ZeroMQ

Seminarski rad

Radovan Župunski

Novi Sad, 2021

Sadržaj

1. Uvod

Arhitektura modernih aplikacija podrazumeva rastavljanje aplikacije na manje celine koje međusobno komuniciraju. Ovakav pristup se pokazao kao izuzetno pogodan za razvoj aplikacije, nadogradnju i održavanje. Međutim, problem sa kojim je nastao usled razdvojenosti delova aplikacije je potreba da ti delovi komuniciraju, kao i potreba koordinacije delova aplikacije međusobno.

Ideja reda poruka je omogućila komunikaciju između aplikacija ili komunikaciju delova jedne iste aplikacije. Sam pojam reda podrazumeva strukturu za prihvatanje poruka koja se karakteriše FIFO organizacijom. Ovo znači da prva poruka koja uđe u red poruka prva i izlazi iz reda. Poruka predstavlja zapakovane podatke spremne za razmenu između pošiljaoca i primaoca poruka. Poruka može da sadrži neke konkretne informacije ili može da inicira pokretanje nekog procesa ili može da sadrži informacije o završenom procesu. Red poruka zapravo predstavlja jednostavnu strukturu. Podrazumeva dve krajnje komponente u komunikaciji. Jedna od njih se naziva proizvođač poruka i ova komponenta proizvodi poruke i šalje ih ka redu poruka. Druga krajnja komponenta se naziva potrošač ili konzument poruka. Ova komponenta se konektuje na red poruka i čita poruke iz reda.

2. ZeroMQ

Ovo poglavlje je namenjeno ZeroMQ biblioteci i u njemu će biti opisani osnovni koncepti, vrste poruka i soketa kao i različiti šabloni razmene poruka.

2.1. Šta je ZeroMQ

ZeroMQ je biblioteka visokih performansi za razmenu poruka koja se koristi u distribuiranim sistemima i konkurentnim aplikacijama. “Zero” tj. nula u nazivu ove biblioteke između ostalog označava i da ZeroMQ iako koristi red poruka ne zahteva postojanje brokera u razmeni poruka. Broker kao komponenta u sistemu razmena poruka predstavlja zaseban modul čija uloga je da posreduje u razmeni poruka između pošiljaoca i primaoca poruka. Kako ZeroMQ nema brokera a samim tim ni dodatnu administraciju oko brokerske komponente nula u nazivu biblioteke dobija još jedno značenje(nula administracije).

ZeroMQ podržava uobičajene šablone razmene poruka preko raznih vrsta transporta. Iako izvorno napisana u C++ programskom jeziku i nazvana libzmq, ZeroMQ biblioteka je podržana u velikom broju drugih programskih jezika.

2.2. ZeroMQ poruke

ZeroMQ poruke su izolovane jedinice podataka koje se razmenjuju izmedju aplikacija ili komponenti jedne iste aplikacije. Same poruke predstavljaju, iz ugla ZeroMQ-a, nečitljive binarne podatke. Zapravo, na samoj mreži ZeroMQ poruke možemo posmatrati kao velike binarne objekte, BLOB (engl. *Binary large object),* proizvoljne veličine.

Najjednostavnija ZeroMQ poruka sastoji se od jednog okvira ili frejma (engl. *frame*). Frejm je moguće posmatrati kao deo poruke tj. kao blok koji u sebi nosi određenu količinu podataka. Na taj način

moguće je slati poruke u vidu liste frejmova i takođe primati poruke kao listu frejmova. ZeroMQ ima mehanizme pomoću kojih može da garantuje da će ili svi frejmovi jedne poruke koji su poslati biti i dostavljeni ili neće biti dostavljen ni jedan frejm. Ovo nam pruža garanciju da primalac neće dobiti nekompletnu poruku u slučaju da jedan od frejmova iz liste frejmova koji sačinjavaju poruku iz nekog razloga ne stigne do primaoca. Ukoliko se javi potreba da za slanjem velikih poruka koje ne mogu da stanu u memoriju, preporuka je da se poruka rastavi na delove i da se onda svaki deo šalje zasebno. Ovo ne smanjuje potrošnju memorije.

2.3. ZeroMQ soketi

Osnovni način mrežne komunikacije čine mrežni soketi. ZeroMQ koristi više različitih soketa i na taj način olakšava implementaciju raznih šablona razmene poruka ali takođe, kroz svoje sokete, ZeroMQ vrši apstrakciju nad osnovnim mrežnim komunikacionim protokolima, prikriva njihovu kompleksnost i na taj način sama upotreba soketa je znatno lakša.

Razlika između konvencionalnih soketa i ZeroMQ soketa je u tome što konvencionalni soketi predstavljaju sinhroni interfejs koji se zasniva na konekciji za prenos bajtova dok ZeroMQ predstavljaju apstrakciju asinhronog reda poruka pri čemu u zavisnosti tipa soketa zavisi i semantika reda poruka. Asinhronost reda poruka kod ZeroMQ soketa se ogleda u tome što su vreme uspostavljanje konekcije, prekidanje konekcije, ponovno konektovanje kao i sam prenos podataka transparentni korisniku i podešeni su od samog ZeroMQ-a. U slučaju da nije moguće dostaviti poruku nekom od primaoca, moguće je skladištiti poruke u red. Konvencionalni soketi omogućavaju samo jedan prema jedan, više prema jedan i jedan prema više razmenu poruka, dok je kod ZeroMQ-a moguće koristi soket koji se može konektovati na više krajnjih tački dok istovremeno

može da prihvata konekcije sa više drugih krajnjih tački što omogućava više prema više vezu.

Životni ciklus soketa ogleda se u 4 faze. Prva faza podrazumeva kreiranje soketa. Zatim u drugoj fazi podrazumeva konfigurisanje opcija na samom soketu. Treća faza se ogleda u kreiranju konekcije od i ka soketu i na taj način se soket uključuje u mrežu. Na samom kraju, četvrta i poslednja faza podrazumeva korišćenje soketa za slanje i primanje poruka.

Tipovi uspostavnja veze sa soketom mogu biti vezivanje (engl. *Bind*) i konektovanje (engl. *Connect*). Osnovna razlika između konektovanja i vezivanja je u tome što prilikom vezivanja na soket ne dolazi do kreiranja reda poruka sve dok se neko ne konektuje na taj soket dok konektovanje na soket automatski podrazumeva i kreiranje reda poruka jer se podrazumeva da postoji barem jedna krajnja tačka u komunikaciji. Ovo može da rezultuje time da ukoliko se nakon vezivanja na soket šalju poruke a da niko pri tom nije konektovan na soket može doći do gubljenja poruka jer ne postoji red poruka u koji bi se poruke sačuvale. Jedan od primera korišćenja vezivanja i konektovanja je na primer da se u razmeni poruka server-klijent, server koristi vezivanje a klijent konektovanje.

Za svaki red poruka na soketu postoji limit koliko poruka može da primi i sačuva. U slučaju da se dođe do limita, u zavisnosti od tipa soketa, dolazi do odbacivanja poruka ili blokiranja primanja poruka.

2.4. Šabloni razmene poruka

Šabloni razmene poruka implementirani su pomoću para odgovarajućih soketa. Ugrađeni šabloni razmene poruka u ZeroMQ biblioteci su: Request-reply, Pub-sub, Pipeline i Exclusive pair šablon.

2.4.1. Request-reply šablon

Request-reply šablon namenjen je servisno orjentisanim arhitekturama i dolazi u dva osnovna tipa a to je sinhroni (REQ i REP soketi) i asinhroni(DEALER i ROUTER soketi).

REQ soket

REQ soket koristi klijent u komunikaciji da pošalje zahtev servisu i da primi odgovor od servisa. Ovaj tip soketa dozvoljava samo naizmeničnu sekvencu slanja i primanja poruka. Svaki poslat zahtev kruži između svih povezanih servisa a svaki odgovor se poklapa sa poslednjim poslatim zahtevom. Ovaj tip soketa može biti konektovan na proizvoljan broj REP i ROUTER soketa i projektovan je za jednostavne request-reply modele. REQ soket ne odbacuje poruke, pa tako ako ni jedan servis nije dostupan operacija slanja se blokira dok se neki od servisa ne oslobodi.

REP soket

Ovaj tip soketa koristi servis da primi zahteve od klijenata i da pošalje odgovor nazad ka klijentu. REP soket dozvoljava samo naizmeničnu sekvencu primanja i slanja poruka. Svaki zahtev se smešta u red poruka a poslat odgovor se usmerava ka klijentu koji je poslao poslednji zahtev. Ukoliko je pošiljalac zahteva izgubljen usled prekida veze ili drugih problema, poruka sa odgovorom se odbacuje.

DEALER soket

DEALER soket je asinhrona zamena za REQ soket. Dakle, koristi ga klijent za slanje zahteva i primanje poruka. U slučaju dostizanja limita skladištenja poruka u redu poruka, ovaj tip soketa ulazi u stanje mirovanja i blokira sve operacije slanja i na taj način prevenira odbacivanje poruka.

ROUTER soket

ROUTER soket je asinhrona zamena za REP soket. Često se koristi kao osnova za server koji komunicira sa DEALER klijentima. Ovaj tip soketa koristi eksplicitno adresiranje pa se tako svaka odlazna poruka šalje ka određenoj klijentskoj konekciji. Prilikom prijema poruke, ROUTER soket će dodati na poruku deo u kojem će se nalaziti identifikator rute sa koje je stigla poruka. Nakon toga se poruka šalje u red poruka. Prilikom slanja poruke, prvi deo poruke se uklanja i identifikator rute se upotrebljava za usmeravanje poruke ka ciljnom klijentu. U slučaju da klijent više ne postoji, poruka se odbacuje. U slučaju da ROUTER soket dostigne limit poruka, ulazi u stanje mirovanja i odbacuje sve novopristigle poruke dok iz tog stanja ne izađe.

2.4.2. Pub-sub šablon

Ovaj šablon razmene poruka namenjen je jedan prema više tipu komunikacije gde jedan objavljivač šalje poruke ka više pretplatnika. Ovaj šablon komunikacije podržan je sa 4 tipa soketa.

Svaki objavljivač mora u prvom frejmu poruke da naznači koja tema poruke je u pitanju a pretplatnici specificiraju za koje teme su zainteresovani. Na ovaj način pretplatnici dobijaju poruke samo određene teme ali i svih pod tema u okviru neke teme. Takođe, pretplatnik dobija sve poruke koje prođu prefiksnu proveru naziva teme. Na primer, ukoliko je pretplatnik pretplaćen na temu “topic” on će dobijati poruke sa temom “topic” ali će isto dobijati i poruke sa temom “topical” jer se “topic” kao naziv teme nalazi u ulozi prefiksa u temi “topical”. Ovo znači da ako neki pretplatnik navede da je zainteresovan za temu čiji naziv reprezentuje prazan string, on će onda dobijati poruke koje pripadaju bilo kojoj temi.

PUB soket

PUB soket koriste objavljivači da pošalju podatke svim konektovanim pretplatnicima. Ova vrsta soketa ne može da prima poruke. U slučaju da PUB soket dođe do limita poruka za pretplatnike, sve poruke koje bi trebalo da budu postulate pretplatnicima bivaju odbačene dok se stanje mirovanja ne završi. Fukcija za slanje poruka nikada ne blokira kod ovog tipa soketa.

SUB soket

Ovu vrstu soketa koristi pretplatnik da bi mogao da primi poruke koje objavljuje objavljivač. Inicijalno, SUB soket nije pretplaćen ni na kakve poruke. Nije moguće slati poruke pomoću ovog soketa.

XPUB soket

XPUB soket je identičan kao i PUB soket s tim što XPUB soket može da primi poruku pretplate od pretplatnika. Pretplatna poruka je 1 bajt u slučaju pretplate ili 0 bajta u slučaju prekidanja pretplate nakon čega ide telo pretplatne poruke. U slučaju da ne postoji prefiks u pretplatnoj poruci, one takođe mogu biti primljene ali nemaju efekta na status pretplate.

XSUB soket

Identično kao i SUB soket, XSUB soket koriste pretplatnici za primanje poruka od objavljivača s tim što je pomoću XSUB soketa moguće i poslati pretplatnu poruku. Pretplatna poruka je 1 bajt u slučaju pretplate ili 0 bajta u slučaju prekidanja pretplate nakon čega ide telo pretplatne poruke. U slučaju da ne postoji prefiks u pretplatnoj poruci, one takođe mogu biti poslate ali nemaju efekta na status pretplate.

2.4.3. Pipeline šablon

Ovaj šablon je namenjen za raspoređivanje zadataka. Prilikom izvršavanja nekih procesa koji imaju više faza, nekoliko čvorova može da raspoređuje posao na druge čvorove a ovi zatim mogu rezultate da šalju ka jednom ili više čvorova koji su određeni za prikupljanje rezultata. Ovaj šablon ne odbacuje poruke sem u slučaju kada se prekine konekcija sa nekim čvorom neočekivano. Čvorovi se nazad mogu pridružiti u bilo kom trenutku.

Za ovaj šablon podržana su dva tipa soketa: PUSH i PULL soketi.

PUSH soket

PUSH soket komunicira sa grupom anonimnih PULL soketa tako što im šalje poruke *round-robin* algoritmom. Operacija primanja poruka nije implementirana na ovom soketu. Kada ovaj soket uđe u stanj mirovanja u slučaju da nema čvorova kojima može da prosledi poruke, on blokira slanje poruka dok ne izađe iz tog stanja mirovanja, dakle ovaj soket ne odbacuje poruke.

PULL soket

Ovaj sokete komunicira sa grupom anonimnih PUSH soketa primajući od njih poruke. Operacija slanja nije implementirana za ovaj soket.

2.4.4. Exclusive pair šablon

Ovaj šablon podrazumeva upotrebu PAIR soketa i uglavnom se koristi za komunikaciju programskih niti u okviru istog procesa.

PAIR soket

PAIR soket može da ostvari samo jednu konekciju istovremeno pa samim tim rutiranje poruka nije potrebno. Kada PAIR soket uđe u

stanje mirovanja zbog toga što je dostignut limit na konektovanom soketu ili ni jedan soket nije konektovan, poruke se ne odbacuju nego se operacija slanja poruka blokira.

3. Poređenje ZeroMQ sa Kafkom i RabbitMQ

U ovom poglavlju biće opisane alternativne tehnologije i biće upoređene sa ZeroMQ bibliotekom.

3.1. ZeroMQ i RabbitMQ

RabbitMQ je broker poruka otvorenog koda koji implementira AMQP (engl. *Advanced Message Queuing protocol*) protokol razmene poruka. Neke od osnovnih komponenti RabbitMQ brokera su proizvođač poruka, primalac poruka, red poruka i komponenta za razmenu poruka. Prizvođač poruka kreira poruke i šalje ih u red poruka ali tako da nikada ne koristi red poruka direktno nego posredstvom komponente za razmenu poruka. Komponenta za razmenu poruka prihvata poruke od proizvođača poruka i prosleđuje ih ka redu poruka. Red poruka predstvalja mesto skladištenja poruka dok primalac poruka predstvalja komponentu koja čita poruke iz reda poruka.

Perzistencija poruka. ZeroMQ ne podržava perzistenciju poruka, što znači da ako je na primer primalac diskonektovan sa mreže sve poruke će biti izgubljene jer nema načina da se te poruke sačuvaju i skladište. Kod RabbitMQ koji podrazumeva brokersku komponentu, perzistencija poruka je podržana.

Zadržavanje poruka. ZeroMQ ne podržava zadržavanje poruka. U slučaju da ne postoji krajnji primalac poruka koja je njemu namenjena biva odbačena, dok to nije slučaj kod RabbitMQ. RabbitMQ briše poruku iz reda tek kada je poruka stigla primaocu.

Performanse. Kako RabbitMQ podržava perzistenciju poruka i kako ima brokersku komponentu znatno je sporiji od ZeroMQ.

Propustna moć. ZeroMQ može da propusti oko 70000 poruka u jednoj sekundi dok RabbitMQ ima propustnu moć desetak puta manju.

Dizajn. ZeroMQ ne koristi brokera i omogućava filtriranje poruka na osnovu prefiksa teme poruka što omogućava primaocima da

primaju isključivo poruke one teme koje žele. RabbitMQ omogućava razmenu poruka tako što prosleđuje poruke preko komponente za razmenu poruka koja zatim poruke usmerava ka redovima sa kojih primaoci mogu da čitaju.

RabbitMQ više pažnje posvećuje skladištenju poruka, filtriranju i praćenju poruka dok ZeroMQ više pažnje usmerava ka na način razmene poruka preko mreže.

3.2. ZeroMQ i Kafka

Kafka je pub-sub sistem razmene poruka. On podrazumeva razmenu poruka između proizvođača poruka koji objavljuje poruke i konzumenta koji se prijavljuje na određene poruke. Poruke su podeljene u teme a u razmeni poruka posreduje broker koji prima poruke od proizvođača poruka i prosleđuje ih konzumentima.

Performanse. ZeroMQ je znatno brži u odnosu na Kafku jer ne zahteva skladištenje poruka na disku, što je podžano kod Kafke.

Skladište poruka. ZeroMQ skladišti poruke u malim baferima u memoriji, dok Kafka skladišti poruke na samom disku.

Garantovanje isporuke. ZeroMQ ne garantuje dostavljanje poruka dok Kafka garantuje.

Zadržavanje poruka. ZeroMQ ne podržava zadržavanje poruka. U slučaju da ne postoji krajnji primalac poruka koja je njemu namenjena biva odbačena dok je kod Kafke podržano zadržavanje poruka.

Kafka iako možda sporija tehnologija ipak raspolaže boljim performansama prilikom velike količine podataka i bolje podržano pristup u odnosu na ZeroMQ.

4. Zaključak

ZeroMQ je vrlo jednostavan sistem razmene poruka pogodan za slučajeve kada je potrebno obezbediti razmenu poruka bez ili sa minimalnim kašnjenjem. Ova biblioteka je podržana u velikom broju programskih jezika i podržava razne šablone razmene poruka. Ipak samo osnovni načini komunikacije i razmene poruka vrlo jednostavno se implementiraju dok svaki kompleksniji pristup zahteva više veštine prilikom implementacije. Ova karakteristika iako zapravo dovodi do otežanog korišćenja ZeroMQ biblioteke zbog povećanja kompleksnosti implementacije u slučaju nekih određenih načina komunikacije, ipak ima i pozitivnu stranu jer pruža slobodu da korisnik napravi i isprojektuje bilo koji tip komunikacije.

Kako ZeroMQ ima svoje vrline, tako ima i svoje mane pa shodno potrebama sistema i cilju razmene poruka može biti dobra ili loša opcija.

5. Literatura

[1] ZeroMQ - Documentation https://zeromq.org/get-started/
[2] CloudAMQP - What is message queuing
https://www.cloudamqp.com/blog/what-is-message-queuing.html
[3] Educba - ZeroMQ vs Kafka
https://www.educba.com/zeromq-vs-kafka/
[4] Educba - ZeroMQ vs RabbitMQ
https://www.educba.com/zeromq-vs-rabbitmq/