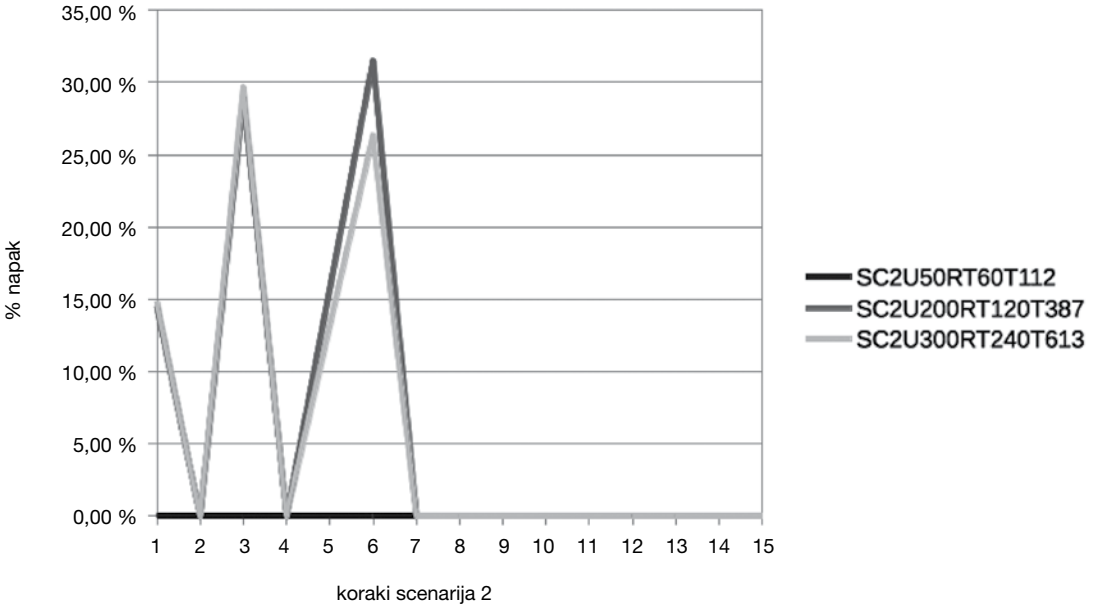
- poročila orodja JMeter,
- strežniški dnevniki,
- aplikacijski dnevniki,
- sistemski dnevniki,
- statistika na delilniku bremena.

Zbrane rezultate orodja JMeter smo pregledali, podatke testov istega razreda smo logično združevali v tabele. Dobljene rezultate smo primerjali med seboj. Slika 3 prikazuje primerjavo rezultatov poganjanja testov SC2. Ker je testni scenarij enak, spreminjajo se samo parametri testa, število hkratnih uporabnikov in izstrelni časi, so podatki primerljivi med seboj. Iz slike je razvidno, kako se s povečanjem števila uporabnikov povečajo tudi odzivni časi korakov testirnega primera.



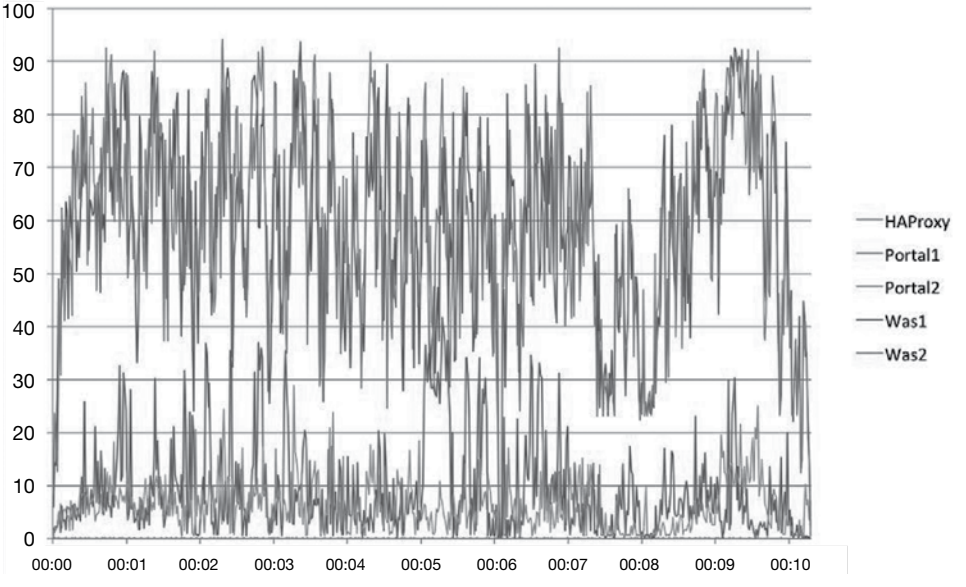
Slika 3: **Povprečni odzivni časi glede na število uporabnikov in izstrelne čase**

Iz slike 3 razberemo, da izstopata koraka 3 in 5, pri katerih se odzivni časi močno povečajo. Slika 4 prikazuje, kako se s povečanjem števila uporabnikov in povečano obremenitvijo sistema poveča tudi odstotek napak posameznega koraka.



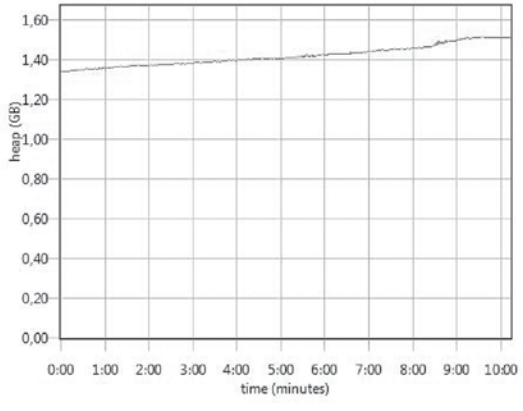
Slika 4: **Odstotek napak glede na število uporabnikov in izstrelne čase**

Slika 5 prikazuje, kakšna je bila procesorska obremenitev na strežnikih med testiranjem s testnim razredom SC2U200RT120T387. Večji del procesorske obremenitve prevzame poslovni sloj aplikacij, kar je razumljivo, saj portalski del predstavlja predstavitveni sloj, poslovni sloj – jedro SICIS – pa predstavljajo strežniki IBM Websphere Application Server, tudi kot podatkovni sloj baze podatkov in sporočilni sistem.



Slika 5: **Procesorska obremenitev strežnikov med izvajanjem testa SC2U300RT240T613**

Za zajem podatkov o virtualnih strojih smo uporabili sistemsko orodje System Activity Report – SAR (SUSE LLC and contributors, 2015). Z orodjem SAR zajemamo vse podatke o sistemu, npr. zasedenost procesorjev, poraba spomina, vhodno-izhodne operacije ter podatke o mrežnem prometu. Orodje SAR shranjuje podatke o sistemu v datoteke, razporejene po dnevih. Iz teh datotek lahko izluščimo podatke tudi kasneje, ko smo končali testiranje. Koristne informacije črpamo tudi iz dnevnika oz. spletne administratorske konzole delilnika bremena. V danem trenutku lahko vidimo, ali se zahteve pravilno pršijo glede na nastavljene uteži, te smo nastavili na 50 odstotkov. Delilnik bremena ima tudi funkcijo kontrole zalednih strežnikov, tako v vsakem trenutku vidimo, ali se zaledni strežnik še odziva, koliko časa je preteklo od zadnjega testa in koliko časa strežnik že deluje. Na koncu nam ostane še analiza javanskih kopic (angl. Java heap). Na spletišču IBM Knowledge Center najdemo napotke, kako optimizirati IBM Websphere Application Server in IBM Websphere Portal, med drugim priporočajo tudi uporabo IBM Support Assistant (IBM Support Assistant, 2015). Ko strežnik pripravimo za analizo, njegovo dnevniško datoteko naložimo v orodje in kot rezultat dobimo poročilo. Primer takega poročila vidimo na sliki 6. Ta prikazuje statistiko javanske kopice med izvajanjem testa SC2U300RT240T613 na strežniku IBM Websphere Application Server št. 1. S slike 6 je razvidno, da se vrednost kopice v časovnem obdobju giblje malo nad 1,3 GB do prek 1,5 GB. Naše priporočilo je, da se začetna vrednost kopice nastavi na 1,3 GB, končna vrednost pa vsaj na 1,5 GB oz. več, odvisno od sistemskih virov, ki so nam na voljo.



Slika 6: **Primer poročila orodja IBM Support Assistant**

3 SKLEP

V prispevku smo obravnavali problem obremenitvenega testiranja. Najprej smo predstavili okolje, ki smo ga izbrali za testiranje z orodjem JMeter, nato smo predstavili metodologijo testiranja ter sistemsko programsko opremo, s katero smo si pomagali pri analizi. Na koncu smo podali rezultate testiranja in analize. Ključnega pomena pri testiranju je bilo napisati testne scenarije, ki zagotavljajo čim bolj realno simulacijo obremenitev. Dobro napisani testni scenariji, izvedeni testi in opravljena analiza nam lahko kasneje služijo kot referenca ob spremembah sistemske oz. aplikativne programske opreme, novih virtualnih strojev ter spremembah na strojni opremi. Dobre testne scenarije mora napisati oseba, ki zelo dobro pozna sistem. Pri delu se je izkazalo, da je dobra praksa tudi sistematično zbiranje in označevanje poročil med testiranjem in odlaganje poročil v SVN.

Način, da vsako serijo testiranja označimo z edinstveno identifikacijsko številko, se je izkazal kot ključen. Testne scenarije in testiranje lahko uporabimo tudi kot analizo »kaj če«, ko s spreminjanjem parametrov testa odgovorimo recimo na vprašanje, kaj se zgodi, če v istem časovnem obdobju dela podvojeno število uporabnikov. Delilnik bremena v prvi vrsti služi kot stroj, ki zahtevke deli med zalednimi strežniki. Med testiranjem smo prišli do ugotovitve, da z raznimi časovnimi omejitvami (angl. timeout) ščitimo zaledne strežnike pred zasičenostjo. Postavitev 2N se je v praksi pokazala kot primerna postavitev za testno arhitekturo. V produkcijskem okolju bi predlagali postavitev 2N + 1, v nekaterih primerih celo 3N, tako da bi dva sklopa postavili na primarno lokacijo, tretji sklop pa na sekundarno lokacijo kot nadomestni informacijski center, geografsko ločeno. Na začetku testiranja je bilo ugotovljeno, da so javanske kopice mnogo premajhne za močno obremenitev, tako da smo na vseh javanski strojih močno dvignili začetne vrednosti. Orodje JMeter se je izkazalo za zrelo orodje, ki ga lahko postavimo ob bok podobnim komercialnim orodjem. Vedenje, kako se informacijski sistem obnaša zdaj in kako se bo obnašal v prihodnosti, je zaželeno, tako se je obremenitveno testiranje izkazalo nujno kot proces, ki ga je treba vpeljati v poslovanje.

4 LITERATURA

[1] Apache JMeter. (2015). Dostopno 11. 12. 2015 na http://jmeter.apache.org/.

[2] Apache Subversion. (2015). Dostopno 11. 12. 2015 na https://subversion.apache.org/.

[3] Chan, H. A. (1995). The benefits of stress testing. *IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A, 18*(1), 23–29. http://doi.org/10.1109/95.370730.

[4] Državni računalniški oblak, MJU. (2015). Dostopno 11. 12. 2015 na http://www.mju.gov.si/si/delovna_podrocja/informatika/drzavni_racunalniski_oblak/.

[5] Erinle, B. (2013). *Performance testing with JMeter 2.9*. Birmingham, UK: Packt Pub.

[6] Gomez Saez, S., Andrikopoulos, V., Leymann, F. in Strauch, S. (2015). Design Support for Performance Aware Dynamic Application (Re-)Distribution in the Cloud. *IEEE Transactions on Services Computing*, *8*(2), 225–239. http://doi.org/10.1109/TSC.2014.2381237.

[7] Halili, E. H. (2008). *Apache JMeter: a practical beginner's guide to automated testing and performance measurement for your websites*. Birmingham, UK: Packt Publ.

[8] IBM Support Assistant. (2015). [CT757]. Dostopno 14. 12. 2015 na https://www-01.ibm.com/software/support/isa/.

[9] IBM System z10 EC. (2013). [CT555]. Dostopno 11. 12. 2015 na http://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_sm/1/897/ENUS2097-_h01/index.html&lang=en&request_locale=en.

[10] Informacijski sistem Carinske uprave RS. (2015). Dostopno 11. 12. 2015 na http://www.fu.gov.si/carina/.

[11] Iyer, L. S., Gupta, B. in Johri, N. (2005). Performance, scalability and reliability issues in web applications. *Industrial Management & Data Systems*, *105*(5), 561–576. http://doi.org/10.1108/02635570510599959.

[12] Jing, Y., Lan, Z., Hongyuan, W., Yuqiang, S. in Guizhen, C. (2010). JMeter-based aging simulation of computing system. V Q. Luo, & X. Ming (ur.), *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE 2010) (vol. 5)*, Changchun (str. 282–285). IEEE. http://doi.org/10.1109/CMCE.2010.5609969.

[13] Kiran, S., Mohapatra, A. in Swamy, R. (2015). Experiences in performance testing of web applications with Unified Authentication platform using JMeter. V H. A. Bin Sulaiman (ur.), *Technology Management and Emerging Technologies (ISTMET), 2015 International Symposium on*, Langkawai Island, Malaysia (str. 74–78). IEEE. http://doi.org/10.1109/ISTMET.2015.7359004.

[14] Kljajić, M. (1994). *Teorija sistemov*. Kranj, Slovenija: Moderna organizacija.

[15] Levine, S. (2015). *Red Hat Enterprise Linux 7 Load Balancer Administration*. Red Hat Inc. Dostopno 9. 12. 2015 na https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/Load_Balancer_Administration/Red_Hat_Enterprise_Linux-7-Load_Balancer_Administration-en-US.pdf.

[16] Redundancy: N+1, N+2 vs. 2N vs. 2N+1. (2014). Dostopno 12. 12. 2015 na https://www.datacenters.com/news/redundancy-n1-vs-2n/.

[17] SUSE LLC and contributors. (2015). *SUSE Linux Enterprise Server 11 SP4 System Analysis and Tuning Guide*. SUSE LLC.

[18] The OSI Model's Seven Layers Defined and Functions Explained. (2014). Dostopno 9. 12. 2015 na https://support.microsoft.com/en-us/kb/103884.

[19] Wu, Q. in Wang, Y. (2010). Performance Testing and Optimization of J2EE-Based Web Applications. V Z. Hu in Z. Ye (ur.), *Proceedings, The Second International Workshop on Education Technology and Computer Science (ETCS 2010), (vol. 2)*, Wuhan, China (str. 681–683). Los Alamitos, CA: IEEE. http://doi.org/10.1109/ETCS.2010.583.

▪

Primož Kastelic je več kot sedemnajst let zaposlen v podjetju 3GEN kot sistemski inženir. Ukvarja se z integracijo informacijskih sistemov, njihovo varnostjo, zanesljivostjo in razpoložljivostjo. V podjetju tudi vodi sistemsko skupino za administracijo operacijskih sistemov in upravljanje z aplikacijskimi strežniki.

▪

Mirjana Kljajić Borštnar je docentka za področje informacijskih sistemov na Fakulteti za organizacijske vede Univerze v Mariboru. Na dodiplomskem in podiplomskem študiju je nosilka številnih predmetov. Njena raziskovalna področja so večkriterijsko modeliranje, simulacijski modeli za podporo odločanju, odkrivanje znanja iz podatkov, sistemi za podporo skupinskemu odločanju in ekspertni sistemi. Za objavljene izsledke eksperimentov na področju raziskovanja odločanja skupin s pomočjo interaktivnih simulacijskih modelov je s soavtorji prejela več mednarodnih priznanj. Kot avtorica ali soavtorica je objavljala v mednarodnih znanstvenih revijah in konferencah. Sodelovala je v več evropskih in nacionalnih projektih. Je članica programskega odbora blejske eKonference, konference o razvoju organizacijskih znanosti, WorldCIST in drugih, kot recenzentka pa sodeluje na mednarodnih in domačih konferencah ter revijah.

▪

Robert Leskovar je redni profesor za področje informacijskih sistemov in kakovosti na Fakulteti za organizacijske vede Univerze v Mariboru. Njegovo raziskovalno področje obsega kakovost in testiranje programske opreme, večkriterijsko odločanje in simulacijo sistemov.