Urejanje in dostava video posnetkov z oblačno arhitekturo mikrostoritev

Uroš Zoretič[1], Grega Jakus[2]

[1]*Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, Ljubljana, Slovenija*
[2] *Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška cesta 25, Slovenija*
E-pošta: uros.zoretic123@gmail.com, grega.jakus@fe.uni-lj.si

Abstract. *Fast distribution of multimedia on the Web, including video content, is crucial to achieve the desired visibility of an organization. To accomplish this goal, several patents and commercial web video editing products have been developed. They are, however, hard to integrate into state-of-the-art content delivery systems based on cloud computing technologies.*

The building blocks of cloud-based applications are microservices. Cloud-based applications usually have complex topology and run in an unpredictable cloud environment. Therefore, each microservice must be virtualized for efficient use of resources and managed by orchestration platforms. To ensure reliable communication among microservices, service meshes which also provide service discovery, load balancing and request routing, are used.

To improve the efficiency of editing and delivery of video clips, we developed a cloud-based application with microservices as its building blocks. Each microservice manages an individual task, e.g. uploading, cutting, creating, streaming, and storing video clips and their metadata. The application was developed using latest popular technologies for communication among microservices, virtualization, orchestration and monitoring.

1 Uvod

Učinkovita in hitra distribucija video posnetkov na različna spletišča je ključnega pomena z vidika prepoznavnosti posameznika in podjetij. Video posnetke je običajno pred objavo potrebno urediti in jih pretvoriti v format, primeren za predvajanje na spletu. Za urejanje posnetkov lahko uporabimo klasična montažna orodja, kot so AVID, Adobe Premiere Pro in Final Cut Pro X. Če želimo zagotoviti predvajanje videa na spletu brez zatikanja, z minimalno zakasnitvijo in možnostjo prilagajanja njegove kvalitete razpoložljivi pasovni širini, je potrebno video posnetke predvajati po delčkih (ang. »chunks«), ki se med predvajanjem v ozadju ločeno prenašajo s strežnikov [1]. S prihodom standarda HTML5 (Hypertext Markup Language revision 5) se video delčki najpogosteje prenašajo s protokolom HTTP (Hypertext Transfer Protocol). Za opis, kje na strežnikih se posamezni video delčki nahajajo, se uporabljajo indeksne datoteke v formatu XML (Extensible Markup Language). Najpogosteje uporabljeni protokoli za pretočno predvajanje video posnetkov na spletu so MSS (Microsoft Smooth Streaming), MPEG-DASH (Moving Picture Experts Group – Dynamic Adaptive Streaming over Hypertext Transfer Protocol) in HLS (Hypertext Transfer Protocol Live Streaming) [1, 2].

Slabost klasičnih orodij in tehnik za urejanje in dostavo video posnetkov je njihova neučinkovitost v primeru, ko je potrebno na spletu objaviti posnetke, ki so bili že predhodno pripravljeni za objavo na platformah VOD (Video On Demand). Čeprav je takšne video posnetke za ponovno objavo navadno potrebno le malenkost spremeniti, je čas njihove obdelave in objave približno sorazmeren dolžini posnetka, kar je zelo neučinkovito.

Dejstvo, da so bili videi že pripravljeni za objavo na spletu, lahko izkoristimo s spletnim urejevalnikom, ki omogoča neposredno manipulacijo s posameznimi video delčki. Na tem področju obstajajo patenti [3, 4] in komercialne rešitve, kot je Typito. Problem obojih je, da so izdelani po klasičnem vzorcu odjemalec-strežnik in bi jih bilo težko razširiti, nameščati, vzdrževati in integrirati v obstoječe sisteme za dostavo vsebin.

Namen prispevka je predstaviti sistem za urejanje in dostavo video posnetkov, ki temelji na oblačni arhitekturi mikrostoritev in rešuje probleme obstoječih rešitev, saj skrajšuje procesa montaže in dostave za splet pripravljenih video posnetkov.

2 Oblačna arhitektura mikrostoritev

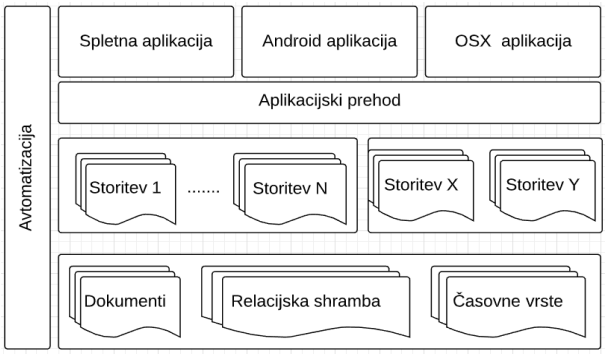
Aplikacije, ki se izvajajo v oblaku, so porazdeljene, prilagodljive, razširljive, odporne na napake, visoko razpoložljive in največkrat sestavljene iz samostojnih enot (mikrostoritev) [5]. Za doseganje omenjenih lastnosti je ključnega pomena ustrezno nadziranje in upravljanje z mikrostoritvami, ki te aplikacije sestavljajo. V nadaljevanju so opisani pomembnejši elementi oblačne arhitekture mikrostoritev.

2.1 Mikrostoritve

Mikrostoritve so gradniki aplikacije, specializirani za določeno funkcionalnost. Lahko so ustvarjene na zahtevo, horizontalno razširljive (več primerkov iste mikrostoritve si razdeli obremenitev), odporne na napake in šibko sklopljene [6]. To pomeni, da posamezna mikrostoritev ni odvisna od delovanja ostalih, temveč so zanjo pomembni le podatki, ki jih prejme in odda. Vsaka mikrostoritev je običajno upravljana neodvisno od ostalih. Arhitektura aplikacije, sestavljena iz mikrostoritev, je prikazana na Sliki 1.

Ker se obremenitev mikrostoritev nenehoma spreminja, se lahko ob velikem številu zahtevkov odzivni časi močno povečajo. Zato je potrebno

zagotoviti prilagajanje števila primerkov mikrostoritve količini zahtevkov ter mednje porazdeliti obremenitve (ang. »load balancing«), tako da vsak primerek opravi približno enako količino dela. Uporabniškim aplikacijam želimo običajno skriti kompleksnost zalednega sistema, za kar uporabimo aplikacijske prehode [6].



Slika 1. Arhitektura aplikacije, sestavljene iz mikrostoritev [6].

Da je aplikacija v oblačnih okoljih odporna na napake, prilagodljiva, horizontalno razširljiva in visoko razpoložljiva poskrbita virtualizacija in orkestracija.

2.2 Virtualizacija mikrostoritev

Virtualizacija omogoča neodvisno upravljanje mikrostoritev z izolacijo, več-najemniškim modelom (ang. »multitenancy«) ter razdeljevanjem in rezervacijo virov, kot so procesorske zmogljivosti ter delovni in trajni pomnilnik. Klasičen način virtualizacije predstavlja uporaba t.i. navideznih strojev (ang. »Virtual Machine«, VM), ki temeljijo na virtualizaciji strojne opreme. Za virtualizacijo mikrostoritev so navidezni stroji neučinkoviti, saj se prepočasi zaženejo in porabijo preveč sistemskih virov [7].

Primernejša je uporaba t.i. *lahke virtualizacije* (ang. »lightweight virtualization«) z vsebniki. Ta se zažene hitreje, saj učinkoviteje izrablja sistemske vire. Te si namreč vsebniki medsebojno delijo, medtem ko ima posamezen navidezni stroj svoje vire izolirane. Vsebniki omogočajo tudi boljšo prenosljivost in interoperabilnost, saj skrijejo kompleksnost in raznolikost programskih jezikov, ogrodij, arhitekturnih vzorcev, vmesnikov in operacijskih sistemov [7].

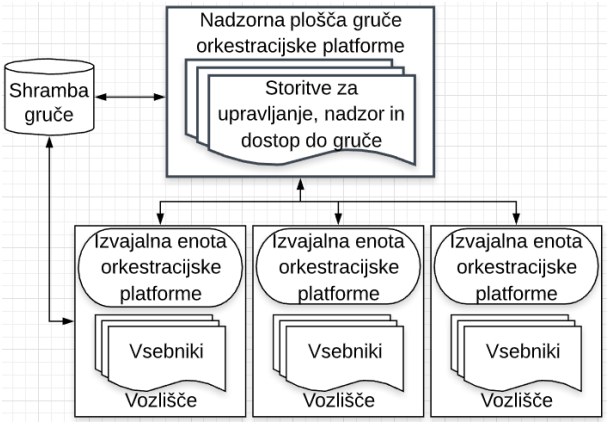
Za različne operacijske sisteme obstajajo različne lahke virtualizacijske tehnologije. Za operacijski sistem Windows je razširjena tehnologija Sandboxie, na Linuxu pa lahko uporabimo tehnologije Docker, LXC in OpenVZ. Najbolj razširjeni so vsebniki Docker, ki smo jih uporabili tudi v naši aplikaciji.

2.3 Orkestracija mikrostoritev

V produkcijskem okolju je vsebnikov pri lahki virtualizaciji zelo veliko. Ker so lahko zaradi različnih napak vsebniki v nekem trenutku neodzivni, jih je za zagotovitev nemotenega delovanja aplikacije potrebno učinkovito upravljati. Ker bi bilo ročno upravljanje zamudno, uporabimo t. i. *orkestracijske platforme*.

Orkestracija vsebnikov omogoča učinkovito upravljanje z viri, samodejno horizontalno razširljivost vsebnikov, visoko razpoložljivost, majhne režijske stroške, deklarativni model upravljanja z nastavitvenimi datotekami, odkrivanje storitev v gruči (ang. »cluster«), razdeljevanje obremenitve, preverjanje zdravja, samozdravljenje, posodobitve in nadgradnje [8].

Gruče orkestracijskih platform običajno sestavljajo nadzorna plošča, ki nadzoruje delovanje sistema, shramba podatkov za njeno konfiguracijo in posamezna vozlišča (ang. »node«). V vozliščih izvajalne enote različnih orkestracijskih platform skrbijo za pravilno delovanje vsebnikov, s katerimi so virtualizirane mikrostoritve [8]. Splošna arhitektura gruče orkestracijskih platform je prikazana na Sliki 2.



Slika 2. Arhitektura gruče orkestracijskih platform [8].

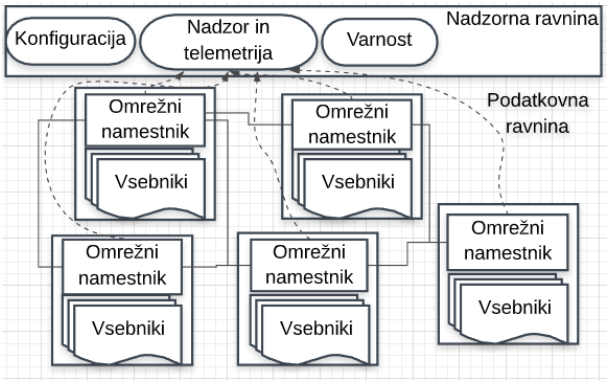
Odprtokodne rešitve za orkestracijo vsebnikov vključujejo Apache Mesos, CoreOS, OpenShift, Docker Swarm in Kubernetes. Slednjo smo uporabili tudi mi.

2.4 Storitvene mreže

Pri kompleksnejših oblačnih aplikacijah orkestracija vsebnikov ni dovolj, saj je potrebno zagotoviti zanesljivo komunikacijo skozi kompleksno topologijo mikrostoritev [9]. To lahko zagotovimo na več načinov, in sicer s programskimi knjižnicami v okviru vsake mikrostoritve, vozliščnimi agenti ali z inteligentnimi omrežnimi namestniki (ang. »network proxy«), ki se izvajajo poleg vsake namestitvene enote [9]. Najbolj uveljavljene storitvene mreže so Airbnb Synapse, AWS App Mesh, Linkerd2 in Istio.

Na splošno so najboljša možnost za namestitev storitvene mreže inteligentni omrežni namestniki, saj so neodvisni od različnih programskih jezikov, ne zahtevajo spreminjanja programske kode v mikrostoritvah in ne vnašajo dodatne zakasnitve pri komunikaciji [9]. Arhitektura takšne storitvene mreže je prikazana na Sliki 3.

Storitvena mreža je sestavljena iz podatkovne in nadzorne ravnine. Omrežni namestniki na podatkovni ravnini skrbijo za odkrivanje storitev, usmerjanje, porazdeljevanje obremenitve, overjanje, pooblastitve in spremljanje delovanja vsebnikov. Nadzorna ravnina upravlja s storitveno mrežo, zbira telemetrijske metrike in konfigurira omrežne namestnike za usmerjanje prometa [9].



Slika 3. Arhitektura storitvene mreže z namestitvijo omrežnih namestnikov [9].

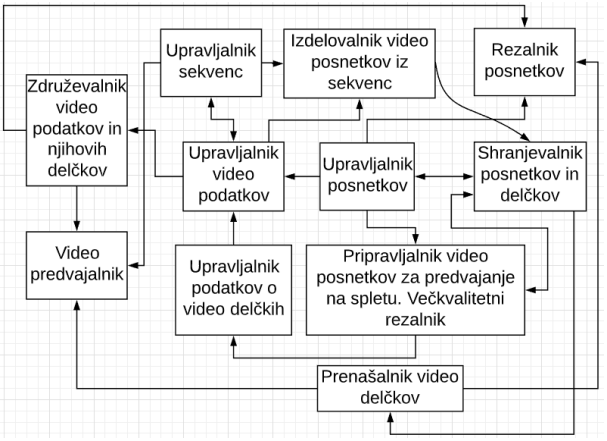
3 Oblačna aplikacija za urejanje in dostavo video posnetkov

Namen predstavljene aplikacije je učinkovito urejanje video posnetkov, pripravljenih za predvajanje na spletu z uporabo oblačnih arhitekturnih vzorcev, predstavljanih v prejšnjem poglavju. Aplikacija omogoča ogled knjižnice video posnetkov, njihovo nalaganje in razrez ter izbiro ali ustvarjanje projektov, v katerih se urejajo video posnetki na časovnicah ustvarjenih sekvenc. Na časovnico sekvence lahko poljubno dodajamo ali odstranjujemo posnetke. Po zaključku urejanja lahko video objavimo, tako da postane dostopen izven izbranega projekta ali pa ga prenesemo v lokalno shrambo. Vse opravljeno delo se sproti shranjuje, tako da lahko uporabnik z urejanjem nadaljuje kasneje.

3.1 Gradniki aplikacije

Izdelana aplikacija je sestavljena iz mikrostoritev. Te izvajajo nalaganje, rezanje in shranjevanje video posnetkov, prepoznavanje njihove vsebine, shranjevanje in pridobivanje metapodatkov o posnetkih in njihovih delčkih, pripravo datotek v zapisu m3u8 po protokolu HLS iz metapodatkov posnetkov, povezovanje video delčkov s pripadajočimi posnetki, izdelavo novih posnetkov iz obstoječih delčkov, pretvorbo posnetkov v različne kvalitete predvajanja ter objavo in predvajanje njihovih delčkov. Na Sliki 4 so prikazane ključne mikrostoritve in povezave med njimi.

Za komunikacijo med mikrostoritvami uporabljamo protokol gRPC (Google Remote Procedure Call), podatki pa so zapisani v formatu Protocol Buffers. gRPC za prenos podatkov uporablja protokol HTTP/2 [10]. S tem je gRPC učinkovitejši od klasičnega vzorca REST API (Representational State Transfer Application Programming Interface), ki ga sicer uporabljamo za uporabniške zahtevke, kot sta predvajanje video posnetka in prenašanje njegovih delčkov.



Slika 4. Mikrostoritvena arhitektura urejevalnika video posnetkov.

Mikrostoritve ki izvajajo izrazito asinhrone operacije, kot je priprava videov za predvajanje na spletu, vsebujejo čakalno vrsto za sporočila. V ta namen smo uporabili RabbitMQ, ki uporablja protokol AMQP (Advanced Message Queuing Protocol).

Pri analizi vsebine video posnetkov smo uporabili storitev Google Video AI. Zaradi velikega števila mikrostoritev in povezav med njimi smo za združevanje rezultatov poizvedb po različnih mikrostoritvah uporabili jezik GraphQL (Graph Query Language). S tem jezikom lahko zahtevamo že filtrirane podatke, s čimer optimiziramo njihovo količino pri prenosu med mikrostoritvami in uporabniškimi vmesniki.

3.2 Delovanje aplikacije

Pred začetkom urejanja v aplikacijo naložimo video posnetke. Za to je zadolžena storitev »Upravljalnik posnetkov«, ki s klicem storitve »Upravljalnik video podatkov« ustvari podatke o posnetku. S klicem storitve »Shranjevalnik posnetkov in delčkov« posnetek naloži v shrambo in pošlje zahtevek v sporočilno vrsto za njegovo pripravo na predvajanje na spletu. Posnetek se z orodjem FFmpeg razreže na pet-sekundne MPEG2-TS (MPEG2 Transport Stream) delčke različnih kvalitet.

Iz pripravljenih video posnetkov lahko izrežemo določen časovni izsek tako, da storitvi »Rezalnik posnetkov« sporočimo začetni in končni čas izseka. »Rezalnik posnetkov« iz izseka ustvari nov posnetek, tako da po časovni oznaki logično poveže obstoječe MPEG2-TS video delčke. Začetni in kočni video delček izbranega časovnega izseka videa se po potrebi razreže z orodjem FFmpeg in shrani s storitvijo »Shranjevalnik posnetkov in delčkov«. Ker je z rezalnikom video posnetkov potrebno razrezati samo začetni in končni pet-sekundni delček izbranega izseka posnetka, prihranimo na času obdelave. Izrezani video je nato brez dodatne obdelave pripravljen za predvajanje.

Za ustvarjanje novih posnetkov je najprej potrebno narediti sekvenco. Storitev »Upravljalnik sekvenc« pri dodajanju ali brisanju posnetkov iz sekvence pridobi od

storitve »Upravljalnik video podatkov« indeks posnetka in ga poveže z novo sekvenco.

Med urejanjem sekvence je mogoče pogledati nastajajoči posnetek s klicem storitve »Video predvajalnik« in podanim indeksnim parametrom sekvence. Storitev »Video predvajalnik« nato iz storitve »Upravljalnik sekvenc« pridobi indekse posnetkov v sekvenci. Nato s klicem storitve »Združevalnik video podatkov in njihovih delčkov« pridobi njihove metapodatke, ki se nahajajo v storitvah »Upravljalnik video podatkov« in »Upravljalnik podatkov o video delčkih«. Iz pridobljenih podatkov »Video predvajalnik« ustvari opisno datoteko m3u8 po protokolu HLS in jo posreduje odjemalcu. Ta nato video posnetek predvaja tako, da ga prenaša po delčkih iz storitve »Prenašalnik video delčkov«.

Sekvenco lahko po zaključku urejanja objavimo s pomočjo »Upravljalnika sekvenc«, ki zahtevo posreduje storitvi »Izdelovalnik video posnetkov iz sekvenc«. Objava posnetka je časovno učinkovita, saj storitev »Izdelovalnik video posnetkov iz sekvenc« pridobi metapodatke o sekvenci in vanjo vključenih posnetkih, jih indeksira v nov posnetek in to sporoči storitvi »Upravljalnik video podatkov«. Ustvarjeni video posnetek je za ogled na voljo s klicem storitve »Video predvajalnik« s podanim indeksnim parametrom videa.

3.3 Izvajanje mikrostoritev na oblačnih platformah

Ker mikrostoritve izvajamo na oblačnih platformah Microsoft Azure, Google Cloud in Amazon Web Service, smo uporabili rešitve za oblačne aplikacije.

Mikrostoritve smo zapakirali v vsebnike z lahko virtualizacijo Docker. Vsebnike upravlja orkestracijska platforma Kubernetes. Za zanesljivo komunikacijo, odkrivanje storitev, usmerjanje, porazdeljevanje obremenitve in opazovanje delovanja skrbi storitvena mreža Istio. Njeno stanje in konfiguracijo lahko spremljamo z orodjem Kiali. Metrike in dnevniške zapise delovanja vsebnikov zbira Prometheus. Vizualiziramo jih s storitvijo Grafana. Zahtevke spremljamo z orodjem Jaeger. Prometheus, Grafana in Jaeger so storitve na nadzorni ravnini storitvene mreže Istio. Postopek gradnje in namestitve smo avtomatizirali po principu DevOps (Development and Operations) z CI/CD (Continuous Integration and Continuous Delivery) [11] orodjem Travis CI. Ročna gradnja in namestitev mikrostoritev bi nam vzela preveč časa.

Za prikaz delovanja smo razvili spletno aplikacijo, ki se izvaja na brez-strežniški (ang. »serverless«) storitvi Google Cloud Run. Ker smo pred uporabnikom želeli skriti kompleksnost sistema, smo z enotnim vmesnikom aplikacijskega prehoda Istio Ingress izpostavili zahtevke, ki jih potrebuje spletna aplikacija.

4 Zaključek

Računalništvo v oblaku je postalo de facto standard za razvoj predvsem spletnih aplikacij, o čemer pričata tudi količina različnih odprtokodnih rešitev na portalu Cloud Native Computing Foundation in statistika uporabe oblačnih tehnologij [12].

Montaža in dostava video posnetkov z oblačno arhitekturo mikrostoritev pohitri proces objave za splet pripravljenih posnetkov na različnih spletiščih. Izdelano aplikacijo je mogoče enostavno vzdrževati in integrirati v obstoječe sisteme za dostavo vsebin. Aplikacijo bomo v bližnji prihodnosti prilagodili in uporabili za hitrejšo dostavo vsebin na različna spletišča podjetja Pro Plus, ki se ukvarja s produkcijo video vsebin.

Literatura

[1] Stockhammer, Thomas. “Dynamic adaptive streaming over HTTP --: standards and design principles.” MMSys (2011).

[2] Mueller, Christopher & Lederer, Stefan & Timmerer, Christian. (2012). An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments. 37-42. 10.1145/2151677.2151686.

[3] Ryan Jenee, Simon Rather, Julian Fruman, Wev-based system for video editing, United States Patent, 2. 9. 2014, https://patentimages.storage.googleapis.com/19/dd/75/1121a71203f3c8/US8826117.pdf

[4] Ian Lovejoy, Ken Wang, Kevin Wong, Eric Vossbrinck, Mark Moore, Henry Dall, Web based video editing, United States Patent, 15. 5. 2015, https://patentimages.storage.googleapis.com/af/dd/d6/c052d357586eae/US9032297.pdf

[5] Nane Kratzke, Peter-Christian Quint, Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study, Journal of Systems and Software, Volume 126, 2017, Pages 1-16

[6] Bob Familiar. 2015. Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to deliver SaaS Solutions (1st. ed.). Apress, USA.

[7] C. Pahl and B. Lee, "Containers and Clusters for Edge Cloud Architectures -- A Technology Review," 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, 2015, pp. 379-386, doi: 10.1109/FiCloud.2015.35.

[8] Alex Williams, The State of the Kubernetes Ecosystem - The New Stack, https://lp.google-mkto.com/rs/248-TPC-286/images/TheNewStack_Book1_TheStateOfTheKubernetesEcosystem.pdf

[9] W. Li, Y. Lemieux, J. Gao, Z. Zhao and Y. Han, "Service Mesh: Challenges, State of the Art, and Future Research Opportunities," 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco East Bay, CA, USA, 2019, pp. 122-1225, doi: 10.1109/SOSE.2019.00026.

[10] Vijay Pai, gRPC Design and Implementation, https://platformlab.stanford.edu/Seminar%20Talks/gRPC.pdf

[11] Martin Fowler, Continuous Integration, https://moodle2019-20.ua.es/moodle/pluginfile.php/2228/mod_resource/content/2/martin-fowler-continuous-integration.pdf

[12] CNCF Survey 2019, https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf