Decentralizirano in odporno dinamično posodabljanje mikrostoritev

Jan Meznarič, Matjaž B. Jurič
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
{jan.meznaric, matjaz.juric}@fri.uni-lj.si

Izvleček

Za zagotavljanje neprekinjenega delovanja aplikacije je potrebno vpeljati dinamično posodabljanje programske opreme, ki med procesom posodobitve ne povzroča izpada storitev. V arhitekturi mikrostoritev za dinamično posodabljanje običajno uporabimo centralizirana orodja za orkestracijo vsebnikov. Članek opisuje izsledke raziskovalnega dela, v katerem razvijamo decentralizirano metodo za dinamično posodabljanje mikrostoritev. Predlagana metoda definira koordinatorje posodobitve za decentralizirano upravljanje procesa posodobitve, ki z mehanizmi za izboljšanje odpornosti na napake izvedejo evalvacijo nove verzije mikrostoritve. Z evalvacijo preprečimo namestitev nedelujoče verzije mikrostoritve, s čimer se izognemo izpadu delovanja aplikacije, posledično pa izboljšamo proces razvoja programske opreme.

Ključne besede: mikrostoritve, dinamično posodabljanje, odpornost na napake, decentralizacija.

Abstract

Applications that require a high availability are updated with dynamic software updating methods, which do not result in any downtime during the update process. Dynamic updates in the microservice architecture are typically coordinated by a centralised container orchestrator. This paper describes the results of the work in progress in which we are developing a decentralised method for dynamic updates of microservices. The proposed method defines update coordinators for the decentralised coordination of the updates process. Update coordinators use fault-tolerance mechanisms to evaluate the newly deployed microservice version. The evaluation prevents the deployment of a faulty microservice version and consequent service outage, all of which improve the software development process.

Keywords: Microservices, dynamic software updating, fault tolerance, decentralisation.

1 UVOD

Sodobne aplikacije morajo zagotavljati visoko stopnjo razpoložljivosti, prav tako pa jih je potrebno stalno posodabljati, na primer zaradi nadgrajevanja funkcionalnosti ali odprave napak. Z uporabo metod dinamičnega posodabljanja lahko dosežemo ničelni čas izpada storitev aplikacije med njenim posodabljanjem, vendar je metode potrebno prilagoditi sistemu in arhitekturi aplikacije. V arhitekturi mikrostoritev za dinamično posodabljanje običajno uporabljamo orodja za orkestracijo vsebnikov, ki so centralizirana in med procesom posodabljanja ne zagotavljajo naprednih mehanizmov za odpornost na napake. V članku opišemo povzetke izvirnega raziskovalnega dela, ki je še v teku, v katerem razvijamo metodo za dinamično posodabljanje mikrostoritev, s katero želimo izboljšati pomanjkljivosti obstoječih pristopov. Cilj raziskovalnega dela je razviti metodo, ki bo postopek dinamičnega posodabljanja koordinirala decentralizirano brez centralnega orodja za orkestracijo in pri tem evalvirala novo verzijo mikrostoritve, s čimer bo preprečila names-

titev nedelujoče verzije in posledično izpad delovanja celotne aplikacije.

2 SODOBNI PRISTOPI K RAZVOJU PROGRAMSKE OPREME

Popularizacija računalništva v oblaku in selitev programske opreme v oblak sta vplivali na način razvoja in arhitekturo programske opreme. S ciljem boljše izkoriščenosti virov in uporabe podpornih oblačnih mehanizmov, kot so samodejno skaliranje, preverjanje vitalnosti in odpornost na napake, so razvili nove arhitekturne pristope, optimizirane za izvajanje v oblaku (Kratzke & Quint, 2017). Posledično so se razvili tudi novi pristopi k razvoju, testiranju in nameščanju programske opreme.

Najbolj razširjen pristop k razvoju programske opreme za izvajanje v oblaku je arhitektura mikrostoritev, v kateri je posamezna aplikacija sestavljena iz večjega števila neodvisnih storitev – mikrostoritev. Razbitje aplikacije na mikrostoritve sledi poslovnim domenam, pri čemer je posamezna mikrostoritev odgovorna za implementacijo ene funkcionalnosti. Posamezna mikrostoritev je pakirana kot vsebnik in nameščena v večjem številu instanc, kar zagotavlja visoko odzivnost in razpoložljivost. Mikrostoritve med sabo komunicirajo preko jasno definiranih vmesnikov na standardnih omrežnih protokolih. Omrežni naslovi instanc mikrostoritev so shranjeni v registru storitev, iz katerega se pridobivajo za potrebe komunikacije med mikrostoritvami. Mikrostoritvene aplikacije so torej distribuirane, njihova modularna zasnova pa omogoča učinkovito skaliranje, pri čemer je posamezne dele aplikacije možno skalirati glede na njihovo obremenjenost (Esposito, Castiglione & Choo, 2016).

Vsaka mikrostoritev ima jasno določeno ekipo, ki je odgovorna za celoten življenjski cikel mikrostoritve, od razvoja in testiranja do nameščanja in vzdrževanja mikrostoritve v vseh okoljih (Zhu, Bass & Champlin-Scharff, 2016). Zaradi neodvisnosti med posameznimi mikrostoritvami, majhnih razvojnih ekip in odgovornosti ekip za celoten življenjski cikel mikrostoritve razvijalci pogosto uporabljajo kratke razvojne cikle s pogostimi namestitvami novih verzij v produkcijsko okolje. Razvojni cikli so podprti z orodji za avtomatizacijo, ki zagotavljajo zvezno integracijo in nameščanje, pri čemer se za nameščanje novih verzij uporablja koncept dinamičnega posodabljanja, ki med prehodom na novo verzijo zagotavlja neprekinjeno delovanje mikrostoritve (O'Connor, Elger & Clarke, 2017).

Razbitje aplikacije na mikrostoritve, njihov neodvisen razvoj in uporaba kratkih razvojnih ciklov z manj celovitega testiranja povečajo potencial za vnos nepredvidenih napak. Slednje se lahko pojavijo zaradi nepravilnih sprememb vmesnikov ali izpada delovanja mikrostoritve zaradi nepravilne konfiguracije izvajalnega okolja. Posledica so napake pri komunikaciji med mikrostoritvami, ki morajo delovati kot celota in zagotavljati kompozitne funkcionalnosti, sestavljene iz funkcionalnosti posameznih mikrostoritev. Mikrostoritve morajo posledično imeti visoko stopnjo odpornosti na napake, ki nemoteno delovanje aplikacije zagotavlja tudi v primeru pojava nepredvidenih napak. Na primer, neuspešen klic mikrostoritve ne sme povzročiti izpada delovanja celotne aplikacije (Killalea, 2016). Mehanizmi za povečanje odpornosti na napake vključujejo preverjanje vitalnosti mikrostoritev, samodejne ponovne klice zunanjih odvisnosti, vpeljavo maksimalnega odzivnega časa, prekinjevalce toka, pregrade in določanje alternativnih operacij, ki se izvedejo v primeru napak. Dotični mehanizmi so tako eden izmed ključnih gradnikov arhitekture mikrostoritev, saj predstavljajo protiutež povečanemu potencialu za napake, ki ga vpelje razbitje aplikacije na več neodvisnih izvajalnih enot (Toffetti, Brunner, Blöchlinger, Spillner & Bohnert, 2017).

3 IZZIVI DINAMIČNEGA POSODABLJANJA MIKROSTORITEV

Dinamično posodabljanje programske opreme je proces, v katerem namestimo novo verzijo aplikacije, ne da bi s tem prekinili zagotavljanje njenih storitev (Hicks & Nettles, 2005). V arhitekturi mikrostoritev posodabljamo posamezne mikrostoritve, in sicer tako, da obstoječo verzijo mikrostoritve nadomestimo z novo verzijo.

Obstoječi načini dinamičnega posodabljanja mikrostoritev temeljijo na orodjih za orkestracijo vsebnikov. Ta tipično upravljajo elastičnost aplikacije in zagotavljajo kakovost storitev, lahko pa so zadolžena tudi za dinamično posodabljanje. V našem delu nas zanima izključno vloga orodij za orkestracijo kot upravljalcev dinamičnega posodabljanja. V procesu dinamičnega posodabljanja skrbnik naloži novo verzijo mikrostoritve, orodje za orkestracijo pa zažene vsebnik z novo verzijo, preveri njegovo delovanje, nanj preusmeri omrežni promet ter odstrani vsebnike s prejšnjo verzijo. Orodja za orkestracijo vsebni-

kov predstavljajo centralizirano rešitev za dinamično posodabljanje, pri čemer je nemoteno delovanje sistema odvisno od nemotenega delovanja orodja za orkestracijo. Prvi cilj našega raziskovalnega dela se osredotoča na razvoj metode za decentralizirano koordinacijo dinamičnega posodabljanja mikrostoritev, s katero želimo nadomestiti centralizirane posodobitvene metode orodij za orkestracijo.

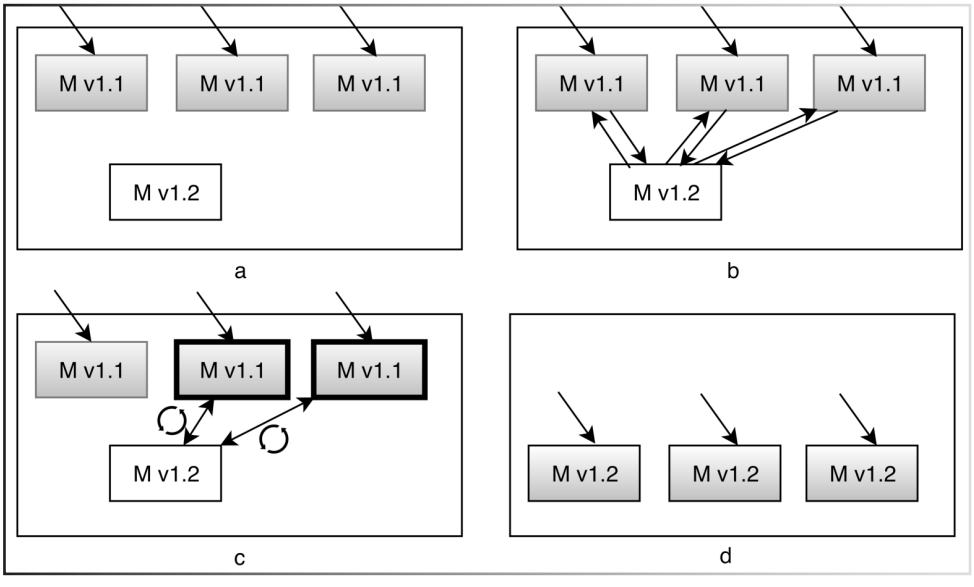
Visoke zahteve po odpornosti na napake v arhitekturi mikrostoritev veljajo tudi za proces dinamičnega posodabljanja, za katerega želimo, da je odporen na napake, ki se lahko pojavijo kot posledica namestitve neustrezne ali nekompatibilne verzije mikrostoritve. Z ustreznimi mehanizmi za povečanje odpornosti na napake želimo preprečiti izpad delovanja aplikacije med posodobitvijo. S tem posledično izboljšamo proces razvoja programske opreme, saj razvijalcem omogočimo, da uporabljajo kratke razvojne cikle s pogostimi namestitvami brez tveganja za izpad delovanja aplikacije zaradi napake v novi verziji. Drugi cilj našega raziskovalnega dela je razviti mehanizme za povečanje odpornosti na napake pri uporabi decentralizirane metode za dinamično posodabljanje mikrostoritev.

4 PREDLOG METODE ZA DECENTRALIZIRANO IN ODPORNO DINAMIČNO POSODABLJANJE MIKROSTORITEV

Za izpolnitev zastavljenih raziskovalnih ciljev predlagamo metodo za decentralizirano in odporno dinamično posodabljanje mikrostoritev. Predlagana metoda je namenjena posodobitvam z manjšimi spremembami mikrostoritev, ki so pogoste v sodobnih razvojnih ciklih. Sem spadajo odprava napak, manjše spremembe ali dopolnitve poslovne logike in dopolnitve vmesnikov, ki ne porušijo združljivosti s prejšnjimi verzijami.

Za decentralizacijo dinamičnega posodabljanja predlagamo vpeljavo komponente za koordinacijo, ki postane sestavni del vsake mikrostoritve. S tem odgovornost za dinamično posodabljanje iz centraliziranih orodij za orkestracijo vsebnikov prestavimo na mikrostoritve. Namesto potrebe po centralnem upravljanju dinamičnega posodabljanja se obstoječa in nova verzija mikrostoritve sami dogovorita o postopku posodobitve, pri čemer posodobitev koordinirajo instance obstoječe verzije, ki jih imenujemo koordinatorji posodobitve. Ker je obstoječa verzija mikrostoritve nameščena v večjem številu instanc, je zagotovljena visoka odpornost na napake, saj se postopek dinamičnega posodabljanja uspešno zaključi tudi ob izpadu enega ali več koordinatorjev posodobitve.

Za povečanje odpornosti na napake v procesu dinamičnega posodabljanja definiramo fazo evalvacije nove verzije. V fazi evalvacije so zahtevki odjemalcev mikrostoritve še naprej usmerjani na instance obstoječe verzije, nova verzija pa je aktivno evalvirana s strani koordinatorjev posodobitve. Ti s klici API-ja za preverjanje vitalnosti testirajo odzivnost nove verzije



Slika 1: **Shema predlagane metode za decentralizirano dinamično posodabljanje z evalvacijo nove verzije mikrostoritve.**

in preverijo rezultate kontrol vitalnosti, ki predstavljajo teste funkcionalnosti mikrostoritve. V primeru pozitivne evalvacije je nova verzija sprejeta, v primeru negativne evalvacije pa zavrnjena brez negativnega vpliva na delovanje celotne aplikacije.

Shema na sliki Slika 1 povzema štiri glavne korake predlagane metode. Pred posodobitvijo so zahtevki odjemalcev mikrostoritve *M* posredovani instancam verzije *v1.1*, saj so te vpisane v registru storitev. Cilj posodobitve je zamenjati instance mikrostoritve *M* verzije *v1.1* z instancami verzije *v1.2*. Instance verzije *v1.1* implementirajo predlagano komponento za koordinacijo, zato lahko prevzamejo vlogo koordinatorjev posodobitve – obstoj instanc s komponento za koordinacijo je predpogoj za začetek procesa dinamičnega posodabljanja s predlagano metodo. V prvem koraku (a) razvijalec v izvajalno okolje namesti vsebnik z novo verzijo mikrostoritve *M* verzije *v1.2*. Instanca verzije *v1.2* ni vpisana v register storitev, zato ne dobiva zahtevkov odjemalcev. V drugem koraku (b) verzija *v1.2* iz registra storitev pridobi naslove instanc mikrostoritve *M* verzije *v1.1* in za koordinatorje posodobitve izbere najbolj odzivne instance. V tretjem koraku (c) izbrani koordinatorji posodobitve s pomočjo API-ja za preverjanje vitalnosti evalvirajo verzijo *v1.2*. Za uspešno evalvacijo mora verzija *v1.2* v določenem času odgovoriti s pozitivnimi statusi posameznih kontrol vitalnosti. Glede na rezultate evalvacije je nova verzija *v1.2* bodisi zavrnjena bodisi sprejeta. Četrti korak (d) prikazuje pozitivno evalvacijo in sprejetje verzije *v1.2*. V primeru pozitivne evalvacije se instance verzije *v1.2* registrirajo v register storitev in posledično začnejo prejemati zahtevke odjemalcev. Instance verzije *v1.1* se iz registra odstranijo in prekinejo svoje izvajanje, njeni vsebniki pa so posledično samodejno ustavljeni. V primeru negativne evalvacije je nova verzija zavrnjena, njene instance prekinejo svoje izvajanje, njeni vsebniki pa so posledično ustavljeni.

Predlagana metoda ne zahteva sprememb obstoječe infrastrukture, mikrostoritve pa ni potrebno izvajati v priviligiranem načinu, saj že imajo vse potrebne pravice za izvajanje metode – pisanje v register storitev, prekinitev lastnega izvajanja in komunikacija z ostalimi instancami iste mikrostoritve, ki je ustrezno zaščitena z mehanizmi avtentikacije. V predlaganem procesu dinamičnega posodabljanja vsaka mikrostoritev tako manipulira le z lastnimi instancami, nasprotno pa imajo orodja za orkestracijo pregled in privilegij upravljanja vseh mikrostoritev v izvajalnem okolju.

Predlagana metoda je definirana v obliki specifikacije, ki jo lahko implementira poljubna mikrostoritev, pri čemer je metoda tipično implementirana kot razširitev mikrostoritvenega ogrodja. Z uporabo razširjenega ogrodja lahko nato razvijalci brez dodatnega dela razvijajo mikrostoritve, ki že vsebujejo implementacijo komponente za koordinacijo in so same sposobne izvajati decentralizirano dinamično posodabljanje z evalvacijo novih verzij. Z razširitvijo obstoječih mikrostoritvenih ogrodij razvijalcem omogočimo enostavno uporabo razvite metode brez potrebe po dodatnem programiranju ali uporabi namenskih centraliziranih orodij za orkestracijo vsebnikov.

5 NAČRT EVALVACIJE IN PREDVIDENI REZULTATI

Predlagano metodo za decentralizirano in odporno dinamično posodabljanje mikrostoritev bomo evalvirali s prototipno implementacijo v obliki razširitve izbranega odprtokodnega ogrodja za razvoj mikrostoritev. Razširjeno ogrodje bomo nato uporabili za izdelavo testne aplikacije, ki jo bomo evalvirali s simulacijami procesa dinamičnega posodabljanja.

V preliminarni evalvaciji smo pripravili prvo prototipno implementacijo predlagane metode ter jo evalvirali s simulacijami dinamičnega posodabljanja preproste mikrostoritvene aplikacije. Pokazali smo, da predlagana metoda uspešno izvede proces dinamičnega posodabljanja brez centraliziranega orodja za orkestracijo. V nadaljnji evalvaciji bomo izboljšali prototipno implementacijo predlagane metode ter definirali več kompleksnejših simulacij. Simulirali bomo nedelujoče koordinatorje posodobitve, s čimer želimo pokazati, da metoda deluje tudi v primeru enega ali več nedelujočih koordinatorjev. Dokazati želimo, da predlagana metoda izboljša robustnost dinamičnega posodabljanja mikrostoritev, zato bomo simulirali namestitve neustreznih verzij mikrostoritev in merili pravilnost evalvacije ter sprejetja ali zavrnitve nove verzije. Želimo se prepričati še, da predlagana metoda nima negativnih vplivov na učinkovitost izvajanja dinamičnih posodobitev, zato bomo merili čas, potreben za vzpostavitev koordinatorjev posodobitve, čas, potreben za izvedbo evalvacije nove verzije, in celoten čas izvedbe posodobitvenega procesa. Pričakujemo, da se proces posodobitve zaključi v enakem ali krajšem času kot pri uporabi centraliziranih orodij za orkestracijo, torej v rangu

nekaj sekund, pri tem pa znatno izboljša odpornost na napake, saj proces posodobitve deluje tudi v primeru izpada koordinatorjev posodobitve ali namestitve neustrezne verzije mikrostoritve.

6 SKLEP

V članku smo povzeli izsledke raziskovalnega dela, v katerem razvijamo metodo za dinamično posodabljanje mikrostoritev, ki deluje brez potrebe po centraliziranem orodju za orkestracijo vsebnikov in izboljšuje odpornost na napake v fazi dinamičnega posodabljanja. Predlagana metoda razbremenjuje razvijalce programske opreme, saj jim omogoča uporabo kratkih razvojnih ciklov s pogostimi namestitvami v produkcijsko okolje, pri čemer morebitna namestitev nekompatibilne ali nedelujoče verzije mikrostoritve ne more povzročiti izpada delovanja celotne aplikacije.

V nadaljevanju raziskave nameravamo definirati vse podrobnosti predlagane metode ter jo evalvirati s pomočjo prototipne implementacije in simulacij. Pričakujemo, da bo razvita metoda pozitivno vplivala na odpornost na napake v procesu dinamičnega posodabljanja mikrostoritev.

LITERATURA

[1] Esposito, C., Castiglione, A., & Choo, K. K. R. (2016). *Challenges in delivering software in the cloud as microservices.* IEEE Cloud Computing, 3(5), 10–14. https://doi.org/10.1109/MCC.2016.105

[2] Hicks, M., & Nettles, S. (2005). *Dynamic software updating. ACM Transactions on Programming Languages and Systems* (TOPLAS), 27(6), 1049-1096. https://doi.org/10.1145/1108970.1108971

[3] Killalea, T. (2016). *The hidden dividends of microservices.* Communications of the ACM, 59(8), 42–45. https://doi.org/10.1145/2948985

[4] Kratzke, N., & Quint, P. C. (2017). *Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study.* Journal of Systems and Software, 126, 1–16. https://doi.org/10.1016/j.jss.2017.01.001

[5] O'Connor, R. V., Elger, P., & Clarke, P. M. (2017). *Continuous software engineering—A microservices architecture perspective.* Journal of Software: Evolution and Process, 29(11), e1866. https://doi.org/10.1002/smr.1866

[6] Toffetti, G., Brunner, S., Blöchlinger, M., Spillner, J., & Bohnert, T. M. (2017). *Self-managing cloud-native applications: Design, implementation, and experience.* Future Generation Computer Systems, 72, 165–179. https://doi.org/10.1016/j.future.2016.09.002

[7] Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). *DevOps and its practices.* IEEE Software, 33(3), 32–34. https://doi.org/10.1109/MS.2016.81

▪

Jan Meznarič je asistent in raziskovalec na UL FRI. Raziskovalno se ukvarja z mikrostoritvami in ostalimi koncepti cloud-native arhitekture, s poudarkom na zvezni integraciji, odpornosti na napake, centralni konfiguraciji, odkrivanju storitev in zbiranju metrik. Raziskovalno in aplikativno sodeluje pri večjem številu projektov in je aktivno udeležen pri razvoju odprtokodnega ogrodja za razvoj mikrostoritev KumuluzEE.

▪

Matjaž B. Jurič je predstojnik Laboratorija za integracijo informacijskih sistemov na UL FRI in mentor start-up podjetij. Je avtor 17 knjig, izdanih pri mednarodnih založbah ter več kot 600 drugih publikacij. Vodil je številne raziskovalne in aplikativne projekte, ponaša pa se tudi s prestižnimi nazivi Java Champion, IBM Champion in Oracle ACE Director. Prejel je več mednarodnih nagrad, med drugim nagrado za najboljšo SOA knjigo (New York), nagrado za najboljši SOA projekt v telekomunikacijah (Las Vegas), nagrado Java Duke's Choice Award Winner (San Francisco) za najboljšo inovacijo v Javi, nagrado za najboljši znanstveni članek s področja storitev, nagrado za najboljšega raziskovalca po mnenju industrije in Zlato plaketo za izjemne zasluge pri razvijanju znanstvenega ustvarjanja.