SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1211

Raspodijeljeno filtriranje tokova senzorskih podataka u računalnom grozdu

Josip Bago

Zagreb, veljača 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 6. listopada 2018.

DIPLOMSKI ZADATAK br. 1211

Pristupnik: **Josip Bago (0036468459)**
Studij: Informacijska i komunikacijska tehnologija
Profil: Telekomunikacije i informatika

Zadatak: **Raspodijeljeno filtriranje tokova senzorskih podataka u računalnom grozdu**

Opis zadatka:

Vaš zadatak je osmisliti, izvesti u programskom jeziku Java te testirati aplikaciju za računalni grozd za filtriranje senzorskih tokova podataka. Senzorska očitanja i trajni upiti sadrže geoprostorni objekt (npr. točku, liniju, poligon). Senzorsko očitanje se sastoji od sljedećih podataka: vrijeme očitanja, geoprostorni objekt koji predstavlja lokaciju očitanja, identifikator senzora, tip senzora, vrsta očitanja, vrijednost očitanja i jedinica očitanja. Trajni upit se sastoji od geoprostornog objekta koji predstavlja geografsko područje interesa te dodatnih uvjeta (npr. intervala vrijednosti očitanja ili ključnih riječi) na osnovu kojih se vrši kontinuirana usporedba s nadolazećim senzorskim očitanjima.
Prilikom izrade sustava iskoristite Javine programske knjižnice JTS (Java Topology Suite) i GeoTools, a senzorska očitanja i trajne upite realizirajte u formatu GeoJSON. Definirajte neophodna sučelja i apstraktne klase koje će omogućiti podršku za numeričke i tekstualne senzore. Istražite geoprostorne indekse koji su podržani u navedenim Javinim programskim knjižnicama, analizirajte njihove karakteristike i performanse te implementirajte raspodijeljeni indeks u računalnom grozdu.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 12. listopada 2018.
Rok za predaju rada: 8. veljače 2019.

Mentor:
Izv. prof. dr. sc. Krešimir Pripužić

Djelovođa:
Doc. dr. sc. Marin Vuković

Predsjednik odbora za diplomski rad profila:
Izv. prof. dr. sc. Miljenko Mikuc

Sadržaj

Uvod

U današnje vrijeme svjedoci smo trenda porasta broja samostalnih računalnih uređaja koji su spojeni na internet, obavljaju određenu funkciju, te imaju mogućnost primanja i odašiljanja podataka. Veliki broj takvih uređaja su različite vrste senzora koje prate raznorazne aktivnosti, te obavljaju primjerice meteorološka mjerenja, prate stanja u prometu i sl. Takvi uređaji stvaraju velike količine podataka koje se odašilju u internetsku mrežu unutar različitih sustava. Jedan od načina skupljanja i obrade, tj. okupljanja tih podataka jest implementacija sustava objavi-pretplati. Pretplate unutar takvog sustava omogućavaju filtriranje samo onih objava, tj. senzorskih očitanja, koja su od interesa za pretplatnika. Interes pretplatnika može biti izražen kroz zanimanje za primjerice objave koje dolaze samo sa određenog područja ili objave koje zadovoljavaju određene vrijednosti.

U okviru ovog rada razmatra se implementacija takvog sustava čiji je zadatak filtrirati senzorske objave na temelju postavljenih geoprostornih i vrijednosnih intervalnih uvjeta pojedinih pretplata. Ovaj rad polazi od pretpostavke da je za implementaciju takvog sustava koji zadovoljava svojstvo skalabilnosti do optimalne razine potrebno poduzeti određene korake, te se okrenuti prema konceptu izvedbe na raspodijeljenom sustavu. Naime, proces pronalaska pretplata čiji su uvjeti za svaku objavu zadovoljeni je ključna funkcionalnost takvog sustava, te ujedno i točka potencijalnog zagušenja. Rizik za pad sustava zbog preopterećenja javlja se iz dva razloga. Povećanjem broja pretplata povećava se vrijeme potrebno za filtriranje svake objave, a povećanjem intenziteta dolazaka objava uz konstantno trajanja filtriranja svake objave sustav se dovodi u nestabilno stanje. Raspodijeljeni sustav uvodi koncept paralelizma, a operacija filtriranja objava se potencijalno može izvoditi paralelno na više računala u primjerice računalnom grozdu.
Još jedan korak koji se može poduzeti uz raspodijeljenu obradu podataka, a koji se razmatra unutar ovog rada, jest pametna preraspodjela podataka, tj. pretplata. Time se može smanjiti broj pretraženih pretplata za filtriranje jedne objave. U tu svrhu predstavljaju se geoprostorne strategije indeksiranja. Razmatranjem programskih knjižnica koje omogućuju lakšu implementaciju raspodijeljenih

sustava koji radi sa geoprostornim podacima, te njihovih prednosti i nedostataka, dolazi se do potrebnih zaključaka o tome koje gotove alate iskoristiti za izradu predloženog sustava za raspodijeljeno filtriranje senzorskih objava na temelju pretplata. Sustav koji je implementiran, te opisan u okviru ovoga rada pruža funkcionalnosti unosa novih, te brisanja postojećih pretplata. Isto tako, sustav omogućava zaprimanje i spremanje objava, filtriranje objava na temelju postojećih pretplata, te povezivanje objava sa pretplatama čiji uvjeti obuhvaćaju značajke objava. Sustav koji je razvijen je modularan sustav sa REST API sučeljem koji se može koristiti kao samostalna aplikacija, ili primjerice mikro usluga unutar većeg sustava, te koji optimalno vrši proces filtriranja objava sa geoprostornim značajkama na računalnom grozdu.

1. Filtriranje geoprostornih objekata

Filtriranje senzorskih objava odrađuje se na temelju njihovih značajki. Pretplate su definirane trajnim upitima nad vrijednostima tih značajki. Trajni upiti pretplate postavljaju određene uvjete koje značajke objava moraju zadovoljavati kako bi se klasificirale kao objave od interesa za tu pretplatu. Značajke senzorskih objava sadrže vrijednosne i geoprostorne atribute. Aplikacija za filtriranje senzorskih objava koja je razvijena u okviru ovog rada podržava rad sa obje vrste atributa.

Objave sadrže vrijednosne atribute koji sadrže brojčane tipove podataka, te geoprostorne atribute koje čini skup vrijednosti koordinata. Skup koordinata tvori točku (jedan par koordinata) ili poligon (više parova koordinata) u geoprostoru. Vrijednosni i geoprostorni atributi zajedno čine geoprostorni objekt.

Pretplate sadrže informacije o trajnom upitu, te skup pridruženih koordinata koje tvore poligon, a time i područje interesa pretplate. Trajni upit sadrži jedinstvene vrijednosti ili intervale iz atributnog prostora značajki objava. Ukoliko područje interesa pretplate siječe ili obuhvaća područje senzorske objave, te ukoliko vrijednosti objave zadovoljavaju uvjete trajnog upita pretplate, objava će filtriranjem biti pridružena toj pretplati. Pretplatnici se ovakvom izvedbom mogu pretplatiti na njima zanimljiva geoprostorna područja, te biti obavješteni ukoliko primjerice senzori očitaju određene vrijednosti koje su zanimljive – npr. količina CO2 u zraku na istočnom dijelu grada Zagreba porasla je iznad 100mg/m2.

Ovakva izvedba zahtjeva kontinuirano iterativno izvođenje operacija usporedbe i utvrđivanja relacija između geoprostornih objekata kod svakog dolaska nove objave u sustav. Takve operacije mogu biti računalno izuzetno zahtjevne, te njihovo trajanje može učiniti sustav neskalabilnim. Ukoliko se navedene operacije obavljaju slijedno jedna za drugom, postoji velika vjerojatnost da će se stvoriti red čekanja i da će se sustav zagušiti kod većeg intenziteta dolazaka objava. Povećanjem broja pretplata (čime se povećava vrijeme usporedbe jedne objave sa svim pretplatama u sustavu) ili povećanjem broja objava (čime se povećava intenzitet dolazaka objava) smanjuje se propusnost sustava.

Rješavanju ovih nedostataka može se pristupiti na dva načina. U sustav bi se mogla uvesti paralelnost izvođenja operacija. Drugi način jest optimizacija procesa pretraživanja geoprostornih objekata pretplata u sustavu. Za adresiranje prvog načina rješavanja nedostatka sustava u ovom radu predlaže se rješenje koje implementira izvedbu na raspodijeljenom sustavu u nastojanju da se osigura skalabilnost sustava. Aplikacija za filtriranje objava koja je razvijena u okviru ovog rada podržava raspodijeljeno uspoređivanje objava nad pretplatama u računalnom grozdu (paralelno istovremeno uspoređivanje) koje obećava bolje performanse sustava. Adresiranje drugog pristupa sastoji se od uvođenja koncepta indeksiranja geoprostornih podataka. Kombiniranjem oba pristupa dobiva se željeno optimalno rješenje koje implementira raspodijeljeni geoprostorni indeks.

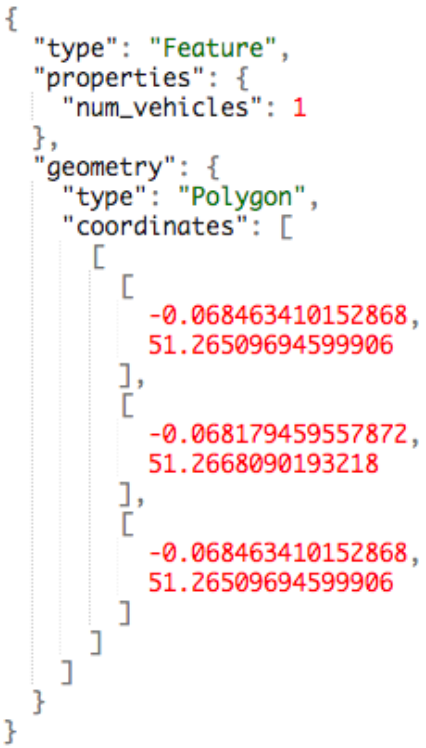
U nastojanju da se izvede ovakva aplikacija, u nastavku ovog poglavlja razmatraju se pojedine programske knjižnice koje se koriste za razvoj aplikacija koje funkcioniraju u raspodijeljenom okruženju - Apache Hadoop i Apache Spark. U tom razmatranju opisani su nedostaci koji se pojavljuju kod tih programskih knjižnica ukoliko se one koriste za implementaciju sustava koji rade sa geoprostornim podacima. Predstavljene su alternativne programske knjižnice koje su razvijene kao ekstenzija nad tim knjižnicama i koje rješavaju navedene nedostatke koristeći vlastitu implementaciju i novo razvijene koncepte, te postojeće programske knjižnice i standarde koji se koriste kod rada sa geoprostornim podacima u programskom jeziku Java. Primjeri takvih knjižnica su JTS, Geotools i Geospark. Rezultat razmatranja je odabir programske knjižnice Geospark koja je ekstenzija Apache Spark programske knjižnice ukoliko se radi sa geoprostornim podacima u raspodijeljenom okruženju. Geospark unutar sebe implementira programske knjižnice JTS i Geotools. Prikladnost programske knjižnice Geospark za izvedbu predloženog rješenja povećana je i činjenicom da Geospark nudi gotove implementacije raspodijeljenog geoprostornog indeksa.

1.1. Geoprostorni i vrijednosni atributi

Geoprostorni podaci, te njima pridružene vrijednosti mogu biti zapisane u nekoliko različitih formata: CSV, GeoJSON, WKT, NetDCF/HDF i ESRI Shapefile. Svaki od tih formata ima propisana pravila i konvencije, te vlastite prednosti i nedostatke. U okviru ovog rada odabran je format GeoJSON za spremanje i manipuliranje geoprostornim podacima. GeoJSON format je izravno kompatibilan u radu sa programskom knjižnicom JTS, a moguće ga je jednostavno serijalizirati u Java objekte pomoću metoda sadržanih u Geotools programskoj knjižnici. Prema tome, ovaj format je pogodan i za rad sa programskim knjižnicama koje implementiraju raspodijeljeno izvođenje operacija u sustavu, a koje u svojoj implementaciji koriste JTS i Geotools programske knjižnice. Primjer takve knjižnice je Geospark za koju će se pokazati da je prikladno rješenje za izvedbu aplikacije u okviru ovog rada.

1.2. GeoJSON

Senzorski podaci, te podaci o pretplatama koji se koriste u okviru ovog rada zapisani su u GeoJSON tekstualnom tipu podatka. GeoJSON je opće prihvaćen format kodiranja geoprostornih podataka koji definira način zapisa geoprostornih struktura u JSON formatu. GeoJSON podržava geometrijske tipove “Point”, “LineString”, “Polygon”, “MultiPoint”, “MultiLineString” i “MultiPolygon” koji su podrzani od strane JTS-a. Osim geoprostornih podataka, ovaj format može sadržavati i ostala svojstva. To su atributi koji primjerice mogu predstavljati izmjerene vrijednosti, mjerne jedinice objave, imena geoprostornog područja i sl.
Struktura GeoJSON formata može se vidjeti na primjerima na slici 1. Pod ključem “geometry” nalaze se geoprostorni podaci. Ključ “type” opisuje geometrijski tip, a pod ključem “coordinates” nalazi se lista koordinata koje definiraju geometrijski tip. Ključ “properties” sadrži ključeve atributa i njihove vrijednosti.



Slika 1. - Primjer GeoJSON zapisa geoprostornog objekta

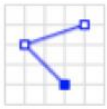
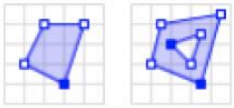
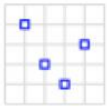
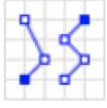
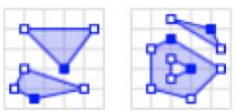
1.3. JTS – Java Topology Suite i Geotools

Rad sa geoprostornim podacima u programskom jeziku Java bio bi izuzetno zahtjevan bez javno dostupne programske knjižnice JTS. Ova programska knjižnica pruža gotov objektni model za Euklidsku planarnu linearnu geometriju zajedno sa skupom geometrijskih funkcija koje se mogu obavljati nad njime. JTS također definira standard za rad sa geometrijskim podacima za izgradnju geoprostornih aplikacija.

Objektni model sadrži klase koje modeliraju geometrije: “Point”, “LineString” i “Polygon”, te geometrijske kolekcije: “MultiPoint”, "MultiLineString" i "MultiPolygon". Za instanciranje tih objekata koristi se klasa tvornica GeometryFactory koja nudi odgovarajuće metode za instanciranje željenih objekata. Geometrije dostupne unutar JTS knjižnice mogu se vidjeti u tablici 1.

Geometrijske funkcije omogućuju određivanje relacijskih odnosa između dvije geometrije. Relacijski odnos opisuje da li se dvije geometrije presijecaju, dodiruju, križaju, prekrivaju, nalaze jedna unutar druge, ne presijecaju ili su jednake. Odnos koji se razmatra unutar pretplate u okviru ovog rada je presijecanje i prekrivanje.

U kombinaciji sa Geotools knjižnicom koja koristi JTS implementaciju, mogu se iskoristiti gotove implementacije serijalizatora za geometrije zapisane u GeoJSON formatu.

Naziv geometrijskog tipa	Slikovni prikaz	Opis geometrijskog tipa
Point		Točka u prostoru određena s jednom koordinatom, odnosno s uređenim parom x,y.
LineString		Skup točaka međusobno povezan dužinama. Svaka navedena točka tvori dužinu s prethodnom i definirane točke ne zatvaraju poligon.
Polygon		Ravnina omeđena s nizom dužina koje su pak kao i kod LineString-a određene s nizom točaka. Nužno je da su prva i zadnja točka jednake kako bi ravnina bila omeđena. Pri definiranju poligona s rupom kao na drugoj slici potrebno je navesti dva polja koordinata od kojih jedno definira poligon a drugo rupu u poligonu.
MultiPoint		Više točka u prostoru, svaka određena svojom koordinatom x,y
MultiLineString		Skup više LineString geometrijskih tipova, svaki od njih je zasebno definiran.
MultiPolygon		Skup više poligona, svaki od njih je zasebno definiran.

Tablica 1. - Geometrije dostupne u JTS

1.4. Geoprostorno indeksiranje

Bilo koja operacija pretraživanja nad skupom podataka može se optimizirati indeksiranjem. Indeksiranje može biti implementirano uzimajući u obzir operacije koje će se izvoditi nad podacima. Jedna od vrsta indeksiranja koja se može primijeniti nad geoprostornim podacima jest geoprostorno indeksiranje. To je proces slaganja geoprostornih objekata u stablastu strukturu prema geoprostornim odnosima objekata u skupu podataka.

Implementiranjem ovog koncepta nad skupom podataka nastoji se izbjeći slijedno pretraživanje geometrijskih objekata. Drugim riječima, postavljanjem upita za primjerice dohvaćanje svih objekata na području Londona, sustav neće pretraživati objekte koji se nalaze u Sjevernoj Americi. Unutar stablaste strukture, područje Velike Britanije se nalazi u odvojenoj grani, te sustav sa sigurnošću zna da geoprostorni objekti koji se nalaze u grani gdje su pohranjeni objekti u Sjevernoj Americi nisu oni koje pokušava naći. U okviru ovog rada razmotrene su dvije strukture, tj. strategije indeksiranja: R-Tree i Quad-Tree. Detaljnije informacije o ovim strategijama indeksiranja mogu se naći u radu [2].

1.1. Obrada podataka u raspodijeljenom okruženju

Raspodijeljeni sustav je sustav čije su komponente odvojeni mrežno povezani računalni sustavi koji komuniciraju jedni s drugima u svrhu raspodijeljenog obavljanja operacija. U takvom sustavu se mogu raspodijeljeno izvoditi operacije koje se mogu razdijeliti na više neovisnih koraka koji se mogu paralelno obavljati.

U okviru ovog rada koristi se raspodijeljeni sustav za povećanje skalabilnosti sustava za filtriranje objava na temelju pretplata. Kao raspodijeljeni sustav koristi se grozd računala koji ima jedno glavno računalo koje delegira obavljanje poslova pomoćnim računalima. Operacija filtriranja objava se sastoji od koraka usporedbe geoprostornih značajki objave sa trajnim upitima pretplata koje su pohranjene u sustavu. Podaci o pretplatama pohranjeni su na svim pomoćnim računalima

unutar grozda računala, a glavno računalo delegira obavljanje posla usporedbe za svaku objavu koja dođe u sustav jednom pomoćnom računalu. Pomoćno računalo nakon obavljanja operacije vraća rezultat glavnom računalu. Ovakvim načinom rada sustav poboljšava svojstvo skalabilnosti jer poslovi ne čekaju na izvođenje, već se delegiraju slobodnim resursima. Drugim riječima, sustav je spreman prihvatiti veći intenzitet dolazaka objava u sustav, te veći broj pretplata može biti pohranjen u sustavu, a da se ne naruši ispravan rad sustava.

Kompleksan proces implementacije bilo kojeg raspodijeljenog sustava, pa i sustava koji je razvijen u okviru ovog rada, može se pojednostaviti postojećim programskim knjižnicama i razvojnim okruženjima koja pružaju objektne i programske modele, te gotove metode za implementaciju takvih sustava. Primjeri takvih knjižnica i razvojnih okruženja su Apache Hadoop i Apache Spark. Iako je Apache Hadoop u današnje vrijeme zastarjela tehnologija koju je jednim dijelom zamijenio Apache Spark, rad Hadoop-a temelji se na bitnim konceptima za rad sa raspodijeljenim sustavima koje Apache Spark također koristi u svom radu. Programski model MapReduce je poslužio kao temelj u razvoju Apache Sparka, a Hadoop-ov HDFS datotečni sustav i danas je jedan od primarnih izbora za raspodijeljeno spremanje podataka koje Apache Spark može koristiti u svome radu. Apache Spark kao nadogradnja nad Hadoop-om pruža više funkcionalnosti ugrađenih u njegove module, te bolje performanse u vidu veće brzine izvođenja, posebice na iterativnim operacijama..

U okviru ovog rada iskorištena je programska knjižnica Geospark koja objedinjuje Apache Spark, programske knjižnice Geotools i JTS, te opcionalno HDFS datotečni sustav u jednu cjelinu koja pruža integrirane i apstrahirane funkcionalnosti raspodijeljene obrade geoprostornih podataka, te nudi gotovu implementaciju raspodijeljenog geoprostornog indeksa. Upotrebom programske knjižnice Geospark razvojni programer je pošteđen od implementiranja pojedinih funkcionalnosti koristeći prije navedene programske knjižnice jer Geospark objedinjuje u svojim metodama i objektnim modelima sve funkcionalnosti koje one pružaju. Osim toga, Geospark rješava mnogobrojne probleme i nedostatke koji se javljaju kod rada sa geoprostornim podacima u klasičnoj Spark implementaciji i Spark-ovim RDD-ovima. Implementacije u Geospark-u su pomno razrađene i

optimizirane za rad sa geoprostornim podacima. U nastavku ovog poglavlja ukratko se predstavljaju sve navedene programske knjižnice i koncepti koji se tiču implementacije raspodijeljenih sustava za obradu podataka, uključujući naposljetku i Geospark programsku knjižnicu.

1.1.1. Apache Hadoop

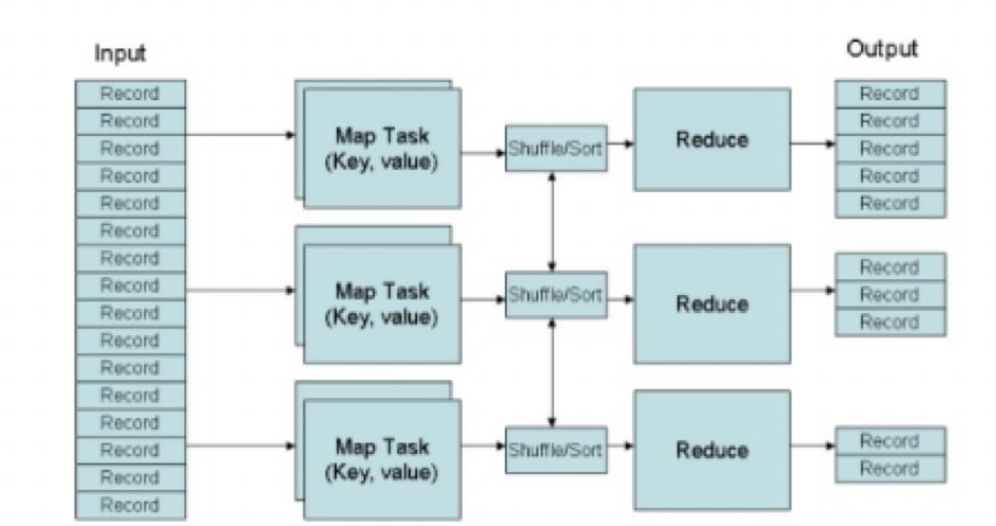
Apache Hadoop programska knjižnica je razvojno okruženje koje omogućava raspodijeljenu obradu velikih količina podataka u računalnom grozdu koristeći jednostavni programski model - MapReduce. Ovaj programski model dizajniran je na način da se može skalirati sa jednog na tisuće računala od kojih svako ima vlastiti procesor i lokalnu jedinicu za pohranu podataka.
Programski model MapReduce iznimno je pogodan za paralelnu algoritamsku obradu velikih količina podataka na grozdu računala. Temelji se na konceptu ključ-vrijednost za identificiranje i rad s podacima, te na konceptu raspodijele podataka u pod-skupove koji se paralelno obrađuju na zasebnim računalima. Nakon paralelne obrade, dobiveni rezultati se spajaju i agregiraju u konačni rezultat.

Koraci izvođenja programa u MapReduce programskom modelu:

1. "Input Split": U ovome prvom koraku podaci se raspodjeljuju u manje pod-skupove podataka.
2. "Mapping": Podaci se u svakom podskupu mapiraju i označuju ključevima te time tvore parove ključ-vrijednost. Ova operacija izvodi se po nekoj unaprijed određenoj logici. Primjerice, operacija mapiranja može se raditi tako da se iz neke velike tekstualne datoteke izbroji broj ponavljanja svake riječi, te se podaci mapiraju tako da je riječ ključ, a broj ponavljanja iste riječi vrijednost.
3. "Shuffling": U ovom koraku prethodno podijeljeni pod-skupovi sa novim ključ-vrijednost podacima se grupiraju ponovno u jedan skup kombiniranjem vrijednosti za svaki ključ iz svakog podskupa podataka.

4. "Reducing": U ovom koraku se kombinirane vrijednosti obrađuju za svaki ključ po određenoj operaciji. U slučaju brojanja ponavljanja riječi, kombinirane vrijednosti iz svakog skupa podataka za svaki ključ, tj. riječ, se zbrajaju te se dobije konačan broj ponavljanja svake riječi iz cijele tekstualne datoteke.

Proces toka podataka u modelu MapReduce može se vidjeti na slici 5.



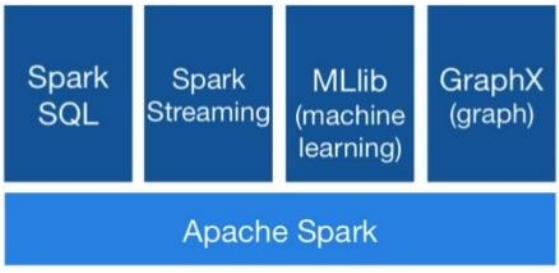
Slika 5. - Prikaz koraka u MapReduce programskom modelu

Druga temeljna osnova funkcioniranja Hadoop-a jest pohrana podataka u posebno definiranom datotečnom sustavu – Hadoop Distributed File System (HDFS).
HDFS datotečni sustav obuhvaća cjelokupni grozd računala i omogućava brz prijenos podataka između pojedinih računala u grozdu. Kada HDFS zaprimi podatke, on ih sprema u razdvojene blokove te ih raspoređuje na sva računala u grozdu i time omogućava konzistentno stanje između različitih podatkovnih sustava pojedinih računala i time omogućava istovremeno paralelno izvođenje operacija nad podacima. Još jedna značajka koja je rezultat prethodno navedenih svojstava HDFS-a, tj. replikacije podataka, jest visoka otpornost na ispade i pogreške u radu.

1.1.2. Apache Spark

Apache Spark je otvoreno javno dostupno okruženje za razvoj programskih riješenja koja rade na raspodijeljenim sustavima i koja inherentno u svojoj implementaciji imaju ugrađena svojstva otpornosti na ispade, te svojstva koja omogućavaju paralelno izvođenje operacija. Spark pruža apstraktno sučelje preko kojega se može razvijati programska rješenja koja su upogonjena na grozdovima računala. Apache Spark je svojevrsna nadogradnja Apache Hadoop-a koja pruža mnogo vise funkcionalnosti od Hadoop-ovih funkcija mapiranja i reduciranja, te u određenim slučajevima pruža znatno bolje performanse.

Dok Hadoop pruža paralelno izvođenje operacija u blokovima isključivo čitajući podatke sa diska, Apache Spark koristi radnu memoriju za brz pristup, pisanje i čitanje privremenih podataka u međukoracima. Hadoop-ov MapReduce je prikladan programski model za statičnu obradu velike količine podataka, ali gubi na performansama ukoliko je riječ o iterativnim algoritmima obrade velike količine podataka. Vrijeme izvođenja iterativnih operacija je sporo na Hadoop-u jer se podaci čitaju i zapisuju na hard diskovima što traje i stvara usko grlo u radu sustava. Apache Spark sa spremanjem podataka u RDD-ove pruža bolje performanse za iterativne operacije. Eksperimentalno je pokazano da za iterativne algoritme Spark nudi ubrzanje i do dvadeset puta povrh MapReduce algoritma [1]. Isto tako, Apache Spark sadrži više različitih modula poput Spark SQL, Spark Streaming, Mllib, GraphX modula uz primarni Spark Core modul koji pružaju mnogobrojne funkcionalnosti (Slika 6.).



Slika 6.

Arhitektura Spark-a je utemeljena na upotrebi apstrakcije prema raspodijeljenim podacima u obliku otpornih raspodijeljenih podatkovnih skupova - RDD ("Resilient

Distributed Dataset"). RDD je apstraktni prikaz podatkovnog skup podataka koji je nepromjenjiv (može se samo čitati) i koji je raspodijeljen u grozdu računala. Apache Spark može koristiti, te najčešće i koristi HDFS (može koristiti i ostale datotečne sustave), te ima svojstvo otpornosti na greške i ispade. Sparkov Dataset API pruža sučelje za stvaranje i upravljanje RDD-ovima. Druga apstrakcija na kojoj se temelji rad Spark-a jesu dijeljene varijable koje se koriste za prijenos informacija između paralelno izvođenih procesa. Spark implementira vlastiti Kryo serijalizator kao alternativu Java serijalizatoru za efikasniji prijenos objekata između čvorova u grozdu.

Moduli unutar Spark-a:

- Spark Core - Osnova projekta koja pruža funkcionalnosti aktiviranja, praćenja, te zakazivanja raspodijeljenih operacija i operacija čitanja i pisanja u datotečni sustav koje su dostupne kroz RDD apstrakciju.
- Spark SQL - Pruža mogućnosti indeksiranja, te postavljanja upita nad podacima spremljenima u RDD-ovima preko DSL jezika (domenski specifičnog jezika za postavljanje upita). Ove funkcionalnosti su ostvarene putem objekta DataFrame koji sa svojim ugrađenim funkcionalnostima pruža prikladnu apstrakciju nad podacima
- Spark Streaming - Modul koji ostvaruje mogućnost podjele podataka u male skupove nad kojima se mogu izvoditi paralelne operacije, te služi za obradu podataka koji dolaze često i u kontinuitetu.
- MLib - Implementacija programskog okruženja za strojno učenje koja se može izvoditi na raspodijeljenom sustavu
- GraphX - sučelje za prikaz podataka spremljenih u RDD-ovima putem grafova

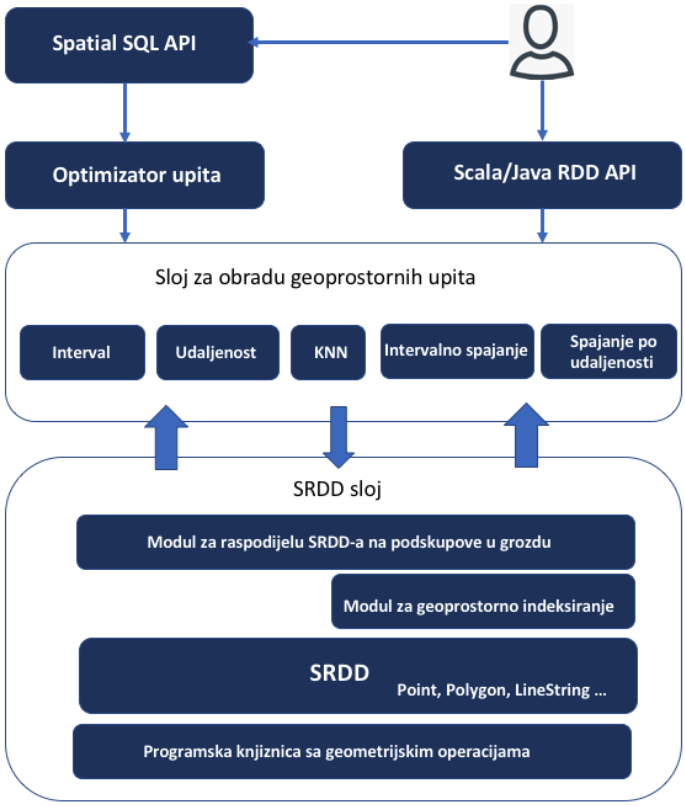
Sustav za filtriranje objava nad pretplatama temelji se na iterativnom obavljanju operacija koje se paraleliziraju. Zbog navedenih svojstava programske knjižnice Apache Spark, te boljih performansi od Apache Hadoop programske knjižnice kod slučajeva gdje se obavljaju iterativne operacije, za svrhu izrade raspodijeljenog sustava kao osnova se odabire Apache Spark programska knjižnica.

1.2. Geospark

Apache Spark programska knjižnica pruža mogućnosti implementacije sustava koji paralelno izvršava operacije na grozdu računala. Ono što nedostaje jest podrška za rad sa geoprostornim podacima, tj. metode i objektni modeli za rad u geoprostornom atributnom prostoru. U tu svrhu razvijena je programska knjižnica Geospark kao ekstenzija nad Apache Spark-om. Geospark omogućava paralelno izvođenje operacija nad geoprostornim podacima na grozdu računala. Apache Spark pokazuje mnogobrojne slabosti ukoliko se nativna implementacija koristi za rad sa geoprostornim podacima bez pomne razrade i optimizacija nad učitavanjem podataka, izvođenjem operacija itd. Primjerice, Apache Spark ne implementira geoprostorne indekse, Kryo i Java serijalizatori dostupni u Sparku nisu optimizirani za prijenos opsežnih geoprostornih objekata između računala u grozdu, geoprostorni podaci nisu optimalno raspodijeljeni u RDD-u, a sami geoprostorni podaci mogu biti spremljeni u nekoliko različitih formata koji ukoliko nisu pomno razmotreni mogu dovesti do implementacija učitavanja i obrade koje nisu efikasne.

Spark moduli koje Geospark koristi su Spark Core, te Spark SQL koji su prošireni sa mogućnostima geoprostornog indeksiranja, te obavljanjem operacija i postavljanjem upita nad geoprostornim objektima u obliku geoprostornih relacija. Geospark proširuje Spark-ove RDD-ove sa metodama i atributima, te konceptima koji podržavaju rad sa geoprostornim podacima. Prošireni RDD-ovi u Geospark-u nazivaju se SRDD-ovima (Spatial Resilient Distributed Dataset). Geospark implementira programski sloj koji se koristi sučeljima na SRDD-ove za izvođenje kompleksnih geoprostornih upita. Sloj za obradu upita sadrži logiku izvođenja pojedinih koraka upita za vrste upita koji su podržani od strane Geospark-a. Optimizator upita služi kao ulazna točka u sloj za obradu upita za tekstualne SQL upite koji su proslijeđeni putem Spatial SQL API-ja. Optimizator na temelju izraza upita stvara optimalan plan izvođenja s obzirom na predviđeno trajanje koraka za svaku moguću varijaciju izvođenja. Razvojni programer može pristupati Geospark sučeljima koristeći Spatial SQL API ili Scala/Java RDD API. Programska

arhitektura koja uključuje sve prethodno navedene dijelove Geospark-a može se vidjeti na slici 7.



Slika 7. - Arhitektura Geospark-a

1.2.1. Spatial RDD sloj (SRDD)

Geosparkovi SRDD-ovi su pohranjene instance raspodijeljenih podatkovnih skupova koje proširuju funkcionalnosti i značajke Spark RDD-ova sa metodama za rad sa geoprostornim objektima i njihovim atributima.
SRDD rješava nekoliko problema koji se pojavljuju kod rada sa geoprostornim podacima na klasičnom Sparkovom RDD-u.

- Raznovrsni izvori podataka - Geoprostorni podaci mogu biti spremljeni u različitim formatima: CSV, GeoJSON, WKT, NetDCF/HDF i ESRI Shapefile. Spark ne sadrži implementaciju koja prihvaća i razumije ove različite formate i ne nudi nativno gotova rješenja učitavanja podataka iz ovakvih

datoteka. To implicira da ručna implementacija učitavanja podataka u Spark-u može dovesti do manjkavih rješenja koja na neefikasan način obrađuju podatke. Geospark implementira optimalne metode unutar SRDD-ova za rad sa različitim formatima datoteka i njihovim specifičnim tipovima pohrane geoprostornih podataka.

- Kompleksni geometrijski oblici - Postoji mnogo vrsta geometrijskih objekata koji mogu poprimiti različite oblike: točke, poligone, konveksne i konkavne oblike, geometrije sa podgeometrijama itd. Uz to, jedna kolekcija geoprostornih objekata može sadržavati nekoliko različitih vrsta geometrijskih objekata, pa i njihove podkolekcije. Ovakvi objekti ne mogu na efikasan način biti podijeljeni u raspodijeljenom podatkovnom sustavu, serijalizirani, te se nad njima teško mogu efikasno provoditi upiti nad geoprostornim značajkama u Sparkovima klasičnim RDD-ovima. SRDD nudi optimizirane metode i objektne modele koji nude optimalnu implementaciju za izvršavanje svih prethodno navedenih operacija nad kompleksnim geometrijskim oblicima i kolekcijama.

- Raspodjela geoprostornih objekata u podskupove podataka - Klasična Spark implementacija kod raspodijele podataka u podskupove prije koraka mapiranja (jedan od koraka u MapReduce programskom modelu koji se koristi u Spark-u) ne uzima u obzir blizinu pojedinih geoprostornih objekata u podatkovnom skupu. Za optimalno izvođenje operacija ključno je da objekti koji su blizu jedan drugom u geoprostornoj prezentaciji prema koordinatama koje su sadržane u geometrijama budu u istom podatkovnom podskupu. Ukoliko su pod-skupovi raspoređeni na taj način, kod izvođenja upita operacije se ne izvode na svim pod-skupovima u pohranjenom RDD-u. SRDD se brine da nakon raspodijele podatkovni pod-skupovi i njihov sadržaj budu raspoređeni na taj način.

- Podrška za geoprostorno indeksiranje - Spark implementacija ne podržava geoprostorno indeksiranje kao sto je Quad-Tree ili R-Tree. Održavanje stablaste strukture koja globalno stvara takve indeksne strukture u Sparku nosi sa sobom potrebu za 15% popunjenijim kapacitetom spremanja

podataka na svakom računalu što predstavlja veliki problem ukoliko je količina podataka jako velika. SRDD sadrži ugrađene koncepte koji omogućavaju optimalnu lokalnu preraspodjelu po indeksima prema potrebama geoprostornih upita.

1.2.2. SRDD objekti i funkcije

Geospark koristeći JTS i Geotools knjižnice podržava rad sa geoprostornim objektima: Point, Multi-Point, Polygon, Multi-Polygon, LineString, Multi-LineString, GeometryCollection i Circle. SRDD može sadržavati bilo koju kombinaciju ovih objekata.

Ugrađene funkcije za paralelno izvođenje geometrijskih operacija na SRDD-ovima pružaju sučelje sloju iznad SRDD sloja (sloju za izvođenje geoprostornih upita) za efikasno izvođenje geometrijskih operacija kao što su detekcija presijecanja geometrijskih oblika, sadržavanja jednog oblika u drugome i sl.

1.2.3. SRDD raspodijela podataka

Apache Spark nakon učitavanja podatkovnog skupa podijeli podatke na pod-skupove indeksiranjem pomoću hash zapisa ili prateći HDFS datotečnu strukturu, te svaki podskup proslijedi jednom računalu u grozdu. Pritom se Apache Spark ne brine o očuvanju bliskosti geoprostornih objekata u pod-skupovima.

Geospark radi prostorne preraspodijele unutar SRDD-a kako bi očuvao bliskost objekata u pojedinim pod-skupovima. Ova se operacija izvodi algoritmom geoprostornog indeksiranja. Geoprostornim indeksiranje je postignuto stanje u kojem se primjerice upit za spajanje geoprostornih objekata sa određenim značajkama izvodi brže jer je za taj upit potrebno uključiti manje čvorova u grozdu koji se provjeravaju. Za neki upit koji radi spajanje po geoprostornim značajkama koje su prisutne u Velikoj Britaniji sustav ne uključuje čvorove koji sadrže primjerice geoprostorne objekte u Sjevernoj Americi. Kod indeksirane stablaste strukture pod-skupova geoprostornih objekata, upit neće ulaziti u grane stabla u kojima zna da nema geoprostornih objekata obuhvaćenih uvjetima upita.

Prostorna preraspodjela SRDD-a je računalno zahtjevna operacija, no Geospark implementira optimalan algoritam za smanjenje vremena i potrebnih resursa za izvođenje preraspodijele i inicijalizaciju ispravnog prostorno raspodijeljenog SRDD-a čiji se pseudokod može vidjeti na slici 1. Bitno je napomenuti i da Geospark uzima u obzir i potrebe aplikacije tako da ne radi globalnu preraspodjelu, već smanjuje vrijeme izvođenja algoritma radeći lokalne preraspodijele medu čvorovima prema potrebama upita. Isto tako, Geospark lokalno dobivene indekse u slučaju čestih sličnih upita privremeno pohranjuje, te ih iskorištava ponovno čime se radi dodatna velika ušteda vremena izvođenja.

Algorithm 1 SRDD spatial partitioning

```
   Data: An original SRDD
   Result: A repartitioned SRDD
   /* Step 1: Build a global grid file at master node                    */
 1 Take samples from the original SRDD A partitions in parallel;
 2 Construct the selected spatial structure on the collected sample at master node;
 3 Retrieve the grids from built spatial structures;
   /* Step 2: Assign grid ID to each object in parallel                  */
 4 foreach spatial object in SRDD A do
 5 |   foreach grid do
 6 |   |   if the grid intersects the object then
 7 |   |   |   Add (grid ID, object) pair into SRDD B;
   |   // Only needed for R-Tree partitioning
 8 |   if no grid intersects the object then
 9 |   |   Add (overflow grid ID, object) pair into SRDD B;
   /* Step 3: Repartition SRDD across the cluster                        */
10 Partition SRDD B by ID and get SRDD C;
11 Cache the new SRDD C in memory and return it;
```

Slika 8. - Koraci i pseudokod algoritma za geoprostornu raspodjelu

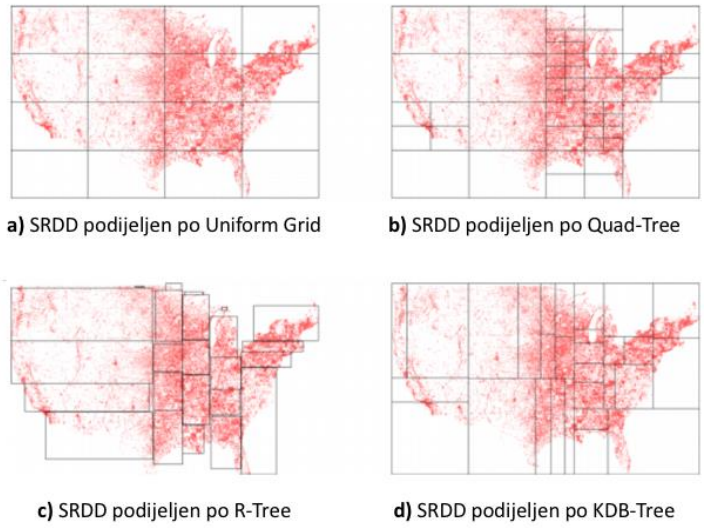
1. Korak: Izgradnja globalne prostorne podjele na sektore - U ovom koraku sustav uzima uzorke iz svakog SRDD podskupa i skuplja ih u Spark glavnom čvoru u grozdu gdje izgrađuje mali podskup podataka koji sadrži geoprostorne objekte iz svakog inicijalno raspodijeljenog SRDD podskupa. Zatim na tom podskupu koji je skupljen na glavnom čvoru sustav izgradi jedan od zadanih indeksa i podijeli podatke na geoprostorne sektore.

Indeksi koje Geospark podržava su, R-Tree, Quad-Tree i KDB-Tree, a pojedine podijele na sektore nakon indeksiranja mogu se vidjeli na slici 9:

- Uniform Grid - Geospark podijeli dvodimenzionalni geoprostor na sektore jednakih dimenzija. Ovakav način indeksiranja pogoduje geoprostorno uniformno raspodijeljenim objektima.

- R-Tree, Quad-Tree, KDB-Tree - Geospark uzima u obzir gustoću geoprostornih objekata na pojedinim prostorima u dvodimenzionalnom geoprostoru. Nakon indeksiranja geoprostorni objekti su smješteni u stablastu strukturu unutar koje su skupovi-djeca pojedinih skupova-roditelja oni skupovi koji se nalaze obgrljeni unutar skupa roditelja.

Neki objekti koji nakon indeksiranja ne upadnu u nijedan sektor (slučaj kod Uniform Grid indeksiranja) smještaju se u overflow podskup.



Slika 9. - Primjeri podijele na sektore ovisno o geoprostornom indeksiranju (preuzeto iz rada [1])

2. Korak: Dodjela identifikacijskog broja dobivenog sektora svakom geoprostornom objektu - U ovom koraku svaki geoprostorni objekt u SRDD-u se smješta u jedan od sektora dobivenih u prvom koraku dodjelom identifikacijskog broja sektora kojem objekt pripada. To se odrađuje tako da

se podskup sa indeksiranim zapisom sa glavnog čvora prekopira na slave čvorove. Svaki slave čvor izgradi novi SRDD sa vrijednostima ključ-vrijednost gdje je ključ identifikacijski broj sektora. Ova se operacija odrađuje paralelno na svakom čvoru i u ovom koraku se za izgradnju SRDD-a koristi cijeli određeni dio grozda računala.

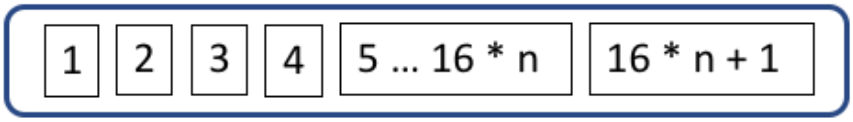
3. Korak: Preraspodjela geoprostornih objekata u SRDD-u - Nakon dodijele identifikacijskih brojeva sektora svakom geoprostornom objektu u svakom slave čvoru, Geospark radi preraspodjelu među lokalno indeksiranim slave čvorovima.

3.1.1. SRDD serijalizacija

Prilikom recimo prostorne preraspodijele, tj. geoprostornog indeksiranja, geoprostorni objekti se prebacuju sa jednog slave čvora na drugi. Apache Spark i Kryo serijalizacija nije dobro rješenje ukoliko su geoprostorni objekti veliki jer sadrže kompleksna geometrijska svojstva. Isto tako, ukoliko je SRDD geoprostorno indeksiran, prenose se informacije i o geoprostornom indeksu. U tom slučaju velika količina podataka tijekom preraspodijele se prenosi preko mreže, te se povećava vrijeme prijenosa objekata i velika količina mrežnih i računalnih resursa se koristi za prijenos, serijalizaciju i deserijalizaciju objekata. Taj se proces može optimizirati. Geospark pruža optimizaciju tog procesa unutar SRDD-a implementacijom posebnog serijalizatora geoprostornih objekata i geoprostornog indeksa. Serijalizator slaže posebnu byte strukturu za prijenos geoprostornog objekta koja komprimira podatke o geometriji i značajkama objekta koja se može vidjeti na slici 10.

- Byte 1 - Određuje koji je tip geoprostornog objekta. Unutar Geosparka svaki tip objekta je enkodiran u jedan byte.
- Byte 2 - Određuje broj geoprostonih pod-objekata u geoprostornom objektu koji se serijalizira.
- Byte 3 - Ukoliko je riječ o geometrijskoj kolekciji koja se serijalizira, ovaj byte određuje tip prvog pod-objekta u kolekciji.

- Byte 4 - Određuje broj koordinata koje se nalaze u prvom geoprostornom pod-objektu.
- Byte 5 do Byte 4 + 16*n - Sprema informacije o n broju koordinata.
- Byte 16*n + 1 - Sprema informacije o broju koordinata na idućem pod-objektu.



Slika 10. - Serijalizirani geoprostorni objekt/kolekcija

Serijalizacija geoprostornih objekata je dio serijalizacije geoprostornog indeksa za koji se koristi stablasta N struktura koja se izvodi N-stablastim serijalizacijskim algoritmom.

Serijalizacija indeksne strukture koristi DFS (Depth-First Search) algoritam za prolazak kroz svaki list stablaste strukture indeksa. Prolaskom kroz stablo, kod nailaska na novi čvor prvo se serijaliziraju geoprostorne granice određene geometrijama objekata u tom čvoru, a zatim se provodi prethodno opisana serijalizacija geoprostornih objekata. Deserijalizacija također koristi DFS algoritam. Prvo se deserijaliziraju granice i objekti čvorova roditelja. Algoritam postupno deserijalizira cijelo stablo.

3.1.2. Sloj za izvođenje geoprostornih upita

Nakon što je SRDD u računalnom grozdu posložen i indeksiran ovaj sloj može putem sučelja na SRDD prosljeđivati kompleksne geoprostorne upite. Geospark unutar ovog sloja podržava nekoliko varijacija geoprostornih upita: upit za izračun udaljenosti, intervalni upit, KNN (K nearest neighbours) upit, intervalni upit za spajanje i upit za izračun udaljenosti nakon spajanja.
Uz sloj za izvođenje upita koji sadrži općenitu logiku izvođenja algoritama za obavljanje upita koji se sastoje od operacija manipulacije i pretraživanja SRDD-a, nalazi se još i optimizator izvođenja geoprostornih SQL upita. Optimizator zaprima

SQL izraz upita, te na temelju tog izraza stvara optimalan plan izvođenja s obzirom na predviđeno trajanje koraka za svaku moguću varijaciju izvođenja.

3.1.3. Geospark API i funkcionalnosti

Geospark-ov API pruža mogućnosti instanciranja SRDD-ova, te pristup apstraktnim funkcionalnostima koje prikrivaju složene operacije kroz izvođenje geoprostornih upita. Kod instanciranja SRDD-ova Geospark pruža mogućnost korištenja univerzalnog SRDD-a (može sadržavati sve vrste geoprostornih objekata), te četiri podvrste SRDD-a: PointRDD, PolygonRDD, CircleRDD te LineStringRDD. Podaci mogu biti učitani iz datoteka u WKT, WKB, GeoJSON, ESRI Shapefile, te CSV/TSC formatiu. Nakon učitavanja podataka može se odabrati geoprostorni indeks koji se želi koristiti na SRDD-u. Instanciranje geometrija geoprostornih objekata unutar SRDD-ova i upiti se mogu implementirati putem API-ja sadržanog u Geospark core modulu kroz ugrađene statičke Java metode, te putem Geospark SQL-a pomoću funkcija koje se koriste u SQL upitu kroz pokrenutu Spark sesiju putem Spatial SQL API-ja. Sve funkcionalnosti i mogućnosti Geospark-a mogu se predstaviti kroz popis funkcija koje nudi Geospark API.

Geospark SQL funkcije (Spatial SQL API):

- Instanciranje geometrija
    - ST_GeomFromWKT(Wkt:string) - konstruira geometriju iz WKT datoteke.
    - ST_GeomFromGeoJSON(GeoJson:string) - konstruira geometriju iz WKB datoteke.
    - ST_Point(X:decimal, Y:decimal) - konstruira geometriju točke
    - ST_PointFromText(Text:string, Delimiter:char) - konstruira geometriju točke iz teksta gdje su koordinate odvojene delimiterom

    - ST_PolygonFromText(Text:string, Delimiter:char) - konstruira geometriju poligona iz teksta gdje su koordinate odvojene razdvojenim znakom koji se predaje kao argument
    - ST_LineStringFromText(Text:string, Delimiter:char) - konstruira geometriju linije iz teksta gdje su koordinate odvojene razdvojenim znakom koji se predaje kao argument
    - ST_PolygonFromEnvelope(MinX:decimal,MinY:decimal, MaxX:decimal, MaxY:decimal) - konstruira geometriju poligona iz specifikacije četiri točke
    - ST_Circle(A:Geometry, Radius:decimal) - konstruira geometriju kruga koristeći referencu na drugu geometriju i iznos radijusa kruga

- SQL funkcije nad geometrijama
    - ST_Distance(A:geometry, B:geometry) - vraća Euklidsku udaljenost između geometrija A i B
    - ST_ConvexHull(A:geometry) - vraća geometriju konveksnog opsega oko zadane geometrije
    - ST_Envelope(A:geometry) - konstruira geometriju opsega oko geometrije
    - ST_Length(A:geometry) - vraća opseg oko geometrije
    - ST_Area(A:geometry) - vraća površinu geometrije
    - ST_Centroid(A:geometry) - vraća točku centroid geometrije
    - ST_Intersection(A:geometry, B:geometry) - vraća presjek izmedu dvije geometrije
    - ST_IsValid(A:geometry) - provjerava ispravnost definicije geometrije ovisno o njenom tipu
    - ST_PrecisionReduce(A:geometry, B:int) - smanjuje broj decimala u koordinatama na B
    - ST_Contains(A:geometry, B:geometry) - vraća true ukoliko A sadrži unutar sebe B
    - ST_Intersects(A:geometry, B:geometry) - vraća true ukoliko se A i B sijeku
    - ST_Within(A:geometry, B:geometry) - vraća true ako je A potpuno unutar B

- ST_Equals(A:geometry, B:geometry) - vraća true ako je A geometrija jednaka B
- ST_Crosses(A:geometry, B:geometry) - vraća true ako A prelazi preko B
- ST_Touches(A:geometry, B:geometry) - vraća true ako se A i B dodiruju
- ST_Envelope_Aggr(A:geometryCollection) - vraća geometriju omotnicu oko svih geometrija u A
- ST_Union_Aggr(A:geometryCollection) - vraća poligon koji je uniija svih poligona u kolekciji poligona A

Primjer jednog SQL upita koji koristi Spatial SQL API može se vidjeti na slici 11. Putem ovog SQL upita korisnik izdvaja sve rute kojima su vozili taksiji iz SRDD-a u kojemu su pohranjene rute, a čiju točku preuzimanja klijenta taksija obuhvaćaju geometrije koje određuju područje Zagreba i Splita.

```
SELECT * FROM TablicaTaksiRute
WHERE ST_Contains(Zagreb,
TablicaTaksiRute.tocka_preuzimanja)
OR ST_CONTAINS(Split, TablicaTaksiRute.tocka_preuzimanja)
```

Slika 11. - Primjer geoprostornog upita putem Spatial SQL zapisa

Geospark Java API:

- Instanciranje geoprostornih objekata
  Instanciranje geoprostornih objekata se kroz Geospark Java API radi u nekoliko koraka koristeći klase koje su sadržane u Geospark i JTS programskim knjižnicama. Podaci se mogu učitati na više različitih načina.

  - Učitavanje u SRDD sa predodređenim tipom geometrija geoprostornih objekata iz datoteke - Instanciranje SRDD-ova na ovaj način se radi instanciranjem podvrste SRDD-a, te predajom

parametara koji specificiraju tip datoteke i dodatne značajke geoprostornih podataka skupa sa URI putem do datoteke sa podacima u konstruktor. Klase koje se instanciraju mogu biti PointRDD, PolygonRDD, LineStringRDD, CircleRDD

- Učitavanje u SRDD sa generičkim tipom geometrija, koji može sadržavati mješavinu tipova geometrija, iz datoteke - Prvi korak se vrši upotrebom statičkih metoda klasa za učitavanje datoteka sa različitim formatima. Za tu svrhu postoje klase GeoJsonReader, ShapeFileReader, DbfFileReader, te WktFileReader. Drugi korak je izvršavanje SQL upita za instanciranje geometrija služeci se metodama iz SQL API-ja čime se dobije Dataset objekt. Treći korak jest instanciranje generičkog SRDD-a statičkom metodom Adapter.toRdd(data:dataset)

- Učitavanje u SRDD iz postojećeg JavaRDD-a sa predodređenim tipom geometrije - Instanciranje SRDD-ova na ovaj način se radi instanciranjem podvrste SRDD-a, te predajom parametara koji opisuju dodatne značajke geoprostornih podataka skupa sa JavaRDD-om u kojem su pohranjeni geoprostorni podaci u konstruktor. Klase koje se instanciraju mogu biti PointRDD, PolygonRDD, CircleRDD, LineStringRDD

- Funkcije
  - Geospark implementira klase sa statičkim metodama putem kojih se može izvršiti neke geoprostorne upite. Klase koje implementiraju te metode su: JoinQuery, KNNQuery, te RangeQuery. Putem ovih metoda moguće je ostvariti jedan dio operacija koje su sadržane u Spatial SQL API-ju programski koristeći isključivo Java kod.

4. Aplikacija za filtriranje senzorskih tokova podataka

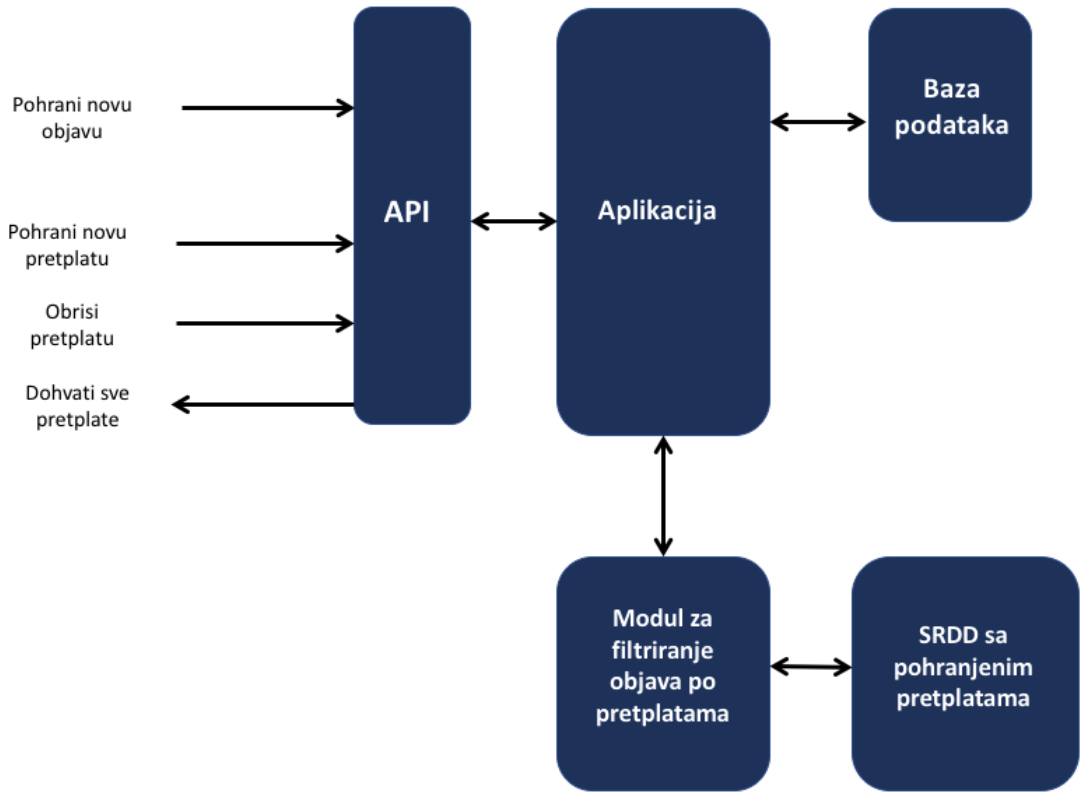
Aplikacija za računalni grozd za filtriranje senzorskih tokova podataka koja je predložena i razvijena u sklopu ovog rada mora omogućiti obradu ulaznih podataka (objave i pretplate). Nakon zaprimanja pretplate sustav mora pretplatu pohraniti. Nakon zaprimanja objave sustav mora objavu pohraniti, te nakon provedene operacije filtriranja i spremanja objave, povezati objavu sa pretplatama čiji uvjeti obuhvaćaju vrijednosti atributa objave. Ulazni su podaci geoprostorni objekti sa senzorskim očitanjima kao dodatnim atributima čija je lokacija određena lokacijom senzora sa kojeg dolaze.

Filtriranje podataka se vrši na osnovu pretplata koje su pohranjene u sustavu, a koje obuhvaćaju određeno geografsko područje i definiraju intervalni uvjet nad dodatnim atributom senzorskog očitanja koji je brojčana vrijednost. Geoprostorni odnos koji se provjerava mora zadovoljavati svojstvo presijecanja ili obuhvaćanja. Operacije filtriranja senzorskih podataka po geoprostornim uvjetima pretplate se odrađuju raspodijeljeno u računalnom grozdu radi poboljšanja svojstva skalabilnosti sustava. Aplikacija osim spremanja pretplata i objava mora omogućiti i dohvaćanje svih pretplata, te objava koje su povijesno pridružene toj pretplati.

4.1. Arhitektura sustava

Aplikacija je izvedena kao REST usluga koji putem vlastitog API-ja omogućava ulazne i izlazne tokove podataka, tj. senzorskih objava i pretplata. Putem REST API-ja podaci o objavama i pretplatama se isporučuju u JSON formatu, točnije JSON formatu propisanom GeoJSON standardom. Podaci se spremaju u bazu podataka pridodijeljenu sustavu. Implementacija usluga omogućava CRUD operacije nad objavama i pretplatama i sadrži poslovnu logiku za serijalizaciju i deserijalizaciju podataka iz JSON formata u objekte u sustavu i obrnuto, te poslovnu logiku za zaprimanje objava i pretplata i njihovu obradu prije filtriranja i spremanja u bazu podataka. Unutar aplikacije nalazi se odvojeni modul za

raspodijeljeno filtriranje objava na temelju pretplata koje su definirane u sustavu. Raspodijeljeni modul filtrira podatke na raspodijeljenom podatkovnom sustavu u kojem su pohranjene pretplate. Pošto su raspodijeljeni podatkovni sustav i baza podataka odvojene instance, bitno je napomenuti da se njihova stanja održavaju konzistentnima. Sustav je razvijen u troslojnoj arhitekturi kontroler-usluga-podatkovni pristup. Kontroler sustava pruža API sučelje prema vanjskom svijetu. Uslužni sloj implementira poslovnu logiku sustava. U uslužnom sloju osim modula za poslovnu logiku sustava postoji i modul za filtriranje objava po pretplatama. Podatkovni pristup predstavlja međusloj između uslužnog sloja i baze podataka, te se brine za obavljanje operacija upisa u bazu i čitanja iz baze. Općeniti nacrt sustava može se vidjeti na slici 12.



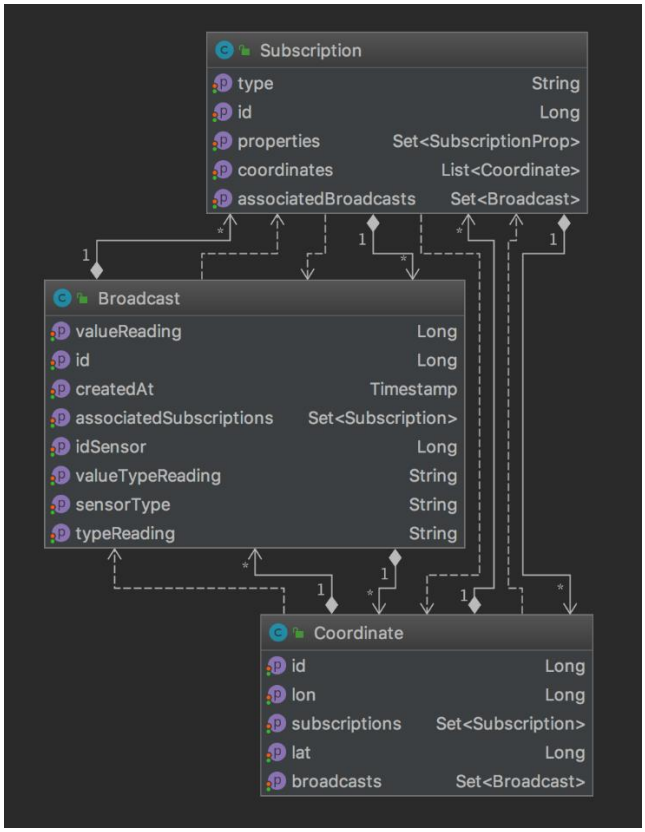
Slika 12. – Arhitektura sustava

4.1.1. Model baze podataka

U bazi podataka spremaju se pretplate, senzorske objave i geoprostorni podaci pa je relacijski model baze podataka poprilično jednostavan. Svaki zasebni model u tablici sadrži polje id koje predstavlja jedinstveni identifikacijski broj zapisa u toj tablici koji je ujedno i primarni ključ tog zapisa. Pretplate i senzorske objave sadrže geoprostorni podatak. Za objavu geoprostorni podatak može biti točka u prostoru ili poligon, dok za pretplatu može biti samo poligon.

Pretplata sadrži tip pretplate, svojstva pretplate koja predstavljaju atributne uvjete, geoprostorni podatak, te vezu na objave koje su bile obuhvaćene tom pretplatom.

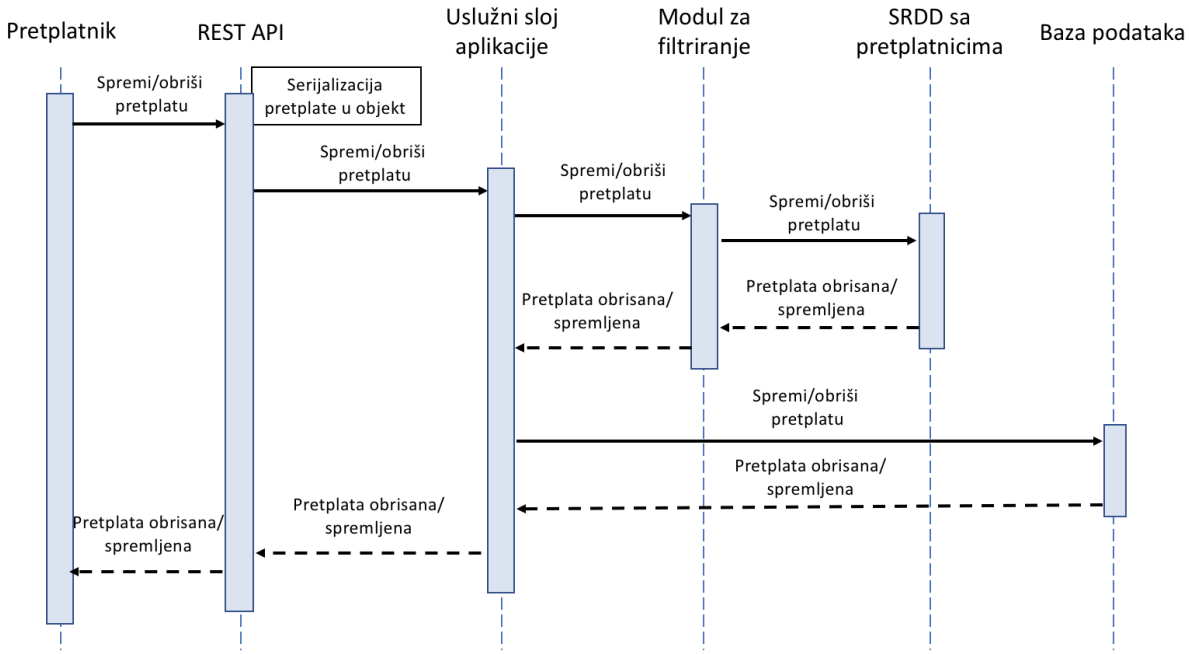
Objava sadrži vrijeme nastanka objave, identifikacijsku oznaku senzora, tip senzora, tip očitanja, vrijednost očitanja, mjernu jedinicu očitanja, te geoprostorni podatak. Tablica koordinata sadrži podatke o koordinatama koje se vežu na objave i pretplate, te im daju geoprostorni opis. Relacijski model baze podataka može se vidjeti na slici 3.



Slika 13. – Relacijski model baze podataka

4.1.2. Poslovna logika uslužnog sloja

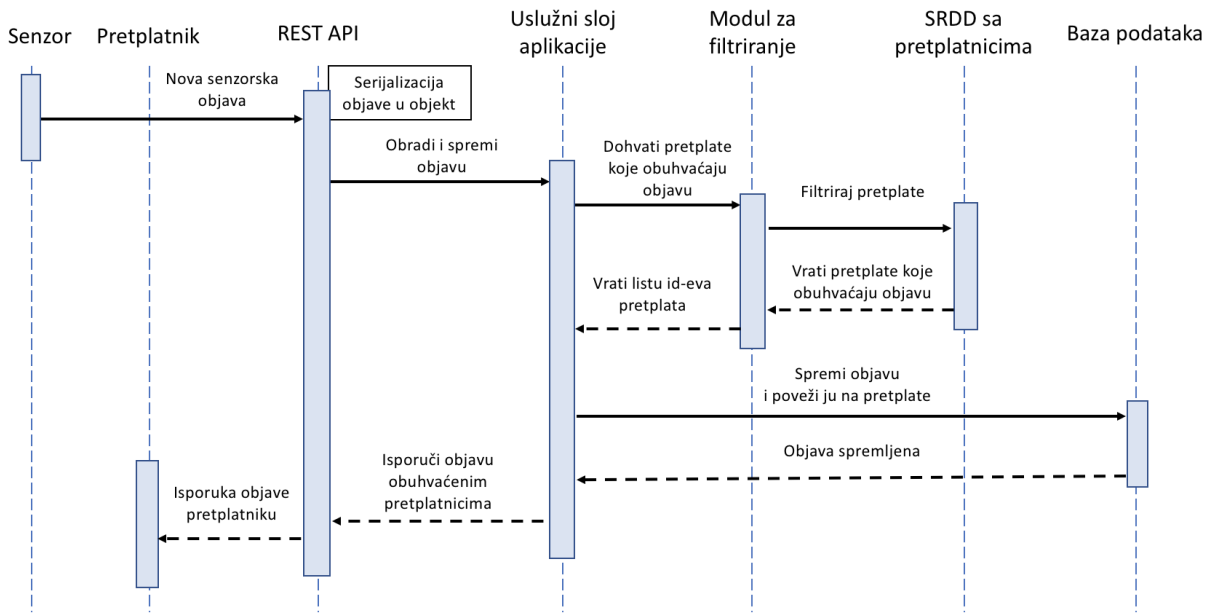
Dodavanje nove ili brisanje postojeće pretplate u sustavu odvija se sekvencijalno, dodavanjem ili brisanjem pretplate u bazi podataka i nakon toga iste operacija na raspodijeljenom podatkovnom sustavu. Nakon što se zaprimi zahtjev za dodavanje nove ili brisanje postojeće pretplate u sustavu, uslužni sloj aplikacije deserijalizira i obradi podatke, te ih pripremi za spremanje u bazu podataka i raspodijeljeni podatkovni sustav. Modul za filtriranje objava kao ulazne podatke prima pretplate koje sprema i raspoređuje u raspodijeljenom podatkovnom sustavu računalnog grozda. Nakon toga uslužni sloj aplikacije spremi pretplatu u bazu podataka. Tok operacija dodavanja nove pretplate i brisanja pretplate može se vidjeti na sekvencijskom dijagramu na slici 14.



Slika 14. – Sekvencijski dijagram dodavanja/brisanja pretplate

Dodavanje nove objave u sustavu odvija se filtriranjem objava po pretplatama, te zapisom objave u bazu podataka i vezanjem objave na pretplate koje obuhvaćaju

tu objavu. Modul za filtriranje objava kao ulazne podatke prima objave koje se na raspodijeljen način uspoređuju u geoprostornom atributnom prostoru sa svim pretplatama koje su zapisane u sustavu po njihovim definiranim uvjetima. Nakon određivanja pretplata koje obuhvaćaju objavu po geoprostornim atributima, dobivene pretplate se slijedno provjeravaju po uvjetima nad vrijednosnim atributnim prostorom. Nakon usporedbe objave i pretplata modul vraća identifikacijske oznake pretplata koje geoprostorno obuhvaćaju objavom ili se sa njom sijeku, te čiji uvjeti obuhvaćaju vrijednost atributa objave. Uslužni sloj pomoću liste identifikacijskih oznaka pretplata koje su obuhvaćene objavom dodjeljuje objavu pretplatama u bazi podataka. Tok operacije dodavanja nove objave u sustav može se vidjeti na sekvencijskom dijagramu na slici 15.



Slika 15. – Sekvencijski dijagram dodavanja nove objave u sustav

4.1.3. REST API

Aplikacija nudi četiri API rute:

- POST ruta za dodavanje nove objave u sustav koja prima podatke o objavi u JSON formatu po GeoJSON standardu
- POST ruta za dodavanje nove pretplate u sustav koja prima podatke o pretplati u JSON format po GeoJSON standardu

- DELETE ruta za brisanje postojeće pretplate u sustavu koja prima parameter id koji je ujedno i primarni ključ pretplate u tablici baze podataka
- GET ruta za dohvaćanje svih pretplata u sustavu
- GET ruta za dohvaćanje pojedinačne pretplate u sustavu koja prima parameter id koji je ujedno i primarni ključ pretplate u tablici baze podataka

4.2. Implementacija sustava

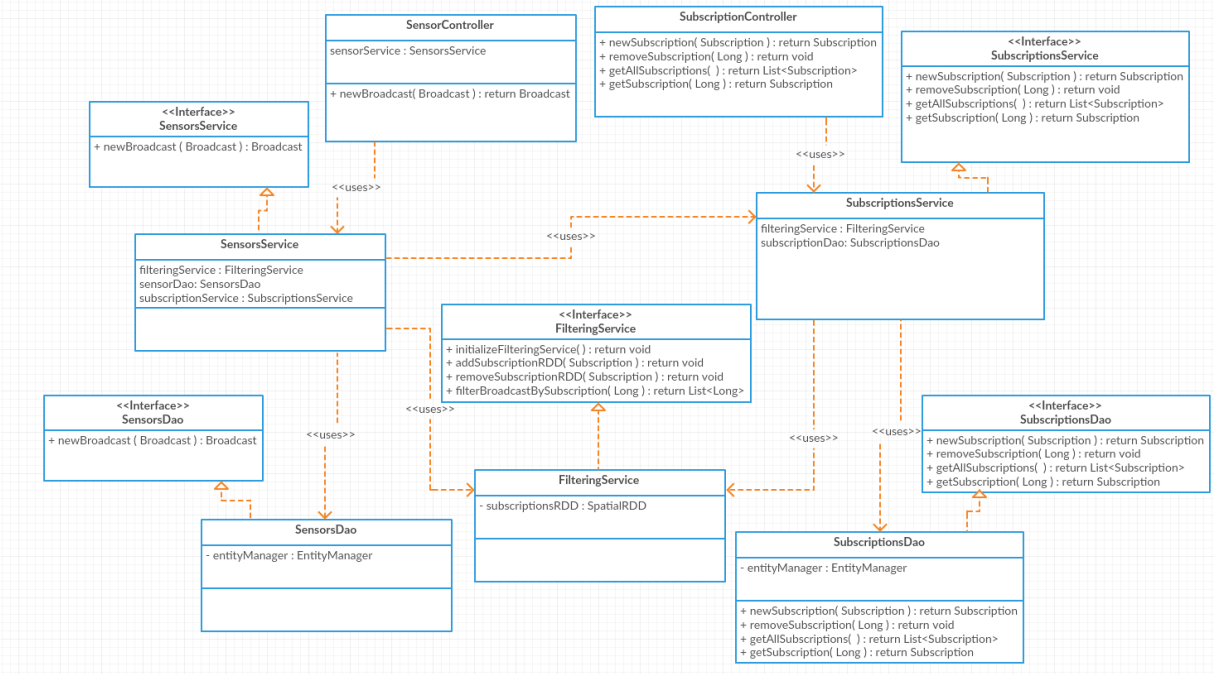
Aplikacija je izvedena u programskom jeziku Java, a kao REST usluga izvedena je modularno pomoću razvojnog okruženja Spring i tehnologije Hibernate za modeliranje i operacije nad bazom podataka. Za ugradnju programskih knjižnica u projekt korišten je Maven alat. Putem Maven alata, tj. posebne .xml datoteke koja se nalazi u korijenskom direktoriju projekta definiraju se ovisnosti o programskim knjižnicama, te njihove verzije koje su potrebne u projektu. Kao baza podataka korištena je u današnje vrijeme gotovo standardna MySQL baza podataka. Modul za raspodijeljeno filtriranje objava implementiran je pomoću Geospark programske knjiznice koja pruža apstraktno sučelje prema Geotools i Java Topology Suite programskim knjižnicama, te modulima Apache Spark programske knjižnice.

Korištenjem tehnologije Hibernate za modeliranje baze podataka, tablice su inicijalizirane implementacijom Java POJO klasa sa Hibernate anotacijama koje dodaju varijable unutar POJO klase kao atribute unutar baze podataka prilikom kompilacije projekta. Veza na bazu podataka definirana je u konfiguracijskim datotekama projekta putem IP adrese baze i porta koji nude konekciju na bazu podataka, te autorizacijskih podataka za ostvarenje konekcije. Kao ulaznu točku za sve operacije sa bazom podataka unutar poslovne logike implementirane u Java klasama projekta koristi se instanca klase EntityManager koja sadrži metode za interakciju sa bazom podataka. Upiti nad bazom podataka definiraju se u statičkim String varijablama i predaju se kroz metodu EntityManager klase createQuery(String query) gdje se kao parametar predaje upit. Upit se definira jezikom HQL (Hibernate Query Language).

Spring pruža ugrađene implementacije za laganu izradu REST API sučelja. Isto tako, razvojno okruženje Spring-a koristi koncept ubrizgavanja ovisnosti, te time uvelike olakšava slaganje klasa u agregacijske i kompozitne odnose preko njihovih sucelja.

4.2.1. Programska arhitektura sustava

Programska arhitektura sustava bazirana je na modelu kontroler-usluga-podatkovni pristup. Dijagram klasa cijele aplikacije se može vidjeti na slici 16.



Slika 16. - Dijagram klasa aplikacije

Dijagram klasa pokazuje sloj kontrolera u gornjem dijelu, uslužni sloj u srednjem, te sloj podatkovnog pristupa u donjem dijelu. Klase iz različitih slojeva međusobno komuniciraju pomoću sučelja u jednom smjeru, od sloja kontrolera prema sloju podatkovnog pristupa.

U sloju kontrolera implementirane su dvije klase od kojih se jedna brine za objave, a druga za pretplate. U ovim klasama sadržane su metode za pristup REST API sučelju. Implementirana je logika rada sa mrežnim sučeljem, serijalizacija JSON sadržaja REST poziva, te sadržaja sadržanog u URL-u pojedinih poziva, te su uspostavljene metode za rad sa REST pozivima, po jedna metoda za svaki REST API poziv. Većina prethodno nabrojanih funkcija kontrolera sadržana je u izvornoj implementaciji Spring razvojnog okruženja, te su funkcionalnosti ugrađene u metode putem Java anotacija koje ubacuju željene implementirane funkcionalnosti

u anotiranu metodu. Klasa SensorController sadrži jednu metodu newBroadcast(Broadcast) za dodavanje nove objave u sustav. Metoda serijalizira sadržaj REST poziva koji je u ovoj aplikaciji propisan GeoJSON standardom. Klasa SubscriptionController implementira metode newSubscription(SubscriptionPojo) koja prihvaća novu pretplatu, removeSubscription(Long) metodu za brisanje pretplate iz sustava koja prihvaća identifikacijsku oznaku pretplate koju se želi obrisati iz sustava, getSubscription(Long) metodu koja dohvaća pojedinu pretplatu zajedno sa njoj pridruženim objavama koja prihvaća identifikacijsku oznaku pretplate koja se dohvaća, te getAllSubscriptions() metodu koja dohvaća sve pretplate sa njima pripadajućim objavama u sustavu. Unutar svake od ovih metoda pozivaju se metode uslužnog sloja za obavljanje određenih poslova unutar aplikacije.

U uslužnom sloju nalaze se tri klase koje implementiraju poslovnu logiku sustava. Unutar svake od klasa postoje metode koje su zadužene za određenu operaciju. Klase usluga unutar svojih metoda pozivaju metode iz sloja podatkovnog pristupa ukoliko žele komunicirati sa bazom podataka, te metode iz drugih klasa usluga. Klasa SubscriptionsService implementira metode newSubscription(Subscription) za izradu nove pretplate, deleteSubscription(Subscription) za brisanje postojeće pretplate, getSubscription(Long) za dohvaćanje pretplate po identifikacijskom broju, te getAllSubscriptions() za dohvaćanje svih pretplata. Kod izrade nove pretplate ova klasa usluga poziva metodu iz FilteringService klase za dodavanje nove pretplate u SRDD. Isto tako, poziva se i metoda iz FilteringService klase za brisanje pretplate iz SRDD-a unutar metode za brisanje pretplate. Unutar klase SensorService i njezine jedine metode newBroadcast(Broadcast) poziva se metoda FilteringService klase za filtriranje objave nad pretplatama. Filtriranje se najprije vrši po geoprostornim atributima, a zatim se dobivene pretplate slijedno prolaze i provjeravaju se vrijednosni atribut objave sa uvjetima svake pretplate. Nakon provedenog filtriranja informacije o pronađenim pretplatama koje uvjetima obuhvaćaju objave vraćaju se u obliku njihovih identifikacijskih oznaka. Sa tim informacijama objava se zapisuje u bazu podataka, te se vežu na vraćene pretplate.

Sloj podatkovnog pristupa definira dvije klase, SubscriptionDao i SensorDao. Ove klase sadrže metode koje pružaju izravnu vezu na bazu podataka, te pružaju usluge interakcije sa bazom podataka putem instance EntityManager klase uslužnom sloju.

5. Eksperimentalna evaluacija

Predmet ovog rada je izrada sustava za filtriranje objava na temelju geoprostornih značajki. Kod takvog sustava ključni dio jest komponenta koja provodi filtriranje objava na temelju postavljenih geoprostornih uvjeta. Geoprostorna relacija koja se implementira zahtjeva da ukoliko je uvjet zadovoljen, da se geoprostorni objekt preklapa sa poligonom uvjeta. Kako bi se dobila optimalna implementacija sustava, tj. optimalna implementacija komponente koja filtrira objave, provodi se eksperimentalna evaluacija različitih karakteristika filtera koji je implementiran pomoću Geospark programske knjižnice. Ispituju se mogućnosti koje Geospark nudi, a to su prostorno particioniranje, te geoprostorno indeksiranje. Pretpostavka od koje se polazi prije eksperimenta jest da su geoprostorne značajke objava ili pretplata sa kojima sustav radi stohastičke. Drugim riječima, poligoni ili točke objava ili pretplata mogu poprimiti bilo kakve oblike na određenom prostoru.

5.1. Opis eksperimenta

Eksperiment se provodi nad objavama i pretplatama koje su generirane nasumičnim odabirom koordinata rubnih točaka. Svaki poligon dobiven je od nasumično odabranog broja koordinata u intervalu od 10 do 50. Ovakav odabir koordinata je izabran jer se kreće od pretpostavke da sustav može zaprimiti različite vrste geoprostornih objekata objava ili pretplata bez nekog predodređenog pravila. Koordinate geoprostornih objekata odabrane su na način da sve koordinate pripadaju području omeđenom poligonom u obliku pravokutnika oko granica Velike Britanije. Vrste geometrija korištene za objave su točke i poligoni, dok su za pretplate generirani geoprostorni objekti čija se geometrija sastoji od isključivo poligona. Eksperiment se provodi uspoređujući isključivo geoprostorne

značajke objava i pretplata. Nadalje, provodi se usporedba utjecaja upotrebe geoprostornih indeksa nad geoprostornim objektima objava i pretplata, te utjecaj prostornog particioniranja podataka na SRDD-ovima između odvojenih datotečnih sustava unutar grozda. Metode prostornog particioniranja koje su evaluirane su: QuadTree, EqualGrid, te RTree. Izvedbe geoprostornih indeksa nad pojedinim SRDD-ovima koje su evaluirane su: RTree, te QuadTree. Broj pretplata se povećava kako bi se procijenio utjecaj porasta pretplata na vrijeme pretraživanja, te trajanje particioniranja i indeksiranja. Na taj način dobiveni su rezultati za brojeve pretplata: 1000, 5000, 10000, 50000, 100000, 500000. Broj objava koje su u eksperimentu slijedno dolazile na filtriranje iznosi 10000. Za procjenu vremena potrebnog za filtriranje radi se prosjek trajanja filtriranja jedne objave. Evaluacija je provedena na grozdu računala u cloud-u u sklopu Digital Ocean platforme. Grozd se sastoji od jednog glavnog računala i tri pomoćna računala (karakteristike računala - vCPU Intel Xeon E5-2650L v3 1.80GHz, 1GB RAM, 25GB SSD)

5.2 Konfiguracija okoline i izvođenje eksperimenta na računalnom grozdu

Za izvođenje eksperimenta konfiguriran je računalni grozd na platformi Digital Ocean koji se sastoji od četiri računala - jedno glavno i tri pomoćna računala. Podizanje računalnog grozda na ovaj način postiže se u nekoliko koraka:

1. Podizanje instanci pojedinih servera - Na Digital Ocean sučelju unajmljuju se četiri servera sa odabirom njihovih performansi u istoj regiji. Nakon najma serveri se instanciraju i pokreću, te se putem mail-a dobiju IP adrese i podaci za SSH pristup tim serverima
2. Početna konfiguracija - Svaki pojedini server se konfigurira tako da se inicijaliziraju repozitoriji za instalaciju potrebnih alata naredbom apt-get update install, te se nakon toga instaliraju potrebni alati: Java, Spark i Scala
3. Mrežna konfiguracija - Na svakom pojedinom serveru ažurira se datoteka hosts sa podacima o IP adresama drugih servera. Konfigurira se SSH konfiguracija za komunikaciju među serverima u grozdu bez potrebne

autorizacije izradom SSH ključa koji se pohranjuje unutar .ssh/authorized_hosts datoteke na svakom serveru.

4. Konfiguracija Spark-a - Ažuriraju se datoteke spark-env.sh, te slaves unutar Spark-ovih datoteka sa podacima o lokalnoj verziji Jave, te IP adresama servera unutar grozda.
5. Pokretanje Spark-a - Spark se pokrece naredbom ./sbin/start-all.sh. Nakon pokretanja Spark-a sustav je spreman za zaprimanje Spark poslova

Nakon sto je Spark pokrenut, na glavno računalo prebacuje se .jar datoteka čije izvođenje testira komponentu filtera. Program se pokreće naredbom spark-submit kojoj se kao parametri predaju .jar datoteka, IP adresa glavnog računala, parametri koji opisuju način korištenja Spark-a, te ulazna datoteka unutar .jar datoteke.

5.3 Rezultati eksperimenta

U prvom dijelu eksperimenta analizira se utjecaj korištenja različitih metoda prostornog particioniranja pretplata među SRDD-ovima na vrijeme izvođenja jedne operacije filtriranja pretplata koje obuhvaćaju objavu. Vremena izvođenja operacije jednog filtriranja pretplata po objavi korištenjem različitih metoda prostornog particioniranja pretplata medu SRDD-ovima u ovisnosti o broju pohranjenih pretplata u SRDD-ovima može se vidjeti u tablici 2. Prostorno particioniranje SRDD-ova donosi ubrzanje vremena izvođenja operacije filtriranja za pojedine metode. Ipak, to ubrzanje nije drastično. Naime, pretplate su generirane nasumično, i veći udio pretplata kod takvog nasumičnog generiranja čine geoprostorni objekti čije rubne točke se nalaze diljem šireg prostora Velike Britanije. Pretpostavka je da zbog toga među SRDD-ovima dolazi do velikog preklapanja istih pretplata koje su particionirane i replicirane, te se gubi efekt optimizacije. Drugim riječima, u operaciji filtriranja sudjeluju u većini slučajeva i dalje sva računala u grozdu. Moglo bi se prema tome pretpostaviti da efekt prostornog particioniranja dolazi do većeg izražaja ukoliko se geoprostorni objekti po kojima se filtrira nalaze površinski na manjim geoprostornim područjima. U okviru ovog rada nije rađen eksperiment nad podacima koji su podređeni pravilima

i generirani po nekom zadanom modelu u smislu da koordinate nisu raspršene po cijelom prostoru. Taj bi se slučaj svakako mogao isto razmotriti, te bi se time ove pretpostavke mogle i dokazati.

No ipak, zaključak u smislu postizanja optimalne izvedbe sustava za pretpostavke eksperimenta jest da QuadTree prostorno particioniranje pretplata donosi malo, ali dovoljno poboljšanje u smislu skalabilnosti unutar ovog eksperimenta sa zadanim pretpostavkama kod porasta broja pretplata. Sa porastom broja pretplata za QuadTree prostorno particioniranje sustav ima kontinuirano najkraće vrijeme izvođenja operacije filtriranja. Stoga se za implementaciju sustava preporuča koristiti ovu metodu prostornog particioniranja.

	1000 pretplata	5000 pretplata	10000 pretplata	50000 pretplata	100000 pretplata
Bez prostornog particioniranja	207ms	244ms	472ms	1939ms	3579ms
QuadTree	202ms	291ms	486ms	1781ms	3081ms
RTree	120ms	324ms	628ms	2120ms	3880ms
EqualGrid	122ms	328ms	586ms	2054ms	3543ms

Tablica 2. - Vremena izvođenja za broj pretplata ovisno u upotrebi metode prostornog particioniranja pretplata

U drugom dijelu eksperimenta analizira se utjecaj korištenja različitih metoda geoprostornog indeksiranja nad SRDD-ovima. Vremena izvođenja operacije jednog filtriranja pretplata po objavi korištenjem različitih metoda geoprostornog indeksiranja pretplata u ovisnosti o broju pohranjenih pretplata u SRDD-ovima

može se vidjeti u tablici 3. Korištenje geoprostornih indeksa donosi malo lošije performanse od implementacije koja ne koristi geoprostorne indekse. Pretpostavka je također da su rezultati takvi zbog nasumično odabranih koordinata geoprostornih objekata pretplata. No isto tako, moguće je i da za ovu vrstu upita geoprostorno indeksiranje pretplata ne optimizira vrijeme izvođenja filtriranja. U svakom slučaju, prema rezultatima ovog dijela eksperimenta za sustav razvijen u okviru ovog rada preporučilo bi se ne koristiti geoprostorni indeks.

	1000 pretplata	5000 pretplata	10000 pretplata	50000 pretplata	100000 pretplata
Bez indeksa	207ms	244ms	472ms	1939ms	3579ms
RTree	380ms	513ms	612ms	2159ms	3603ms
QuadTree	397ms	668ms	783ms	2054ms	3658ms

Tablica 3. - Vremena izvođenja za broj pretplata ovisno u upotrebi metode geoprostornog indeksiranja pretplata

Daljnjim eksperimentiranjem kombiniranja prostornog particioniranja pretplata sa geoprostornim indeksiranjem dolazi se do zaključka da se kombinacijom prostornog particioniranja pretplata, sa geoprostornim indeksiranjem podataka u SRDD-ovima dolazi do optimalne izvedbe. Odabirom RTree geoprostornog indeksa koji se pokazao boljim od QuadTree indeksa, no ipak lošijim od izvedbe filtera bez geoprostornog indeksa, te kombiniranjem sa metodom QuadTree prostornog particioniranja pretplata koja se pokazalo najboljom u usporedbi metoda prostornog particioniranja dolazi se do optimalnih rezultata. Vremena izvođenja operacije jednog filtriranja pretplata po objavi za slučajeve filtriranja kod kojeg se ne koristi particioniranje i indeksiranje, filtriranja sa prostorno particioniranim SRDD-ovima metodom QuadTree, te filtriranja sa prostorno

particioniranim i geoprostorno indeksiranim SRDD-ovima, mogu se vidjeti u tablici 4.

	1000 pretplata	5000 pretplata	10000 pretplata	50000 pretplata	100000 pretplata
Bez indeksa i prostornog particioniranja	207ms	244ms	472ms	1939ms	3579ms
QuadTree prostorno particioniranje	202ms	291ms	486ms	1781ms	3081ms
QuadTree prostorno particioniranje u kombinaciji sa RTree indeksom	224ms	274ms	430ms	1682ms	3054ms

Tablica 4. - Usporedba vremena izvođenja za kombinirani slučaj QuadTree prostornog particioniranja i RTree geoprostornog indeksiranja pretplata

Zaključak je da se filtar uz pretpostavku rada sa nasumično odabranim područjima pretplata bez definiranih ograničenja implementira na način da se SRDD-ovi sa pretplatama prostorno particioniraju po QuadTree indeksu, te da se pojedini SRDD-ovi zatim geoprostorno indeksiraju RTree indeksom.

Zaključak

Rad sa geoprostornim objektima u raspodijeljenoj okolini uvelike je olakšan sa postojećim programskim knjižnicama koje olakšavaju implementiranje komponenata takvog sustava za izvedbu operacija nad geoprostornim podacima. Zaključak je da trenutno u zajednici otvorenog koda postoji široka paleta gotovih rješenja za rad sa geoprostornim podacima. Posto je rad sa geoprostornim podacima u većini slučajeva kompleksan, široki izbor alata i knjižnica pruža ohrabrenje za implementaciju sustava za rad sa takvim podacima koji su skalabilni i optimalni u svojemu radu. Primjer knjižnice koja omogućava rad sa geoprostornim podacima u raspodijeljenoj okolini preko jednostavnog apstraktnog API-ja je Geospark programska knjižnica. Pomoću nje u okviru ovog rada razvijena je komponenta za filtriranje geoprostornih objekata na temelju geoprostornih uvjeta. Komponenta je uklopljena u sustav koji implementira poslovnu logiku i bazu podataka za rad sa senzorskim objavama i pretplatama. Razvijeni sustav je REST API usluga koja se primjerice lagano može uklopiti kao mikrousluga u neki postojeći sustav te pružati uslugu obrade geoprostornih informacija na temelju modela objavi-pretplati.

Provedeno testiranje komponente za filtriranje provedeno je nad nasumično odabranim podacima čije su rubne točke u geoprostoru nasumično raspršene. Eksperiment nije pokazao drastična poboljšanja za pojedine vrste particioniranja i geoprostornog indeksiranja. Ipak, uzimajući u obzir male razlike između pojedinih rezultata pokazalo se da je optimalna izvedba filtera za sustava koji je razvijen u okviru ovog rada, sa pretpostavkom podataka sa stohastičkim geoprostornim značajkama, ona izvedba koja prostorno particionira SRDD-ove metodom QuadTree, te koristi RTree geoprostorni indeks. Ova izvedba donosi bolje performanse kod porasta broja pretplata što čini sustav malo skalabilnijim. Eksperiment je proveden samo na jednoj vrsti podataka: nasumično odabranim geoprostornim objektima na području Velike Britanije. Pretpostavka je da različite vrste podataka donose sa sobom drugačije rezultate ukoliko se koriste druge metode prostornog particioniranja ili geoprostornog indeksiranja nad tim podacima. Isto tako, u eksperimentu je provedeno testiranje na upitu prekrivanja ili

presijecanja geoprostornih objekata. Moguće je da različite vrste geoprostornog indeksiranja ili particioniranja donose poboljšanja kod izvedbe drugih vrsta upita nad geoprostornim objektima. Kod izrade sustava koji implementira druge vrste upita, ili radi upite nad geoprostornim objektima koji ima određenija svojstva preporučljivo bi bilo ponovno provesti eksperiment i donijeti nove zaključke.

Raspodijeljeni sustav će svakako imati bolje performanse od sustava filtera koji je implementiran na jednom računalu. Glavno poboljšanje se nalazi u činjenici da je raspodijeljeni sustav skalabilan u smislu da se dodavanjem još jednog računala u grozd mogu pojačati performanse sustava jer se dodavanjem još jednog računala povećava kapacitet za paralelnom obradom operacija. Time se postiže horizontalno skaliranje, dok je kod monolitne arhitekture sa jednim računalom koje primjerice koristi vise dretvi za postizanje paralelizma to nemoguće postići jer zasebno računalo ima ograničene performanse koje nisu proširive.

Literatura

[1] Jia Yu; Zongsi Zhang; Mohamed Sarwat, Spatial data management in apache spark: the GeoSpark perspective and beyond, lipanj 2017.

[2] Borna Radotic, Geoprostorni sustav objavi-pretplati za ucinkovito filtriranje toka senzorskih podataka, lipanj 2018.

[3] Shengti Pan, The Performance Comparison of Hadoop and Spark, ozujak 2016.

[4] Guttman, A. R-Trees: A Dynamic Index Structure for Spatial Searching. SIGMOD'84, Proceedings of Annual Meeting, Boston, 1984, pp. 47-57.

[5] Apache Spark, https://en.wikipedia.org/wiki/Apache_Spark, sijecanj 2019.

[6] Geospark, Tutorial, http://geospark.datasyslab.org/, sijecanj 2018.

[7] Tom MacWright, More than you ever wanted to know about GeoJSON, https://macwright.org/2015/03/23/geojson-second-bite.html, ozujak 2015.

[8] Christian Bauer; Gavin King; Gary Gregory, Java Persistence with Hibernate, Second Edition, 2006.

Sažetak

Raspodijeljeno filtriranje tokova senzorskih podataka u računalnom grozdu

Cilj ovog rada je razmatranje predloženog rješenja za implementaciju sustava koji služi za filtriranje senzorskih objava na temelju pretplata. Predlaže se rješenje koje se temelji na paralelnom obavljanju operacija usporedbe uvjeta pretplate sa značajkama objava na raspodijeljenom sustavu, točnije računalnom grozdu. Cilj ovog rješenja je poboljšati svojstvo skalabilnosti sustava omogućavanjem paralelnog izvođenja operacija filtriranja. Objave i pretplate se generiraju kao geoprostorni objekti zapisani u GeoJSON formatu koji sadrže geoprostorne i vrijednosne atribute. U razmatranju se naglasak stavlja na geoprostorne značajke objava i uvjete pretplata, te način na koji se one međusobno uspoređuju tijekom operacije filtriranja. Razmatraju se programske knjižnice koje olakšavaju implementaciju sustava za raspodijeljeno uspoređivanje geoprostornih objekata po njihovim relacijskim odnosima u prostoru, te koje omogućavaju implementaciju raspodijeljenog geoprostornog indeksa. Provodi se eksperimentalna evaluacija predloženog rješenja. Rezultat ovog rada je sustav koji omogućuje rad sa objavama i pretplatama, implementira zadane funkcionalnosti filtriranja, te ih nudi putem REST API usluge.

Kljucne rijeci: geoprostorni podaci, filtriranje, GeoJSON, raspodijeljeni sustav, raspodijeljena obrada podataka, geoprostorni indeks, objavi-pretplati

Summary

Distributed filtering of sensor data in a cluster

The goal of this thesis is a review of a proposed solution to implement a system which filters sensor broadcast base on subscription stored in a system. Proposed solution involves parallelizing operations of comparing subscription conditions with broadcast values on a computer cluster. The goal of proposed solution is to implement better scalability into the system by introducing parallel operations. Subscriptions and broadcasts are generated as geospatial objects which are stored in GeoJSON format which contains geospatial and value attributes. In the review special attention is put upon the way how the system should compare geospatial properties of broadcasts and subscriptions during filtering operations. Software libraries are introduced and reviewed which enable easier implementation of distributed systems which are working with geospatial data, evaluating the geospatial objects depending on their mutual geospatial relations, and implementing out of the box solutions of geospatial indexing of the data. Experimental evaluation is performed upon the devised solution. The result of this thesis is a system which works with subscriptions and broadcasts, implements filtering functionalities and offers them through REST API service.

Keywords: geospatial data, filtering, GeoJSON, distributed system, distributed data, geospatial indexing, publish-subscribe