Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/13BE29Kovacev

RAZVOJ APLIKACIJE ZA AŽURIRANJE MODELA ELEKTROENERGETSKE MREŽE U OKVIRU CLOUD OKRUŽENJA

DEVELOPMENT OF AN APPLICATION FOR UPDATING THE POWER NETWORK MODEL WITHIN THE CLOUD ENVIRONMENT

Bogdan Kovačev, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad bavi se kreiranjem aplikacije za ažuriranje modela elektroenergetske mreže u Cloud okruženju korišćenjem Azure Service Fabric platformu. U radu su takođe izvršena i testiranja vremena akcije dodavanja novih entiteta u model mreže za različiti broj klijenata kako monolitne aplikacije, tako i aplikacije implementirane u Cloud okruženju.*

Ključne reči: *NMS, Cloud, Azure, mikroservisi, elektroenergetski sistem*

Abstract – *This paper is deticated to the creation of applications for updating power grid models in the Cloud environment using the Azure Service Fabric platform. The paper also tests the time of action of adding new entities to the network model for a different number of clients, both monolithic applications and applications implemented in the Cloud environment.*

Keywords: *NMS, Cloud, Azure, microservices, electroenergy system*

1. UVOD

1.1 Azure Service Fabric

Azure Service Fabric [1] je platforma kao usluga (PaaS) koja je dizajnirana da olakša razvoj, primenu i upravljanje visoko skalabilnim i prilagodljivim aplikacijama za Microsoft Azure platformu na Cloud-u. U osnovi to je arhitektura koja se zasniva na mikroservisnoj arhitekturi u kojoj programeri pretvaraju tradicionalne monolitne aplikacije u skup diskretnih servisa.

Uvođenjem Cloud-a, skaliranje i pouzdanost su napravili potpuno novu revoluciju u implementaciji na različite načine u celom sistemu.

Service Fabric omogućava izgradnju i upravljanje skalabilnim i pouzdanim aplikacijama koje su sastavljene od mikroservisa koji su pokrenuti na više mašina, odnosno na klasteru. Koristeći Service Fabric programeri i administratori mogu izbeći rešavanje složenih infrastrukturnih problema i umesto toga se fokusirati na implementaciju kritičnih, zahtevnih zadataka znajući da su oni skalabilni, pouzdani i da je njima moguće upravljati.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Vukmirović, vanr.prof.

1.2 Elektroenergetski sistem

Elektroenergetski sistem (EES) je tehnički sistem čiji je osnovni zadatak da osigura kvalitetnu isporuku električne energije uz minimalne troškove. Tehnološki proces u EES sastoji se iz sledećih faza: proizvodnja, prenos, distribucija i potrošnja. Za svaki EES postoji glavni centar upravljanja odakle se upravlja proizvodnjom električne energije. Moderni i veliki EES obuhvataju velika područja jedne ili više država pa se upravljanje EES vrši iz više centara. Upravo iz tog razloga se EES deli na mnogo velikih delova koje je potrebno nadzirati i pratiti stanja u kojima se ti delovi mreže nalaze.

2. OPIS KORIŠĆENIH TEHNOLOGIJA I ALATA

Azure Service Fabric je platforma distribuiranih sistema koja olakšava implementaciju i upravljanje skalabilnim i pouzdanim mikroservisima.
C# je jezik iz familije C jezika, koji je u potpunosti baziran na principima OOP(objektno orijentisanog programiranja).
.NET Framework predstavlja programsko okruženje koje služi za lakši razvoj programa.
WPF (*Windows Presentation Foundation*) predstavlja tehnologiju za pravljenje klijentskih aplikacija na *Windows* platformi, koja nudi mnogobrojne opcije za podešavanje izgleda aplikacije.
Microsoft Visual Studio je softversko razvojno okruženje koje omogućava razvoj aplikacija za sve *Microsoft*-ove platforme i koristi .NET Framework pa je zato pogodan za ovaj projekat.

3. OPIS PROBLEMA

Tematika problema jeste da se omogući rad aplikacije za ažuriranje modela elektroenergetske mreže u Cloud okruženju korišćenjem Azure Service Fabric platformu. Aplikacija ima mogućnost prikaza informacija željenog objekta učitanog iz CIM profila kao i omogućavanje korisnicima izmenu, dodavanje i brisanje željenih entiteta. Potrebno je korisniku omogućiti prikaz informacija u realnom vremenu, onemogućiti mu rad sa nevalidnim entitetima, brzo ažuriranje informacija, kao i obaveštenja ukoliko je došlo do promena nekog entiteta sa kojim korisnik trenutno radi. Mikroservisna arhitektura predstavlja evoluciju servisno orijentisane arhitekture (SOA) i sačinjena je od skupa nezavisnih servisa od kojih svaki implementira jedan poslovni zahtev. Glavni cilj jeste podeliti aplikaciju na više manjih nezavisnih mikroservisa i omogućiti međusobnu komunikaciju

mikroservisa kako bi se omogućio neometani rad aplikacije u Cloud okruženju. Neki od benefita korišćenja miroservisne arhitekture u odnosu na tradicionalnu monolitnu arhitekturu su: nezavisno objavljivanje servisa na produkcione servere bez posledica po ostatak sistema, veća skalabilnost, nezavisan razvoj servisa, itd.

4. TEORIJSKE OSNOVE

4.1 Mikroservisna arhitektura

Mikroservisna arhitektura [2] poslednjih godina postaje sve popularnija, a naročito u velikim kompanijama koje, zbog njenih mogućnosti, sve više prelaze sa monolitnih aplikacija na mikroservisnu. Među prvima su prešli Amazon i Netflix, kao i kompanije sa ogromnom bazom korisnika kao što je SoundCloud. Manje kompanije i organizacije teže prelaze na mikroservisnu arhitekturu jer još uvek među ljudima vlada mišljenje da je lakše održavati jednu aplikaciju nego više njih i da je to dobro samo za velike kompanije. Međutim, ključna stvar jeste prepoznati pravi trenutak u kojem bi trebalo preći sa monolitne na mikroservisnu arhitekturu, kada monolitna aplikacija nije više optimalna. Problemi koji se najčešće javljaju u monolitnim aplikacijama su: uska grla, održavanje, spor razvojni ciklus, spora isporuka, kao i zastareli softver.

Veliki problem u monolitnim aplikacijama sa velikom obradom podataka predstavlja i održavanje. Ako se obrada tih podataka smesti u jednu veliku grupu pozadinskih procesa, obično je potreban celi tim programera kako bi održao samo taj sistem. Zbog svega ovoga, timovi obično ulože više vremena na rešavanje tehničkih problema nego na pravljenje nekih novih i zanimljivijih stvari. Ovo automatski vodi do novog problema, a to je spora isporuka, što se dešava aplikacijama koje su vremenom nakupile mnogo asset-a i zavisnosti. Ovakve monolitne aplikacije ne odgovaraju ni novopečenim programerima, a ni onima koji su tu od nastanka aplikacije. Novim programerima je potrebno mnogo više vremena da razumeju kod i da budu sposobni da unesu neku malu promenu u kodu.

4.2 Korišćenje Cloud-a

Cloud [2] zamenjuje skupu IT infrastrukturu, omogućava veću fleksibilnost u radu, jednostavnost i sigurnost za poslovne procese uz značajnu uštedu. Garantuje potpunu privatnost i bezbednost podataka uz najsavremenije sisteme zaštite. Ima automatski backup koji omogućava pouzdan oporavak sistema u slučaju nekih poremećaja i kvarova. Podacima se može pristupati sa bilo kog uređaja, bitno je samo da je taj uređaj povezan na Internet. Među najpopularnijim Cloud platformama danas su Microsoft Azure Service Fabric i Amazon Web Services.

5. IMPLEMENTACIJA REŠENJA

5.1 Arhitektura rešenja

Arhitektura rešenja se sastoji od tri Service Fabric aplikacije pokrenute u Cloud okruženju i dve pokrenute na lokalu. Service Fabric aplikacije: NMS, Transaction Coordinator i Calculate Engine. Na lokalu se nalazi Importer koji učita model iz XML dokumenta i prosleđuje ga na dalju obradu ka GDA Microservice-u. Druga aplikacija na lokalu jeste korisnička UI aplikacija koja pruža korisniku prikaz informacija entiteta različitih tipova, pretragu entiteta po svim vrednostima atributa koje sadrže entiteti tog tipa, dodavanje i brisanje entiteta, pa i izmenu vrednosti atributa željenih entiteta. Na slici 1 se nalazi prikaz arhitekture realizovanog rešenja sa svim Service Fabric aplikacijama i njihovim mikroservisima. Prikazani su komunikacioni kanali izmedju mikroservisa koji međusobno komuniciraju, kao i tipovi mirkoservisa (Stateless ili Stateful).

Transaction Coordinator Microservice je zadužen za kontrolu distribuiranih transakcija između servisa (NMS i Calculation Engine). Potrebno je da se oba servisa jave Transaction Coordinator-u i tek tada se smatra da je transakcija uspešno završena u suprotnom se vrši rollback. Prilikom podizanja servisa, potrebno je da se prijave Transaction Coordinator-u pozivom Enlist metode. Back up (kopija čitavog modela) se čuva po principu Deep Copy.

NMS Transaction Microservice Nakon uspešno izvršene akcije učitavanja modela i kreiranje Delte, kreira se kopija modela po principu Deep Copy. Ako se uspešno podignu svi servisi i omogući se transakcija modela izmedju NMS-a i CE-a poziva se metoda Commit gde se vrši akcija transakcija modela između pomenutih servisa.

NMS Command Microservice je servis zadužen za primanje komandi koje je izdao korisnik korišćenjem UI aplikacije. Komande koje može da primi su vezane za model mreže gde korisnik ima mogućnosti dodavanja novog entiteta, brisanje postojećeg entiteta, kao i izmenu vrednosti atributa negok već postojećeg entiteta.

NMS GDA Microservice je zadužen za kreiranje učitavanje modela, kreiranje Delte, ažuriranje modela, kao i kreiranje kopije modela (Deep Copy). Kad se učita model, kreira Delta i ustanovi da su svi servisi uspešno pokrenuti i spremni za transakciju, učitani model se dalje prosledjuje Calculate Engine-u na obradu. Kada korisnik izda komandu za ažuriranje modela (insert, update ili delete entiteta) i komanda sa svim potrebnim parametrima stigne do GDA Microservice-a vrši se pozivanje metode ApplyDelta. Ova metoda vrši obradu pristiglih podataka i izvršava željenu akciju korisnika, ukoliko je moguće izvršiti.

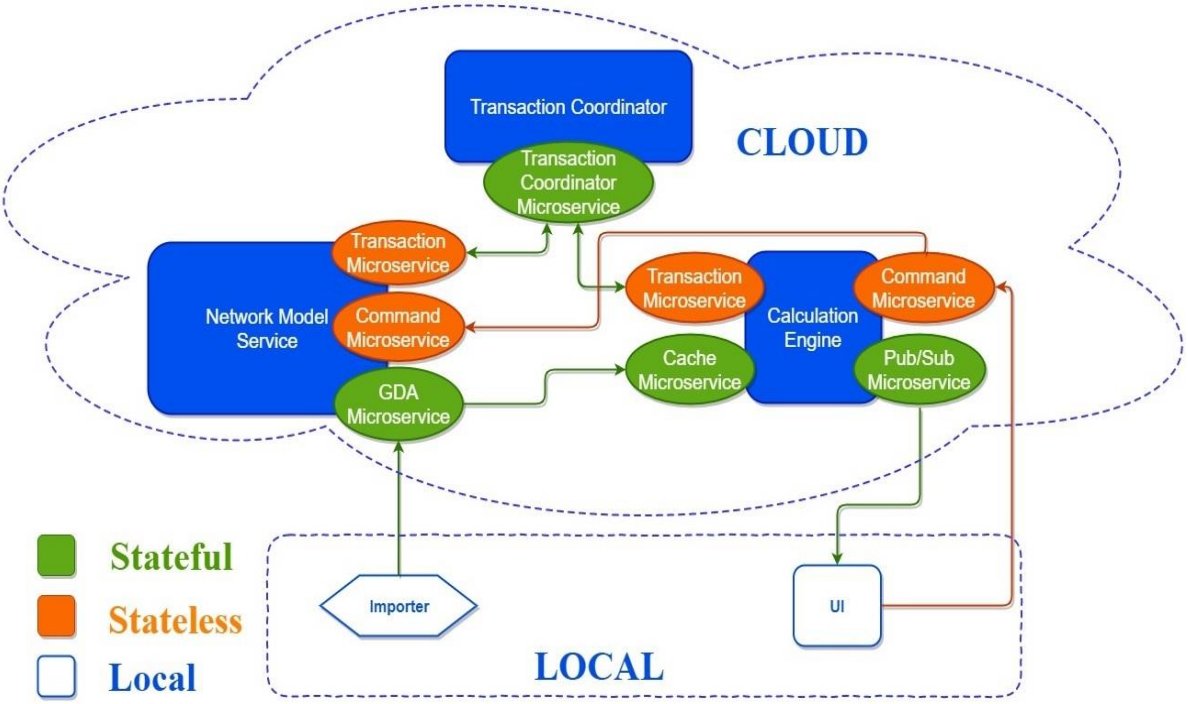
CE Transaction Microservice nakon uspešno izvršene akcije učitavanja modela i kreiranje Delte i nakon potvrde o uspešnosti podizanja servisa, omogućena je transakcija modela izmedju NMS-a i CE-a. Model se prosledjuje do Cache Microservice-a na dalju obradu.

CE Command Microservice je servis zadužen za primanje komandi koje je izdao korisnik korišćenjem UI aplikacije. Jedan deo komandi je vezan za metode koje pruža GDA Microservice, dok je drugi deo komandi vezan za Cache Microservice. Komande vezane za GDA Microservice prosledjuju se ka NMS Command Microservice na dalju obradu, dok se drugi deo komandi prosledjuje ka Cache Microservice-u.

Cache Microservice je kreiran kao Stateful mikroservis zbog toga što se na njemu nalazi model mreže prosleđen sa NMS-a. Kada model stigne sa NMS-a, Cache Microservise kreira IReliableDictionary u kom se čuva pristigli model mreže. Nakon podizanja aplikacije i pretplaćivanje korisnika na Pub/Sub mehanizam, Cache Microservice iz modela izvlači GID-ove entiteta koji se

prosleđuju ka UI aplikaciji kako bi se korisniku omogućio prikaz GID-ova entiteta.

Pub/Sub Microservice pruža mogućnosti Publisher-Subscriber mehanizma. Pub/Sub servis je servis baziran na Publisher/Subscriber obrascu. Obrazac je dizajniran po principu pretplate na određeni događaj. Klijent se pretplaćuje na određene događaje, a servis je dužan da klijenta obavesti o događaju. Komunikacija sa klijentima se obavlja putem poruka koje klijent kasnije obradi. Pretplaćeni klijenti se moraju čuvati na servisu da bi servis znao koga je potrebno obavestiti o događajima. Komunikacija između klijenta i servisa je asinhrona što znači da klijent nije blokiran dok čeka na poruke. Informacije o klijentima Pub/Sub servis čuva u IReliableDictionary da bi se i prilikom gašenja servisa informacije o klijentu sačuvale.



Slika 1. *Arhitektura rešenja*

5.3 Azure Service Fabric node

Service Fabric Cluster [3] je mrežno povezan skup virtuelnih ili fizičkih mašina na kojima se mikroservisi pokreću i vrši se njihovo upravljanje. Mašina ili VM koji je deo klastera naziva se čvor klastera. Klasteri se mogu skalirati na hiljade čvorova. Ako se klasteru dodele novi čvorovi, Service Fabric rebalansira replike servisne particije i instance na povećanom broju čvorova.Tip replike određuje njenu ulogu u skupu replika:

Primary (P): U skupu replika postoji jedna primarna koja je odgovorna za izvođenje operacija čitanja i pisanja.

ActiveSecondary (S): Ovo su replike koje primaju ažurirana stanja od primarne, primenjuju ih i zatim vraćaju potvrde.

IdleSecondary (I): Ove replike prave primarne replike. Oni primaju stanje iz primarne pre nego što mogu da se unaprede u aktivnu sekundarnu

None (N): Ove replike nemaju odgovornost u kompletu replika.

5.4 Testiranje

Testirana je aplikacija za ažuriranje modela elektroenergetske mreže pre i posle njene implementacije u Cloud okruženju. Aplikacija ima mogućnost dodavanja, brisanja i izmenu entiteta modela mreže. Dodavanje novog entiteta u model mreže predstavlja najkompleksniju akciju u odnosu na prethodno navedene akcije i zbog toga se u ovom testu vrši merenje brzine dodavanja novih entiteta u model mreže od strane različitog broja klijenata istovremeno.

Tabela 1. *Rezultati testiranja*

Br. klijenata	Monolitna aplikacija [s]	Aplikacija u *Cloud* okruženju[s]
10	1	1
50	2	1
100	4	3
250	8	6
500	17	11
1000	31	19

U tabeli 1 prikazana su vremena izvršavanja akcije dodavanja novih entiteta u zavisnosti od broja klijenata koji istovremeno zahtevaju kreiranje entiteta. U prvoj koloni prikazan je broj klijenata koji zahtevaju kreiranje novih entiteta, u drugoj koloni se nalaze vremena izvršavanja akcije vezane za monolitnu aplikaciju pre implementacije u Cloud okruženju, dok se u trećoj koloni nalaze vremena vezana za aplikaciju implementiranu u Cloud okruženju.

Vrednosti u tabeli predstavljane su u sekundama i zaokružene su na celobrojne vrednosti. Iz tabele se jasno može zaključiti da su vremena izvršavanja akcije dodavanja novog entiteta za mali broj korisnika približna, dok se sa porastom broja klijenata vrednosti međusobno

razlikuju. Sa povećanjem broja korisnika, monolitnoj aplikaciji treba više vremena za izvršenje akcije u odnosu na aplikaciju implementiranu u Cloud okruženju i na osnovu toga se može zaključiti da je rad monolitne aplikacije sporiji u odnosu na aplikaciju koja je implementirana u Cloud okruženju.

6. ZAKLJUČAK

Cilj multiklijentskog korisničkog rešenja za ažuriranje modela elektroenergetske mreže jeste ostvarivanje mogućnosti praćenja trenutnih stanja modela mreže sa svim njenim elementima i kontrolisanje toka podataka između elemenata sistema. Aplikacija pruža mogućnosti jednostavne izmene modela, dodavanja i brisanja entiteta, te i prikaz svih informacija o entitetima iz modela.

Azure Service Fabric je platforma kao usluga koja je dizajnirana da olakša razvoj, primenu i upravljanje visoko skalabilnim i prilagodljivim aplikacijama za Microsoft Azure platformu na Cloud-u koja predstavlja arhitekturu koja se zasniva na mirkoservisnoj arhitekturi. Teži se ka tome da se tradicionalne monolitne aplikacije transformišu u skup diskretnih mikroservisa. U ovom radu je opisana jedna takva transformacija gde je izvršena transformacija monolitne aplikacije u aplikaciju sačinjenu od više diskretnih mikroservisa pokrenutih u Cloud okruženju.

7. LITERATURA

[1] Haishi Bai, Programming Microsoft Azure Service Fabric, 2016..

[2] M.Simonović, Tehnologija Cloud Computing-a, Univerzitet Singidunum, 2013.

[3] Haishi Bai, Programming Microsoft Azure Service Fabric, 2nd Edition, 2018.

Kratka biografija:

Bogdan Kovačev rođen je 20.04.1996. godine u Novom Sadu. Diplomirao je na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjeno softversko inženjerstvo 2019. godine.