Časovni razporejevalniki in brezstrežniško okolje

Uroš Zagoranski
Univerza v Mariboru,
Fakulteta za elektrotehniko, računalništvo in informatiko,
Koroška cesta 46, 2000 Maribor, Slovenija
uros.zagoranski@student.um.si

Povzetek

V prispevku smo se osredotočili na časovne razporejevalnike (ang. cron job schedulers) v brezstrežniškem (ang. serverless) okolju in njihovo zanesljivo uporabo. Primerjali smo razporejevalnike, implementirane s pomočjo zabojnikov s tistimi, ki so gostovani v oblaku z uporabo pristopa funkcije kot storitve. Ugotavljali smo, katere so posebnosti časovnih razporejevalnikov v brezstrežniškem okolju in kdaj je le-te sploh smiselno uporabiti. Na praktičnem primeru smo predstavili, kako jih lahko vključimo v večji sistem in na kakšen način najlažje rešimo morebitne težave, ki jih ob izbiri brezstrežniškega okolja zavestno prevzamemo. Ugotovili smo, da so razporejevalniki v FaaS (ang. Function as a Service) okolju najprimernejši zaradi enostavnega in hitrega razvoja ter nizkih stroškov obratovanja.

Ključne besede brezstrežniška arhitektura · sodobni pristopi implementacije informacijskih rešitev · časovni razporejevalniki · funkcija kot storitev · zabojnik kot storitev

1 Uvod

Tempo življenja se v zadnjih letih veča in zaradi tega smo ljudje vse bolj časovno obremenjeni. S tem se posledično veča tudi potreba po časovnem razporejanju rutin, da se kakšna informacija v presežku le-teh ne izgubi. Že od nekdaj so se razvijali sistemi, ki so omogočali takšno in drugačno razporejanje, a to ni bilo še nikdar enostavnejše, kot je v današnji dobi računalništva v oblaku. Velik nabor ponudnikov oblačnih storitev omogoča izbiro med le-temi, ki pa ravno zaradi velike ponudbe včasih ni trivialna. Izbiro prave storitve otežuje predvsem pomanjkanje analitičnih pregledov in primerjav storitev, ki jih ponujajo različne korporacije. S tem namenom smo želeli predstaviti razlike, prednosti in dodatne izzive, ki nastanejo pri razvoju razporejevalnikov v oblaku z uporabo funkcij kot gonilne sile v primerjavi s tistimi, ki so implementirani v (bolj) klasičnih aplikacijah, s pomočjo orkestracije zabojnikov. Poleg tega smo našteli in predstavili glavne sisteme, ki ponujajo časovno razporejanje v brezstrežniškem okolju in na praktičnih primerih uporabe pojasnili, kateri izmed glavnih ponudnikov je za posamezen primer boljši in zakaj.

Raziskovalni vprašanji, ki smo ju obravnavali sta:

- **Katere so posebnosti časovnih razporejevalnikov v brezstrežniškem okolju?**
- **Kdaj je smiselno uporabiti časovne razporejevalnike brezstrežniškega okolja?**

V nadaljevanju članka so v drugem poglavju predstavljene posebnosti sodobnega pristopa razvoja programskih rešitev, to je z zabojniki in s pomočjo funkcij kot storitev, ter njuna primerjava. Tretje poglavje pokriva tematiko časovnih razporejevalnikov na splošno ter se podrobneje posveča razporejevalnikom v brezstrežniškem okolju, pregledno povzema glavne ponudnike takšnih storitev in navaja primere uporabe. Četrto poglavje predstavlja realen primer uporabe časovnih razporejevalnikov v namen razporejanja opravil v računovodskem servisu. V petem poglavju odgovarjamo na raziskovalna vprašanja in uvajamo diskusijo, v zadnjem poglavju pa povzemamo rezultate raziskave in glavna nova dognanja, pridobljena z njeno pomočjo.

2 Sodobni pristopi implementacije informacijskih rešitev v namen razporejanja opravil

Vse več razvijalcev informacijskih rešitev pomeni večji nabor možnosti in preferenc pri razvoju programske opreme. Zaradi tega raziskovalno podjetje Gartner vsako leto izdela poročilo trendov v informacijsko tehnološkem svetu za prihodnje koledarsko leto. V zadnjem času lahko na tej lestvici vse pogosteje zasledimo besedne zveze kot so umetna inteligenca, obogatena resničnost, analitika, avtomatizacija ipd. [12] Virtualizacija virov in programiranje v oblaku sta se hitro zlili s klasičnimi pristopi implementacije informacijskih rešitev, zato ni čudno, da je rast tega področja in inovativnost rešitev v tej kategoriji v pozitivnem trendu. Med napovedmi za bližnjo prihodnost se v Gartnerjevih raziskavah tako pojavljajo besedne zveze, ki omenjajo zabojnike, njihovo orkestracijo, funkcije kot storitev in brezstrežniški pristop razvoja programske opreme splošno. [6]

Brezstrežniški pristop razvoja programske opreme je model računalništva v oblaku, v katerem ponudnik oblačnih storitev v imenu svojih strank skrbi za strežnike in dodeljevanje strojnih virov. Glavna posebnost takšnega pristopa je, da aplikacija nima dodeljenih računalniških virov, če ta ni v uporabi. To se odraža v nižji ceni delovanja, saj le-ta temelji na dejanski količini sredstev, ki jih aplikacija porabi v določenem časovnem intervalu. [1]

Trenutno stanje brezstrežniških sistemov na trgu je v Gartnerjevih analizah primerjano s predpostavkami za bližnjo prihodnost, prav tako pa je podana finančna ocena

DOI https://doi.org/10.18690/978-961-286-516-0.16
ISBN 978-961-286-516-0

brezstrežniških sistemov v bližnji prihodnosti. Pri tem napovedujejo, da se bo iz lanske globalne kvote prihodkov v višini 465,8 milijonov dolarjev do leta 2024 višina prihodkov kar podvojila, in sicer naj bi znašala 944 milijonov dolarjev. Največjo rast naj bi po ocenah dosegli ravno orkestracija zabojnikov ter brezstrežniški pristop razvoja informacijskih rešitev, kateri tematiki bomo podrobneje opisali v nadaljevanju. [11]

Druga Gartnerjeva raziskava ocenjuje, da bo v prihodnosti drastično narasla tudi uporaba platform, ki kot sprožilce aktivnosti v sistemu uporabljajo najmanjše možne komponente, to so funkcije. Trenutno približno 20 odstotkov globalnih IT podjetij uporablja »funkcijski« (FaaS) način implementacije, do leta 2025 pa bi naj ta številka narasla na kar 50 odstotkov. [4]

Porast interesa uporabe oblačnih infrastruktur se odraža tudi na številu ponudnikov. Med vodilne uvrščamo naslednje:

- Amazon Web Services,
- Google Cloud Platform,
- Microsoft Azure,
- IBM Apache OpenWhisk.

V tekmovanju teh ponudnikov se kot glavni akter že vrsto let pojavlja platforma Amazon Web Services (v nadaljevanju AWS), a v zadnjem času pridobiva vse več tekmecev, med katerimi sta najbolj aktualna predvsem Microsoft Azure in Google Cloud Platform (v nadaljevanju GCP). A če povzamemo obe prej omenjeni področji (zabojnike in funkcije), v ospredje brez konkurence stopi Google, saj si lasti tako orkestracijsko rešitev Kubernetes kot tudi ogromno platformo za programiranje v oblaku, GCP. [7] Podjetja želijo, da se razvijalci osredotočajo predvsem na razvoj programske opreme, ne pa tudi na vzdrževanje, vzpostavitev in druge podporne aktivnosti. Pri zabojnikih zato pogosto zasledimo tudi osebo, ki je zadolžena za skrb in upravljanje s celotno arhitekturo (DevOps), katere pa pri funkcijskem pristopu načeloma ne potrebujemo.

2.1 Zabojniki: Container as a Service (CaaS)

Vse pogostejši trend uporabe zabojnikov za implementacijo informacijskih rešitev se odraža tudi pri nastanku dokaj nove paradigme računalništva. CaaS predstavlja nadgradnjo kategorije IaaS (Infrastructure as a Service), kjer korporacije v najem ponujajo strežnike, virtualne stroje, omrežja in shrambe, pri čemer so njihovi uporabniki odgovorni za upravljanje infrastrukture in nameščanje aplikacij nanjo. Zraven tega CaaS ponuja še zabojniške stroje in funkcionalnosti za orkestracijo le-teh. [19] Za nas je najzanimivejši podrazred CaaSa, KaaS (Kubernetes as a Service), saj je zaradi svoje robustnosti, zrelosti in bogatega nabora funkcionalnosti Kubernetes postal de-facto standard in najpogosteje uporabljena tehnologija, ko je govora o razvoju informacijskih rešitev s souporabo zabojnikov. [13]

Kubernetes je odprtokodni sistem za orkestracijo, ki omogoča avtomatizirano upravljanje z zabojniki, prav tako pa upravlja celoten življenjski cikel zabojnikov, pri čemer uporabniki upravljajo z izvedbo in interakcijo aplikacij. [8] Aplikacije so tipično sestavljene iz več zabojnikov, ki so porazdeljeni po različnih virtualnih in fizičnih gostiteljih. Kubernetes nam pri tem pomaga pri upravljanju s strukturiranjem zabojnikov v tako-imenovane stroke (ang. pods), ki okolje bogatijo z dodatnim nivojem abstrakcije. Na ta način je poenostavljeno razporejanje z viri, prav tako pa časovno razporejanje, ki je za nas najzanimivejše. [10]

Pristop z zabojniki je primeren predvsem za ekipe, ki prisegajo na podrobnejši nadzor nad storitvami kontejnerizacije. Prav tako omogoča avtomatizacijo uvajanja in vračanja v prejšnje stanje, fine nastavitve količine procesorske moči in pomnilniškega prostora, samodejno celjenje zabojnikov, ki so bili pri izvajanju neuspešni ter upravljanje s konfiguracijami. [8]

2.2 Funkcije: Function as a Service (FaaS)

FaaS je nov koncept v oblačnem računalništvu, podoben PaaSu (Platform as a Service) z vidika omogočanja enostavnejšega razvoja programske opreme in razbremenitve razvijalcev upravljanja s strežniki in operacijskimi sistemi. Prednost funkcij v oblaku v primerjavi s prej omenjenima konceptoma je predvsem v poslovnem vidiku. Medtem, ko se pri PaaSu plačuje čas izvajanja niti, se pri FaaSu plačuje izvajalni čas specifične funkcije. S tem se lahko konkretno zvišajo performančne karakteristike ogromnih sistemov, hkrati pa znižajo obratovalni stroški same informacijske rešitve. [1]

Z začetki v letu 2014 z Amazonovo Lambdo je do danes koncept FaaS postal priznan in dobro sprejet način implementacije informacijskih rešitev. Dokazano je veliko bolj skalabilen, elastičen, razvijalcem prijazen ter cenovno ugoden kot predhodne oblačne arhitekture. Poleg AWSove Lambde v to kategorijo uvrščamo tudi Microsoft Azure Functions, Google Cloud Functions, IBM Apache OpenWhisk. [2]

Pristop s funkcijami v oblaku je primeren predvsem za projekte, ki imajo neenakomerno razporeditev izvajanja, izjemno visoke ali neznane zahteve za skaliranje sistema, so vezani na zunanje dogodke, sestavljeni iz kratkoživečih diskretnih funkcij, ali pa lahko s klici operirajo brez stanja. Pogosto jih lahko zasledimo pri mobilnih aplikacijah, IoT senzorskih komponentah in časovno aktiviranih akcijah. [4]

2.2.1 Primerjava obeh skupin

V začetku porasta informacijske tehnologije je postala popularna gradnja monolitnih aplikacij, ki pa se danes vse manj pojavlja na seznamu trendov. V zadnjem času so jih pretežno zamenjale mikro-storitve in posledično se je vpeljala uporaba zabojnikov, s katerimi sta razporejanje in razvoj aplikacij veliko enostavnejša. Zabojniki so se v IT svetu prvič pojavili že pred letom 2000, zato ima ta koncept daljšo in bogatejšo zgodovino, kot funkcije v oblaku. Kljub temu t. i. eksplozija popularnosti obeh konceptov sega v podobno časovno obdobje. Zabojniki so namreč postali izjemno aktualni šele z Dockerjem v letu 2013, FaaS pa z AWSovo Lambdo v letu 2014. [16]

FaaS predstavlja višji nivo abstrakcije kot CaaS. Slednji namreč abstrahira nivoje strojne opreme, virtualizacij-

Tabela 1: Primerjava CaaS in FaaS [9]

	CaaS	**FaaS**
Prenosljivost	Zabojniki predstavljajo standardizirano enoto, zaradi česar je njihova prenosljivost med ponudniki enostavna	Slaba standardizacija, večinoma odvisno od ponudnika, razporejevalniki kljub temu delno standardizirani
Dvig in premik	Obstoječe aplikacije enostavno oviti v zabojnike, če to ni že izvedeno, zabojniki nimajo vpliva na delovanje razporejevalnikov	Ni mogoč, obstoječe aplikacije potrebujejo adaptacijo ali popolno re-implementacijo
Razdrobljenost	Enote ponavadi razdeljene na komponente, ponujena možnost razporejanja specifičnih funkcij, tudi tistih, ki skrbijo za razporejanje	Aplikacija razdrobljena na majhne enote, ki jih je enostavno razporejati (funkcije), idealno za implementacijo razporejevalnikov
Izvajanje	Upravljano s strani razvijalca	Upravljano s strani ponudnika v oblaku
Fleksibilnost	Visoka	Srednja
Primeri uporabe	Celotne spletne aplikacije, mikro-storitve, upravljanje s podatki	Specializirane funkcije, asinhrono procesiranje, dogodkovno vodena arhitektura
Izzivi	Zahteva razumevanje delovanja zabojnikov, težja implementacija razporejevalnikov	Upravljanje odvisnosti, sledenje in upravljanje več mikro-storitev hkrati
Primernost	Učinkovito za gostovanje osnovnih aplikacij, manj primerno za implementacijo razporejevalnikov	Najprimernejše za lansiranje novih aplikacij, izjemno primerno za implementacijo razporejevalnikov

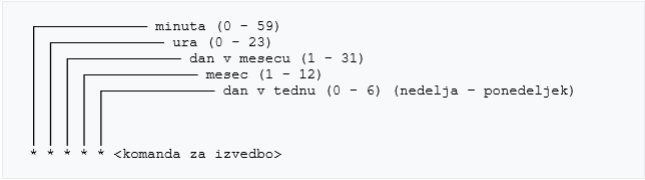
skih strojev ter operacijskega sistema, pri čemer pa prvi na vse prej omenjene nivoje dodaja še abstrakcijo izvedbe programske kode in aplikacije. Obe skupini prenašata odgovornost manipulacije z aplikacijo in funkcijami na uporabnika storitve.

Če na kratko povzamemo celotno tabelo 1, ki predstavlja primerjavo prej omenjenih pristopov lahko ugotovimo, da v splošnem pristop z zabojniki nudi več svobode pri implementaciji, nadzora nad izvajanjem ter enostavno migracijo v primeru, da neka obstoječa aplikacija še ne uporablja takšnega pristopa. Na drugi strani je pristop s funkcijami v oblaku primernejši in lažji za vzdrževanje, je pa manj fleksibilen in slabo standardiziran, zato se tehnične podrobnosti, ki jih je potrebno upoštevati ob implementaciji, razlikujejo med posameznimi ponudniki. Kateri pristop uporabiti je odvisno predvsem od primera uporabe, pri čemer je za spletne aplikacije z mikro-storitvami primernejša izbira CaaS modela, za dogodkovno vodene aplikacije z asinhronim procesiranjem pa se bo bolje izkazal FaaS model.

3 Časovni razporejevalniki

Časovno tempiranje aktivnosti je v današnjem svetu čisto običajna aktivnost, zato je vse pogostejša tudi uporaba sistemov, ki skrbijo za izvajanje določenih funkcij ob specifičnem času, na željeno frekvenco ali pa z določenimi zamiki. Razporejevalniki nas tako v informacijsko tehnološkem svetu spremljajo praktično od začetka. Ti so prisotni v operacijskih sistemih, omrežnih komponentah, konec koncev pa tudi v I/O operacijah (operacijah pisanja in branja).

Medtem, ko se pri vseh prej naštetih komponentah termin razporejanje pojavlja v obliki razporejanja virov, je za našo raziskavo veliko zanimivejše časovno razporejanje opravil. V tradicionalnih sistemih se takšni problemi večinoma rešujejo z aplikacijskimi strežniki in orkestracijo zabojnikov, v novi dobi oblačnega računalništva pa se vse pogosteje pojavlja prej omenjeni pristop FaaS. Za izvedbo specifičnih funkcij ob določenem času v informacijskih rešitvah takšnega tipa skrbijo funkcije, imenovane časovni razporejevalniki, ki v vseh sistemih, ne glede na izbiro tehnološkega sklada, delujejo po enakem principu. Poleg enakega principa imajo tudi enako obliko sprejema podatkov, sestavljeno iz petih spremenljivk in klica funkcije ali komande. Splošna oblika »cron« izraza je prikazana na sliki 1.



Slika 1: Splošna oblika »cron« izraza

Med najpogostejšimi izrazi, ki jih zasledimo v imenih razporejevalnikov sta angleški besedi »cron« in »scheduler«. V sklopu prej opisanih razporejevalnikov najdemo glavne ponudnike takšnih storitev, to so Googlov Kubernetes, Amazonovo orodje Lambda v kombinaciji s storitvijo CloudWatch Events, Googlovi Cloud Scheduler in Scheduled Cloud PubSub ter Microsoftov Azure Time Trigger.

3.1 »Zabojniško« okolje

Zabojnike lahko uporabimo za različne namene, eden pomembnejših je procesiranje in obdelava nalog v ozadju. To je pogost primer uporabe, ki pa je vse prej kot trivialen in odpira vrsto vprašanj, med drugim je vprašljivo sledenje izvajanja, skaliranje in upravljanje s ponovnimi zagoni. Zraven razporejevalnikov z viri, ki so v razdelani arhitekturi z orodji kot sta Docker in Kubernetes neizbežni, se v teh okoljih srečujemo tudi s časovnimi razpore-

jevalniki, ki znatno vplivajo na optimizacijo virov, da je uporaba teh kar se da učinkovita. [14]

Zabojniki sami po sebi omogočajo uporabo časovnih sprožilcev, ki pa z vpeljavo več instanc in posledično kompleksnosti zahtevajo uporabo posvečenega orodja, kot je na primer Kubernetes. Stanje posameznega zabojnika se tako ne more prenašati med preostalimi zabojniki zaradi varnostnih razlogov, zaradi česar je potrebno najti druge načine za izpostavitev skript. Kubernetes prav tako dinamično generira imena zabojnikov, zaradi česar je dostopanje do njih oteženo. [17]

3.2 »Funkcijsko« okolje

Časovni razporejevalniki sami po sebi ne veljajo za najbolj zanesljive. To pride do izraza v funkcijskem brezstrežniškem okolju, kjer jih pogosto najdemo kot popolnoma neodvisne komponente. Njihova edina naloga je, da se izvedejo ob določenem času, zaradi česar se lahko zgodi, da se določena funkcija, ki je bila klicana v sklopu nekega razporejevalnika, izvede neuspešno. Do tega lahko pripelje nekaj dejavnikov, med drugim začasna prekinitev delovanja infrastrukture (ki je v oblaku redka, a mogoča), neuspešna obdelava podatkov, ali pa prekinitev povezave s podatkovno bazo. Ker razporejevalniki v brezstrežniškem okolju kot parametre prejmejo le čas izvedbe, ni potrebna validacija vnesenih parametrov. Iz tega vidika so izjemno zanesljivi, saj (načeloma) ne more priti do napake pri sami izvedbi. Večjo težavo predstavlja dejstvo, da se predpisane funkcije po izvedbi preprosto »ugasnejo« in jih uspešnost zaključka klicane funkcije ne zanima. [15]

Med glavne primere uporabe lahko uvrstimo naslednje:

- vzdrževalna naloga re-indeksiranja podatkovne baze, ki se mora izvesti n-krat v določenem časovnem intervalu,
- deaktivacija poteklih uporabniških računov,
- nadzor nad prostorom pomnilnika,
- kreacija varnostnih kopij občutljivih podatkov,
- upravljanje s predpomnilnikom,
- razpošiljanje glasila ali posebne ponudbe prijavljenim uporabnikom,
- periodično preverjanje slepih povezav na spletni strani,
- kodiranje videov in večjih datotek v ozadju.

Iz navedenega lahko vidimo, da nam časovni razporejevalniki pomagajo pri marsikaterem rutinskem opravilu, ki bi bilo z izbiro drugačnega pristopa razvoja programske opreme zahtevnejše za implementacijo. Kot že omenjeno lahko za pristop k rešitvi problema izbiramo iz širokega spektra ponudnikov, zato sta v naslednjih podpoglavjih predstavljena dva najpomembnejša, GCP Cloud Scheduler in AWS CloudWatch Events.

3.2.1 Google Cloud Platform – Cloud Scheduler

Podjetje Google ponuja veliko storitev, ki so strnjene v ogromno platformo, imenovano Google Cloud Platform. Če se za implementacijo časovnega razporejanja rutinskih opravil odločimo za GCP, je potrebno za dosego cilja uporabiti kombinacijo dveh storitev, to je Cloud Functions in Cloud Scheduler. Slednja sama po sebi uporablja vse, kar prva ponuja, zato nam za povezavo med njima ni treba skrbeti.

Cloud Scheduler je v celoti upravljan razporejevalnik, ki omogoča razporejanje paketnih opravil, opravil z velikimi podatki ter operacij z oblačno infrastrukturo. Glavna prednost izbire tega razporejevalnika je zanesljivost, saj njegovo distribuirano infrastrukturo upravlja Google. S tem zagotavlja vsaj-enkratno dostavo sporočil na cilj, hkrati pa poenostavlja koncept »crontabov«, saj omogoča specifikacijo želenega urnika preko t. i. »cron« izraza. Zraven tega ponuja tudi močno orodje za beleženje izvedbe in uspešnosti ter enostavno konfiguracijo pravilnika za ponovni poskus v primeru napak. V osnovi so lahko na posameznem računu brezplačno gostovane tri instance razporejevalnika, vsaka naslednja instanca pa stane 0,10 dolarjev na mesec. [5]

3.2.2 Amazon Web Services – CloudWatch Events

Podobno kot Google tudi Amazon lasti platformo, ki ponuja velik nabor storitev, to je Amazon Web Services. V njenem sklopu lahko najdemo storitev AWS Lambda, ki sama po sebi ne ponuja časovnega razporejanja. Za dosego tega je Lambdo potrebno kombinirati z eno od treh storitev, ki to omogočajo: CloudFormation, CloudWatch Events in EventBridge, ki je posodobljena verzija storitve CloudWatch Events. Medtem ko AWS Lambda ponuja le storitve za oddaljeno izvajanje funkcij, se preostale tri storitve osredotočajo na sistemski tok dogodkov v realnem času. Storitev CloudWatch ostaja v tako imenovanem stanju pripravljenosti in ob določenih sprožilcih, kot so na primer časovni sprožilci, reagira. Slednja omogoča specifikacijo pravil za zagon funkcij, na primer na n-minut, dni, ali pa uporabi kompleksnejše razporejanje, za katero se prav tako uporablja t. i. »cron« izraz. [3]

Storitev CloudFormation v kombinaciji z AWS SAM (Serverless Application Model) omogoča tudi večstopenjsko izvajanje rutin, pri čemer je pisanje SAM predlog za brezstrežniško izvajanje povsem neboleče. Zaradi odlične podpore integracije različnih storitev znotraj AWS platforme je pri implementaciji razporejevalnikov meja le nebo. [18]

3.2.3 Primerjava predlaganih razporejevalnikov

Oba prej omenjena ponudnika sta razvila vrhunski rešitvi za časovno razporejanje, a vendar ima vsaka svoje prednosti.

Medtem ko GCP Cloud Schedule ponuja 3 brezplačne instance razporejevalnika AWS CloudWatch Events zagotavlja »zastonjsko« različico z maksimalno 100 zagoni na mesec. Opazna razlika je tudi v ceni - medtem, ko slednji za 10.000 zagonov ponuja ceno 0.30 dolarjev na mesec, GCP zagotavlja fiksno ceno 0.10 dolarjev na vsako dodatno instanco razporejevalnika. Dokumentacija Googlove storitve je napisana nekoliko bolj površinsko, Amazonova pa je podprta z ogromno primeri uporabe. Pri kompleksnosti v ospredje stopi Google, saj je povezovanje te storitve z drugimi iz nabora GCP trivialno, AWSove storitve pa so nekoliko zahtevnejše za integracijo. Hitrost izvedbe

je pri obeh ponudnikih izjemno visoka, zato med njima v tem aspektu praktično ni razlike. Če na kratko povzamemo primerjavo je v primeru potrebe po več različnih časovnih razporejevalnikih z majhnimi frekvencami zagona ter za eksperimentalne namene bolje izbrati AWS, v primeru fiksnega (manjšega) števila razporejevalnikov in točno zadanih ciljev kaj z razporejevalniki želimo doseči pa GCP.

4 Primer uporabe

Za bolj plastično predstavo delovanja časovnih razporejevalnikov v funkcijskem okolju uporabimo primer predstavljenega računovodskega servisa na sliki 2 (na naslednji strani), ki skrbi za pošiljanje plačilnih listov vseh zaposlenih v podjetjih s katerimi sodeluje. Servis želi popolno avtomatizacijo pošiljanja plačilnih listov preko e-pošte. Ker e-poštna sporočila vsebujejo kritične podatke za zaposlene je potrebno zagotoviti, da so ta sporočila stoodstotno poslana. To lahko dosežemo s kombinacijo razporejevalnikov, ki omogočajo časovno tempiranje sporočil in sporočilnih sistemov, ki ponujajo zagotovljeno pošiljanje sporočil. BPMN diagram iz slike 2 je zasnovan splošno, zato ga je mogoče implementirati enotno, ne glede na izbiro ponudnika funkcijske arhitekture.

Z zahtevo po zanesljivosti naraste tudi kompleksnost zasnovane arhitekture. Kot že omenjeno, lahko pride med izvajanjem zgoraj opisanega procesa do motenj v delovanju. Z zankami je tako predstavljeno lovljenje napak znotraj objave sporočil na temo. V primeru, da instanca med izvajanjem kode na prvi ali drugi temi pade, se po ponovnem zagonu stanje zaradi shranjevanja vmesnih korakov vedno ohrani. S tem je doseženo, da se vsa sporočila zagotovo pošljejo, prav tako pa, da se vsa sporočila pošljejo natanko enkrat. V ta primer uporabe je mogoče vključiti tudi druge prej omenjene možnosti. Ker plačilni listi veljajo za občutljive podatke, bi računovodski servis lahko razporejevalnike uporabil za kreacijo njihovih varnostnih kopij. Podobno bi se lahko izvedla deaktivacija zaposlenih, ki v sistemu več niso aktivni, ali pa, v primeru dodatnih funkcionalnosti v sistemu, omogočilo kodiranje večjih datotek v ozadju.

5 Diskusija

Kot lahko razberemo iz drugega poglavja, je pristopov k implementaciji časovnih razporejevalnikov rutin v oblačnih arhitekturah veliko, zato pri izbiri le-teh prevzamemo tudi nekaj posebnosti, ki jih je potrebno upoštevati pri implementaciji, izvedbi in vzdrževanju takšnih sistemov. Prav to nas je zanimalo v sklopu prvega raziskovalnega vprašanja, ki se glasi:

- **Katere so posebnosti časovnih razporejevalnikov v brezstrežniškem okolju?**

Posebnosti časovnih razporejevalnikov v brezstrežniškem okolju se delijo na tiste, ki za uporabnika predstavljajo prednosti ter tiste, ki zanj predstavljajo slabosti, kar je razvidno iz tabele 1. Med posebnosti, ki jih je potrebno upoštevati pri implementaciji, spadata nekoliko slabša podpora personalizacije ter pomanjkanje standardizacije. Iz tega razloga se je potrebno pri izbiri ponudnika poučiti o specifikah razporejevalnika, ki ga ponujajo. Razlike, ki jih zasledimo med ponudniki so predvsem finančne ter performančne. Kljub temu izbira brezstrežniškega okolja s funkcijskim pristopom, predstavljena v poglavju 3.2, prinaša veliko prednosti, med drugim enostavne modifikacije in nastavitev razporejevalnikov preko »cron« izraza ter zagotovljena vsaj enkratna izvedba funkcije, ki je poklicana v sklopu razporejevalnika, kar je pri klasičnih sistemih nekoliko težje doseči.

Med posebnosti lahko uvrstimo tudi to, da so razporejevalniki v brezstrežniškem okolju samostojna komponenta. To pomeni, da so ločeni od preostalega dela kode, zaradi česar se lahko v primeru težav z osnovno aplikacijo na njih še vedno zanesemo. Posebnost je tudi ta, da je ozke časovne omejitve težko doseči, najnižji razpon dveh izvedb je namreč zaradi možnosti časovnega prekrivanja pri izvedbi dveh klicev ena minuta. V sklopu prednosti, ki jih pridobimo z izbiro brezstrežniške arhitekture pa je smiselno odgovoriti tudi na drugo raziskovalno vprašanje, ki je bilo rdeča nit celotne vsebine članka:

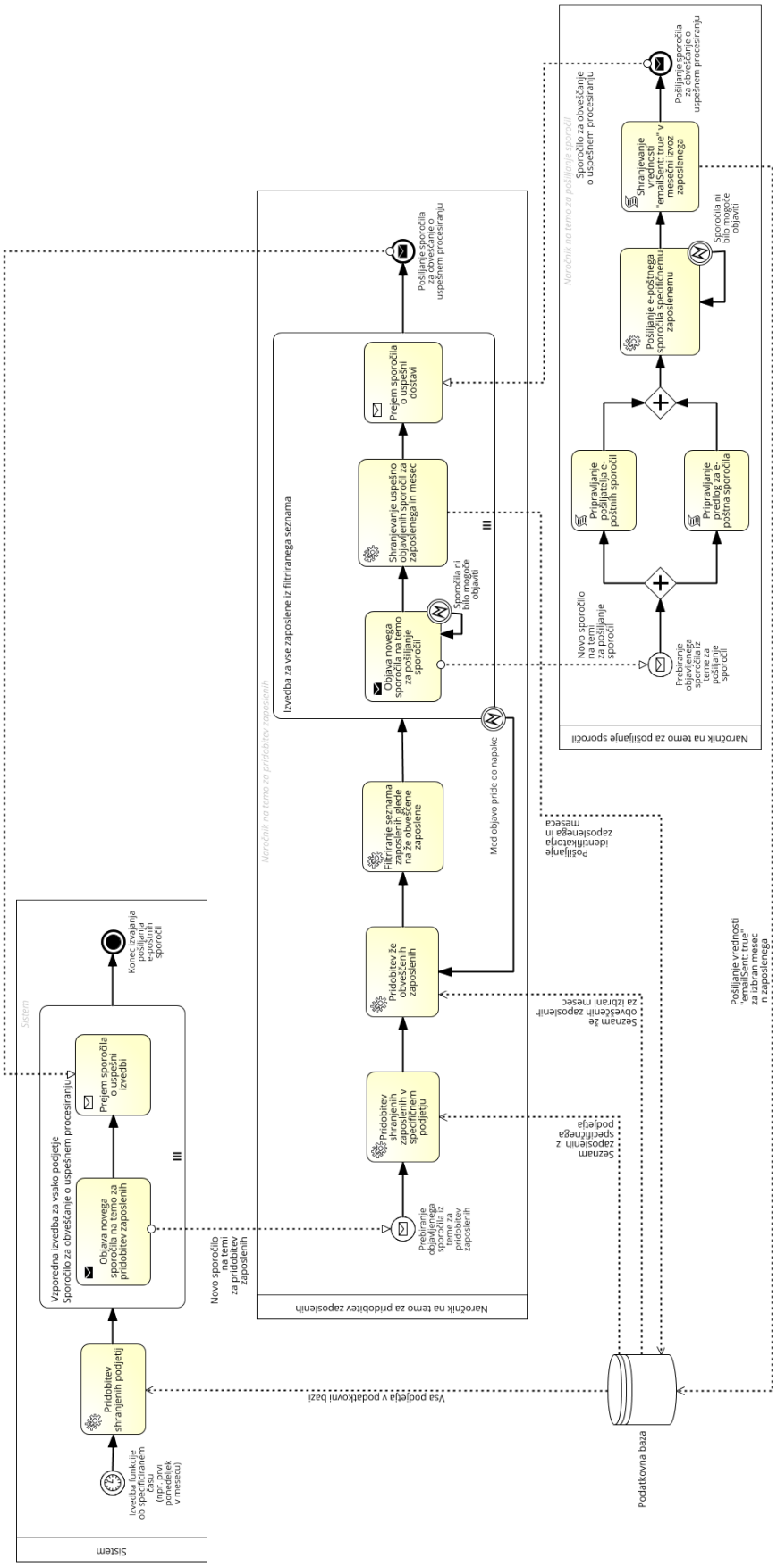
- **Kdaj je smiselno uporabiti časovne razporejevalnike brezstrežniškega okolja?**

Na prvi pogled razporejanje rutin ni nekaj, s čimer bi se srečevali pogosto. A realnost je precej drugačna, hitro lahko ugotovimo, da se časovne razporejevalnike uporablja za marsikatero opravilo, kar je razvidno iz nabora najpogostejših primerov uporabe iz poglavja 3.2. Smiselno jih je torej uporabiti predvsem za opravila, za katera želimo, da se izvedejo v ozadju in brez potrebe po poseganju upravljalca. Kot že omenjeno v poglavju 4 so to na primer vzdrževalne naloge re-indeksiranja podatkovne baze, deaktivacija poteklih uporabniških računov na nekem portalu, nadzor nad prostorom pomnilnika in upravljanje z njim, kreacija varnostnih kopij občutljivih podatkov, razpošiljanje e-poštnih sporočil določenim uporabnikom ter kodiranje večjih datotek v ozadju. Časovne razporejevalnike brezstrežniškega okolja z uporabo funkcij kot storitev je torej smiselno uporabiti vedno, ko je smiselno uporabiti vse druge časovne razporejevalnike, zraven tega pa imajo razporejevalniki tega tipa dodano vrednost hitre implementacije in malega vložka s strani uporabnikov, da storitev preverjeno deluje.

Izbira brezstrežniškega okolja pa ima še mnogo prednosti. Ena od teh je, da se razporejevalniki zaženejo le takrat, ko je nujno potrebno. Ravno nasprotno od tega lokalni razporejevalniki za nemoteno delovanje zahtevajo napravo, prižgano 24 ur na dan. Ločevanje komponent, v našem primeru razporejevalnikov, v našo kodo prinese svežino in olajša upravljanje s posameznimi deli projekta, saj so manjše enote veliko bolj obvladljive. Glavna prednost razporejevalnikov v brezstrežniškem okolju pa je vsekakor odgovor na vprašanje: zakaj bi ponovno implementirali nekaj, kar je že na voljo ter deluje zanesljivo in hitro.

6 Zaključek

V prispevku smo raziskovali pristope implementacije časovnih razporejevalnikov v brezstrežniškem okolju ter njihovo zanesljivo uporabo. Poglobili smo se v primerjavo



Slika 2: BPMN diagram zasnove razporejevalnika računovodskega sistema

dveh predstavnikov brezstrežniškega okolja, zabojnikov in pristopa FaaS. Podrobneje smo spoznali glavna ponudnika funkcij kot storitev, AWS CloudWatch Events in GCP Cloud Scheduler, ki sama po sebi omogočata implementacijo in zanesljivo uporabo časovnih razporejevalnikov.

Z analizo literature in odgovorom na raziskovalni vprašanji, zastavljeni na začetku raziskave smo ugotovili, da je časovne razporejevalnike predvsem v okolju FaaS izjemno enostavno razviti, prav tako pa ne zahtevajo veliko vzdrževanja. Smiselno jih je uporabiti za rutinska opravila, s katerimi se nimamo časa ukvarjati, zato nas lahko razbremenijo, hkrati pa so cenovno izjemno dostopni. Če vse skupaj povzamemo je časovne razporejevalnike brezstrežniškega okolja torej priporočljivo uporabiti vedno, ko se v sklopu projekta pojavi potreba po reševanju ponavljajočih oziroma rutinskih opravil.

Literatura

[1] Alqaryouti, O., and Siyam, N. Serverless Computing and Scheduling Tasks on Cloud: A Review. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)* (mar 2018), 1–14.

[2] Aske, A. M. *SCHEDULING FUNCTIONS-AS-A-SERVICE AT THE EDGE*. PhD thesis, WASHINGTON STATE UNIVERSITY, Vancouver, may 2018.

[3] AWS. What Is Amazon CloudWatch Events? - Amazon CloudWatch Events, 2021.

[4] Costello, K. The CIO's Guide to Serverless Computing. *Gartner* (jun 2020).

[5] Google. Cloud Scheduler | Google Cloud, 2021.

[6] Iams, T. Gartner Blog Network. *Gartner* (may 2019).

[7] Kohgadai, A. 6 Container Adoption Trends of 2020 | StackRox. *StackRox* (mar 2020).

[8] Kubernetes. What is Kubernetes? | Kubernetes, feb 2021.

[9] Leger, Y., and Broshar, A. FaaS vs CaaS: Comparing Use Cases and Responsibilities, feb 2021.

[10] Marathe, N., Gandhi, A., and Shah, J. M. Docker swarm and kubernetes in cloud computing environment. In *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019* (apr 2019), vol. 2019-April, Institute of Electrical and Electronics Engineers Inc., pp. 179–184.

[11] Moore, S. Gartner Forecasts Strong Revenue Growth for Global Container Management Software and Services Through 2024. *Gartner* (jun 2020).

[12] Panetta, K. Gartner Top Strategic Technology Trends for 2021. *Gartner* (oct 2020).

[13] Pereira Ferreira, A., and Sinnott, R. A performance evaluation of containers running on managed kubernetes services. In *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom* (dec 2019), vol. 2019-December, IEEE Computer Society, pp. 199–208.

[14] Rodriguez, M. A., and Buyya, R. Container-based cluster orchestration systems: A taxonomy and future directions. *Software - Practice and Experience 49*, 5 (may 2019), 698–719.

[15] Singhvi, A., Houck, K., Balasubramanian, A., Danish Shaikh, M., Venkataraman, S., and Akella, A. Archipelago: A Scalable Low-Latency Serverless Platform. Tech. rep., University of Wisconsin-Madison, Wisconsin, nov 2019.

[16] Tozzi, C. Why Is Docker So Popular? Explaining the Rise of Containers and Docker – Channel Futures, jul 2017.

[17] Walker, J. How to Use Cron With Your Docker Containers, jan 2021.

[18] Yilmaz, E. 3 Ways to Schedule AWS Lambda and Step Functions State Machine Executions , jan 2020.

[19] Zhang, R., Chen, Y., Zhang, F., Tian, F., and Dong, B. Be Good Neighbors: A Novel Application Isolation Metric Used to Optimize the Initial Container Placement in CaaS. *IEEE Access 8* (sep 2020), 178195–178207.