SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1981

FILTRIRANJE GEOPROSTORNOG TOKA PODATAKA KORIŠTENJEM PLATFORME APACHE FLINK

Andrea Perica

Zagreb, lipanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1981

FILTRIRANJE GEOPROSTORNOG TOKA PODATAKA KORIŠTENJEM PLATFORME APACHE FLINK

Andrea Perica

Zagreb, lipanj 2019.

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

Zagreb, 8. ožujka 2019.

DIPLOMSKI ZADATAK br. 1981

Pristupnik: **Andrea Perica (0036483924)**
Studij: Računarstvo
Profil: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Filtriranje geoprostornog toka podataka korištenjem platforme Apache Flink**

Opis zadatka:

Vaš zadatak je osmisliti, izvesti u programskom jeziku Java te testirati aplikaciju za računalni grozd za filtriranje geoprostornog toka podataka. Elementi toka podataka i trajni upiti sadrže geoprostorni objekt (npr. točku, liniju, poligon). U toku podataka se nalaze elementi koji se sastoje od geoprostornog objekta koji predstavlja lokaciju te dodatnog sadržaja. Trajni upit se sastoji od geoprostornog objekta koji predstavlja geografsko područje interesa te dodatnih uvjeta na sadržaj elementa na osnovu kojih se vrši kontinuirana usporedba s nadolazećim elementima toka podataka.
Prilikom izrade sustava iskoristite Javine programske knjižnice JTS (Java Topology Suite), GeoTools i GeoSpark. Istražite geoprostorne indekse koji su podržani u navedenim Javinim programskim knjižnicama, analizirajte njihove karakteristike i performanse te implementirajte raspodijeljeni indeks u računalnom grozdu korištenjem platforme Apache Flink.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 28. lipnja 2019.

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

Djelovođa:
Izv. prof. dr. sc. Boris Milašinović

Predsjednik odbora za diplomski rad profila:
Izv. prof. dr. sc. Igor Mekterović

Sadržaj

Uvod

Diljem Zemlje svakodnevno se prikupljaju velike količine podataka koje donose informacije o vrijednostima mjerenih parametara te o lokaciji na kojoj je mjerenje provedeno. Prate se trenutni vremenski uvjeti, brzine vjetrova, visine valova, vodostaji rijeka, lavine, te ostale pojave kao što su primjerice prometne gužve. Zbog poznatog položaja na Zemlji, navedeni podaci nazivaju se geoprostornima. Geoprostorni podaci mogu biti predstavljeni u obliku točke, linije, poligona ili terena [1] te takvi pohranjeni u memoriju računala. Budući da su operacije nad navedenim podacima računalu zahtjevne, bitno ih je smisleno organizirati u određene strukture ovisno o načinu zapisa geoprostornih komponenti, njihovom položaju na karti svijeta te operacijama koje će se nad njima provoditi.

Nadalje, osnova za mogućnost rada s velikim količinama podataka je raspodijeljena obrada koja zahtijeva složenu i robusnu programsku pozadinu. Danas u tu svrhu postoje platforme koje korisniku omogućuju koncentraciju na rješavanje samog problema bez potrebe za implementacijom pozadinskih mehanizama kao što su primjerice raspodjela poslova ili oporavak nakon ispada čvora. Neke od njih su: Apache Hadoop, Apache Spark, Apache Flink, Apache Storm. Svaka takva platforma ima posebnosti u svojoj izvedbi te je stoga potrebno analizirati problem i odabrati onu platformu koja nudi funkcionalnosti primjerene za izgradnju rješenja. [2]

Ovaj rad bavi se raspodijeljenom obradom velike količine geoprostornih podataka. Pritom proučava mogućnosti koje nude knjižnice za rad s geoprostornim podacima, njihovu prostornu raspodjelu te pohranu u prostorne indekse. Kao potpora za raspodijeljenu obradu koristi se platforma Apache Flink. Ujedinjenjem navedenih tehnologija razvijena je aplikacija za filtriranje geoprostornog toka podataka. Filtriranje je izvedeno u sustavu objava i pretplata u kojem su objave točke na Zemljinoj površini, a pretplate poligoni koji omeđuju područje interesa korisnika.

Aplikacija je iskorištena za provedbu eksperimenta kojem je cilj proučiti kako odabir strukture za organizaciju geoprostornih podataka utječe na brzinu izvođenja programa. Prilikom mjerenja korišteni su i dodatni parametri koji omogućuju detaljnije vrednovanje rezultata.

Velika zahvala mentoru, izv. prof. dr. sc. Krešimiru Pripužiću, na pomoći, vodstvu i omogućenim resursima prilikom izrade ovog rada.

1. Platforma Apache Flink

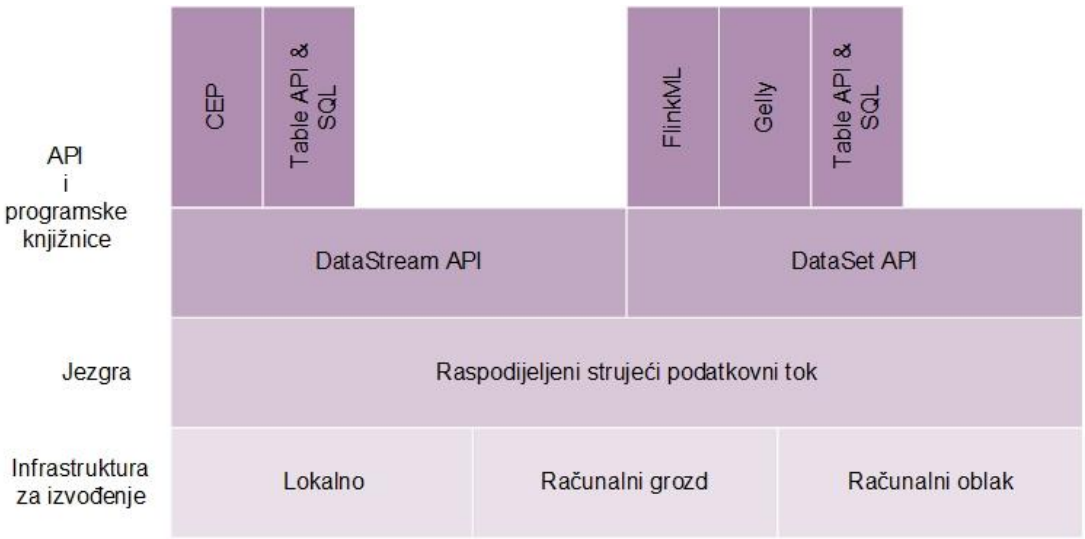
Apache Flink je platforma za raspodijeljenu obradu uz održavanje stanja nad ograničenim i neograničenim tokovima podataka. Flink je izveden na način koji podržava rad u svim često korištenim okruženjima računalnog grozda, precizno vrši obradu podataka uz visoku propusnost i malo vrijeme kašnjenja [3] [4].

U nastavku su objašnjeni određeni koncepti Flink platforme, s naglaskom na one koji su pomogli pri izradi programskog rješenja razvijenog u sklopu ovog rada.

1.1. Arhitektura

1.1.1. Komponentni složaj

Flink je sustav sastavljen od slojeva od kojih svaki predstavlja drugačiju razinu apstrakcije. Slika 1 prikazuje složaj komponenti.



Slika 1: Komponentni složaj **[5]**

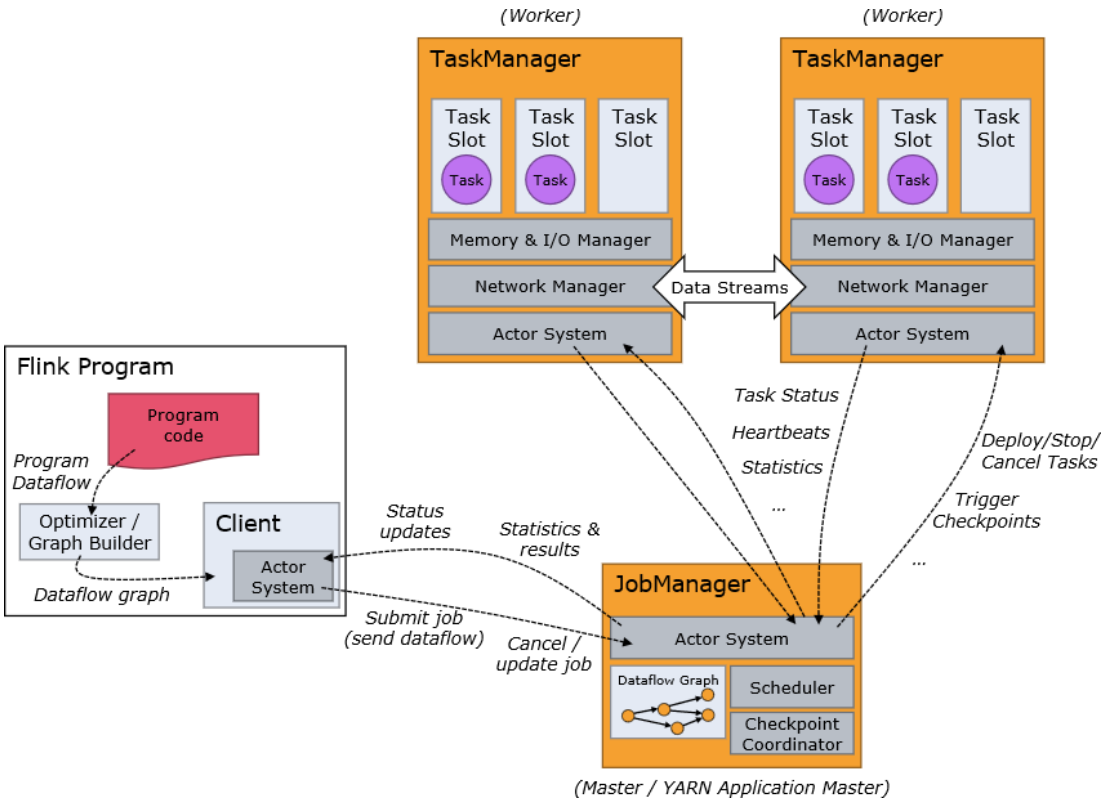
U sloju jezgre (*Core*) pomoću procesa *Job Manager* i *Task Manager* organizira se raspodijeljena obrada. Program se prima u obliku koji se naziva *JobGraph*, neparalelnoj reprezentaciji aplikacije.

Na razini iznad nalaze se dva osnovna sučelja za programiranje aplikacija (API), *DataStream* i *DataSet*, s pripadajućim knjižnicama. Oni generiraju *JobGraph*, svaki koristeći kompilacijske procese specifične za svoj način obrade podataka.

Poslovi se mogu izvoditi u lokalnom okruženju, na računalnom grozdu ili u oblaku (infrastruktura za izvođenje - *Deploy*).

1.1.2. Izvođenje u raspodijeljenoj okolini

Prilikom izvršavanja Flinkovih aplikacija koriste se dvije vrste procesa.



Slika 2: Procesi **[6]**

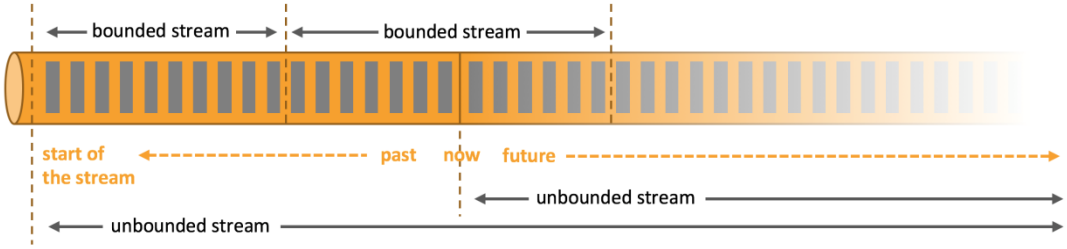
Job Manager je glavni proces koji usklađuje raspodijeljeno izvođenje zakazujući zadatke (*tasks*) te koordinirajući razne mehanizme kao što su

održavanje kontrolnih točaka i oporavak nakon kvara. Osnovna postava okruženja mora sadržavati barem jedan takav proces, no moguće ih je pokrenuti i više kako bi se postigla bolja dostupnost sustava. Više pokrenutih *Job Managera* se organizira na način da je jedan glavni, dok su ostali pričuvni. *Task Manager* je proces radnik i on izvršava podzadatke, elemente tokova podataka sprema u međuspremnik i razmjenjuje s drugim *Task Managerima.* U sustavu mora biti barem jedan proces radnik. [6]

Na slici 2 vidljiv je međusobni položaj dvaju opisanih procesa te klijenta (*Client*). Klijent priprema tokove podataka, šalje ih *Job Manageru* i pokreće izvršavanje programa. Nakon navedenih koraka moguće ga je isključiti iz sustava te kao takav nije dio okoline izvršavanja Flink programa.

1.2. DataStream i DataSet

Podaci koje Flink prima uglavnom nastaju u obliku toka događaja. Primjerice, tijekom mjerenja senzora temperature zraka, interakcije na web stranici ili bankovne transakcije. Neograničen tok (*unbounded stream*) ima definiran početak, ali nedefiniran kraj. Obrada podataka vrši se kontinuirano, redoslijedom kojim podaci pristižu u tok. Zbog toga je često potrebno paziti kojim se redom podaci šalju na izvor toka. Ograničen tok (*bounded stream*) ima definiran i početak i kraj. U slučaju ograničenog toka, moguće je čekati da podaci pristignu u potpunosti, te tek tada pokrenuti obradu toka fiksne veličine. Flink koristi posebne algoritme i strukture podataka za obradu toka točno određene i nepromjenjive veličine za dobivanje što boljeg rezultata.. Slika 3 vizualizira ograničen i neograničen tok.



Slika 3: Ograničen i neograničen tok **[3]**

Apache Flink slijedi filozofiju ujedinjenog pristupa obradi skupnih (*batch*) i strujećih (*streaming*) podataka. Skupna obrada promatra se kao poseban slučaj strujeće, gdje se obrađuje tok koji ipak u jednom trenutku dođe do svog kraja. Flinkov primarni rad je sa strujećim podacima, ali uz to nudi i dovoljan broj mogućnosti za rad nad skupnim podacima. Cilj korištenja ovakvog jedinstvenog pristupa jest postizanje visokih performansi u oba okruženja izvođenja programa. [7]

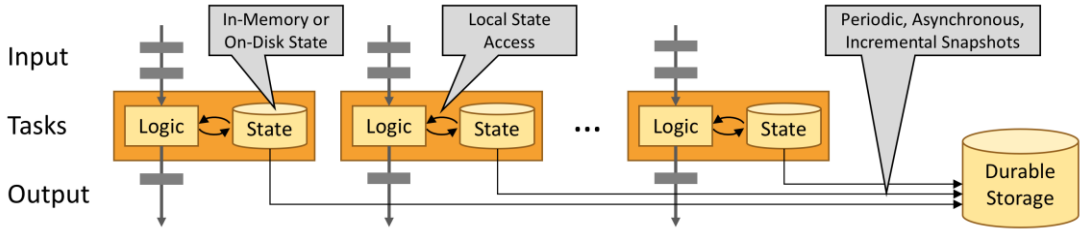
Flink koristi dva različita razreda za reprezentaciju podataka: `DataStream` i `DataSet`. Moguće ih je opisati kao nepromjenjive kolekcije podataka koje smiju sadržavati duplikate. Nepromjenjive su u smislu da nastaju iz zadanog izvora podataka te nije moguće dodati podatke iz drugog izvora ili ih ukloniti. Prvi se razred teoretski može sastojati od neograničene količine podataka te se nad njim primarno provodi strujeća obrada, dok je u slučaju potonjeg broj podataka konačan i oni se obrađuju skupno. [7]

Prilikom izrade programskog rješenja korišteno je *DataStream* sučelje za programiranje aplikacija, odnosno API, stoga se teoretska objašnjenja u nastavku odnose na navedeni aspekt rada platforme Flink.

1.3. DataStream API

1.3.1. Rad sa stanjem

U uvodu rada spomenuto je da Flink omogućuje rad sa stanjem. Stanja su zapravo podaci koji se održavaju uz dijelove toka gdje su potrebni pri nekom izračunu. Stanje se čuva u radnoj memoriji, a ukoliko njegova veličina prekorači dostupnu memoriju, pohrana se djelomično prebacuje na disk.



Slika 4: Pristup stanju **[3]**

Slika 4 prikazuje kako se položaj određenog stanja u sustavu nalazi odmah uz logiku koja se provodi nad ciljanim dijelom toka. Uz to, vidljiv je i mehanizam tolerancije kvarova. Strelice koje smjeraju od stanja (na slici „State“) prema mjestu trajne pohrane (na slici „Durable Storage“) označavaju periodičko, asinkrono pospremanje stanja u kontrolne točke. Izradom sigurnosnih kopija (*checkpointing*) Flink omogućuje povratak u konzistentno stanje i pritom garantira da se neće pojaviti duplikati podataka i da neće biti izostavljenih podataka [3] [8].

Postoje dvije osnovne vrste stanja koje se koriste pri izradi aplikacija u Flinku: *Keyed State* i *Operator State*. *Keyed State* stanje vezano je uz tokove razdijeljene po ključu (`KeyedStream`) i može se koristiti isključivo u funkcijama i operatorima koji djeluju nad takvim tokom. S druge strane, *Operator State* je stanje vezano uz pojedinu paralelnu instancu operatora. Jednostavnije, *Keyed State* se opisuje kao *Operator State* koje je razdijeljeno na određen način i svakom ključu toka je dodijeljen jedan njegov dio. [9]

1.3.2. Operatori

Operatori su funkcije koje pretvaraju jedan ili više tokova (`DataStream`) u novi `DataStream`. Preoblikovanja se nad tokom provode od njegovog izvora (`Source`) do ponora (`Sink`), a moguće je prepoznati više različitih tipova. Transformacije primjerice mogu mijenjati tip elemenata u toku i pritom na izlazu davati jednak ili različit broj elemenata, sakupljati elemente u jedan provođenjem neke matematičke operacije ili razdjeljivati tok na temelju određenog ključa. Primjeri transformacija [10]:

- Pretvaranje toka jednog tipa u tok drugog tipa na način da iz svakog ulaznog elementa nastaje točno jedan ili različit broj izlaznih (`DataStream<A> -> DataStream<B>`) : `map`, `flatMap`

- Filtriranje kojim se za svaki element izvede `boolean` funkcija te se na izlaz puštaju samo oni elementi za koje funkcija vraća istinu (`DataStream<A> -> DataStream<A>`) : `filter`

- Podjela toka na više paralelnih tako da svi elementi s istim ključem pripadnu u istu paralelnu instancu (`DataStream -> KeyedStream`) : `keyBy`

- Kumulativno sjedinjivanje nad tokom razdijeljenim po ključu (`KeyedStream -> DataStream`) : `sum`, `min`, `max`

- Ujedinjenje dvaju ili više tokova istog tipa u jedan koji sadrži sve elemente svih tokova koju se ujedinjuju (`DataStream<A>* -> DataStream<A>`) : `union`

- Povezivanje dvaju tokova u jedan uz zadržavanje tipova njihovih elemenata (`DataStream<A>, DataStream<B> -> ConnectedStreams<A,B>`) : `connect`

- Raspodjela elemenata toka u particije po uniformnoj razdiobi (`DataStream -> DataStream`) : `shuffle`

- Podjela elemenata toka u particije na niskoj razini; vlastita implementacija (`DataStream -> DataStream`) : `partitionCustom`

Valja posebno istaknuti ProcessFunction. To je operacija za obradu toka na niskoj razini. Može se opisati kao `FlatMapFunction` funkcija koja daje dodatni pristup elementima toka (*events*), stanju (*state*) i mjeračima vremena (*timers*) [11]. U slučaju rada s tokom koji je podijeljen po ključu, moguće je pristupiti vrijednosti ključa elementa nad kojim se funkcija trenutno izvršava.

Nadalje, transformacija koja jako dolazi do izražaja pri obradi beskonačnih tokova je podjela na prozore (`window`, `windowAll`). Prozori dijele beskonačni tok podataka na odjeljke (*buckets*) konačne veličine nad kojima se mogu vršiti izračuni [12]. Podjelu na prozore moguće je izvesti nad običnim `DataStream` tokom i nad `KeyedStream` tokom. Razred koji je zadužen za dodjelu nadolazećeg elementa jednom ili više prozora se naziva `WindowAssigner`. Flink već ima neke unaprijed definirane dodjeljivače: *tumbling windows*, *sliding windows*, *session windows*, *global windows* te omogućuje implementaciju vlastitoga.

1.4. Tolerancija na kvarove

Flink ima razvijen mehanizam otpornosti na kvarove za konzistentan oporavak stanja strujećih aplikacija. Slike raspodijeljenog strujećeg toka podataka i stanja operatora se kontinuirano bilježe i spremaju u kontrolne točke na specificirano mjesto (primjerice, na HDFS). Ukoliko dođe do kvara, raspodijeljeno strujanje se zaustavlja i sustav se vraća u zadnju uspješno zabilježenu kontrolnu točku.

Garantirano je da će se i u slučaju pojave kvara u sustavu svaki element toka pojaviti točno jednom. Kako bi se ovakav način rada održao, potrebno je uložiti dodatno vrijeme. Ono se uglavnom kreće u rasponu od nekoliko milisekundi, no u pojedinim slučajevima zamijećen je značajan porast u vremenu obrade toka. Za aplikacije koje ne mogu priuštiti produljenje vremena obrade postoji opcija da se tolerancija na kvarove održava uz pojavu svakog elementa toka barem jednom. Dakle, postoji mogućnost pojave duplikata elemenata.

1.5. Skalabilnost

Kako bi raspodijeljeno izvodio zadatke, Flinku su potrebni resursi računalnog grozda. Platforma ima posebno implementirane modele za komunikaciju sa često korištenim upraviteljima resursa grozda. Komunikacija se vrši putem REST poziva, a resursi se zahtijevaju temeljem specificiranog paralelizma te u slučaju ispada nekog čvora za mogućnost oporavka.

Skalabilnost je osigurana mogućnošću paralelizacije posla na tisuće zadataka koji se istovremeno izvode na grozdu. Na taj način aplikacija može koristiti gotovo neograničene količine resursa procesora, glavne memorije, diska i mrežnih sučelja. [3]

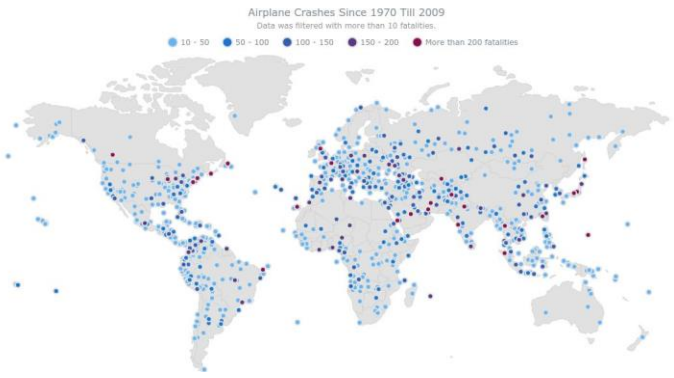
Također, korištenje stanja pri obradi toka značajno doprinosi skalabilnosti. Budući da se stanje čuva u radnoj memoriji (dok ne premaši njen kapacitet), dohvaćanje vrijednosti potrebnih pri izvođenju operacija nad dijelovima toka je znatno brže nego u slučaju kad se isti podaci nalaze na disku.

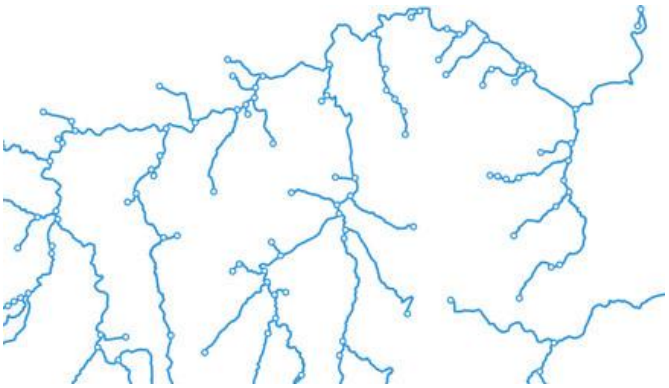
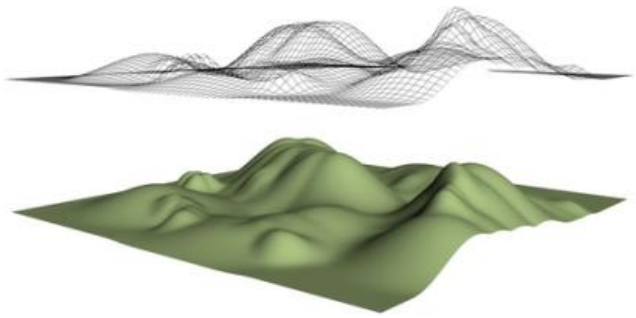
2. Geoprostorni podaci

Prostorni ili geoprostorni podaci su podaci o lokacijama i oblicima geografskih obilježja i odnosima među njima, uglavnom pohranjeni kao koordinate i topologije [13]. Koordinate određuju točku na Zemljinoj površini, a spajanjem točaka dobivaju se drugi geometrijski oblici (linije, poligoni) koji opisuju geografska obilježja na Zemlji. Tako se primjerice poligonom mogu opisati granice neke države. Topologija specificira međusobne odnose geometrijskih oblika. „Ti odnosi mogu biti jednostavni (npr. udaljenost), ali uključuju i složenije pojmove kao što su susjednost i povezanost“ [14]. Korištenjem topologija moguće je definirati raskrižja, točke, na mjestu presijecanja dviju ulica, linija ili razlomljenih linija. Također, u slučaju dva susjedna poligona, može se definirati da između njih ne postoje praznine [15]. Na ovaj način precizno je opisana granica između dviju susjednih država.

Osim točkama, linijama i poligonima, geografska obilježja mogu biti predstavljena u obliku terena. Teren je površina koju bilo koja vertikalna linija presijeca najviše u jednoj točki [1]. Korištenjem terena bilježe se reljefni oblici.

U nastavku su navedeni načini opisivanja geoprostornih podataka uz vizualne primjere.

Točka	 **Slika 5:** Zrakoplovne nesreće od 1970. do 2009. **[16]**

Linija	 **Slika 6:** Mreža rijeka **[17]**
Poligon	 **Slika 7:** Savezne države SAD-a **[18]**
Teren	 **Slika 8:** Primjer terena **[19]**

Zrakoplovna nesreća je opisana točkom koju definiraju geografska širina i dužina mjesta na Zemljinoj površini gdje je avion pao. Mrežu rijeka čini mnoštvo povezanih linija koje precizno mogu opisati tok svake rijeke. Površine saveznih država SAD-a omeđene su poligonima, a reljefni oblik neke planine opisan je pomoću terena. Topologija bi u ovim slučajevima primjerice definirala mjesto ulijevanja jedne rijeke u drugu ili točnu granicu između dviju saveznih država.

Geoprostorni podaci zapisani su u datoteke posebnog formata iz kojih se dalje mogu učitavati u geoinformacijski sustav (GIS), geoprostornu bazu podataka ili u neko drugo programsko okruženje. Poznati formati za zapis vektorskih tipova prostornih podataka (npr. točke, linije, poligona) su Shapefile, kojeg je razvio Esri, te GeoJSON, koji se bazira na JSON formatu. Nakon što su podaci modelirani i zapisani u memoriju računala, javlja se pitanje koje operacije se mogu izvršavati nad njima i kako. Gotove implementacije operacija nude se u sklopu geoinformacijskih sustava i programskih knjižnica.

2.1. Operacije nad prostornim podacima

Prema [14], to su operacije koje kao argumente primaju geoprostorne tipove podataka, a za rezultat vraćaju ili geoprostorni tip ili skalarnu vrijednost. Isti izvor dijeli operacije u tri skupine: geometrijske operacije, topološke relacije i operacije nad grafovima.

- Geometrijske operacije

Skupovne geometrijske operacije prostorne objekte gledaju kao skupove te nad njima provode izračune poput traženja presjeka ili unije. Aritmetičke operacije računaju duljinu krivulje ili površinu poligona, a od ostalih geometrijskih operacija navodi se ona za određivanje konveksne ljuske nekog skupa.

- Topološke relacije

Topološke relacije opisuje *model 9 presjeka*. Uspoređuju se dva objekta promatranjem međusobnih odnosa njihovih unutrašnjosti, granica i vanjštine. Na ovaj način moguće je odrediti nalazi li se jedan objekt unutar drugoga, ili se dva objekta možda dodiruju, ili su pak u potpunosti odvojeni.

- Operacije nad grafovima

Primjer ove operacije je pronalaženje najkraćeg puta između dviju točaka na karti.

2.2. Geoprostorni indeksi

Geoprostorni podaci su složeni, sadrže detaljan opis geometrijskih osobina i dodatne atribute (npr. ime geografskog obilježja). Zbog toga njihov zapis može dosegnuti veličinu od nekoliko gigabajta [20]. Složeni zapis slijedi i složena obrada. Neki analitički zadaci zahtijevaju puno memorijskih i procesorskih resursa, osobito pri povećanju broja geografskih elemenata uključenih u obradu [21]. Kako bi se vrijeme izvođenja operacija smanjilo, podatke valja smisleno organizirati u određene strukture.

Geoprostorni indeks je struktura podataka dizajnirana tako da omogući brzi pristup prostornim podacima [22].

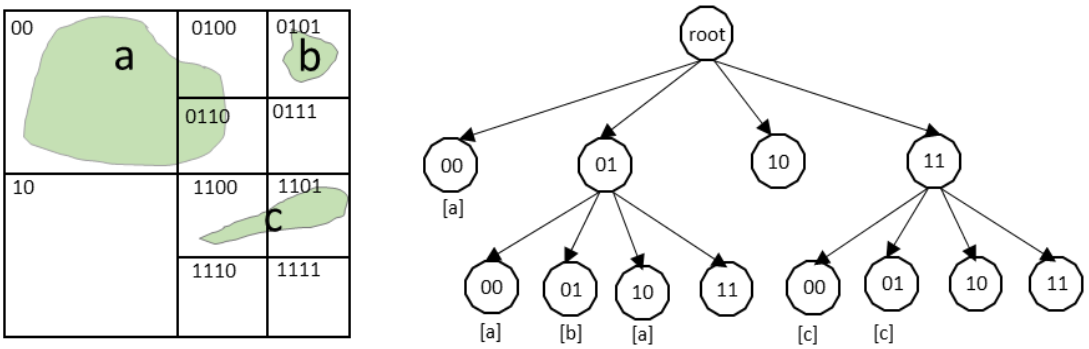
Bez uporabe indeksa pretraga bi se obavljala slijedno, ispitujući sve elemente zadanog skupa zasebno. Korištenjem indeksa dobivaju se primjetna ubrzanja obrade te su oni ključni pri manipulaciji s velikim količinama prostornih podataka. [23]

Prilikom izgradnje indeksa koristi se najmanji ograničavajući pravokutnik kako bi se geografski element približno opisao jednostavnijom strukturom. Isti se koristi i prilikom pretraživanja elemenata u indeksu. Geografski element na temelju kojeg se vrši pretraga utvrđuje se približno pravokutnikom. U skupu dobivenih rezultata mogu se pojaviti *lažni pozitivni* rezultati pa je zato potrebno još jednom provjeriti svaki rezultat, ovaj put uspoređujući sa stvarnim geografskim oblikom.

U sljedeća dva poglavlja opisane su dvije strukture često korištene pri indeksiranju geoprostornih podataka. Pojavljuju se i u programskom rješenju ovog rada. Ove strukture mogu se koristiti i u višim dimenzijama, no ovdje su objašnjene u dvodimenzionalnom prostoru zbog jednostavnosti.

2.2.1. Quadtree

Quadtree je često korištena struktura za prostorno indeksiranje. Bazira se na podjeli dvodimenzionalnog prostora na ćelije i izgradnji stabla u kojem svaki unutarnji čvor ima četiri djeteta i samo listovi sadrže informacije o geografskim objektima.

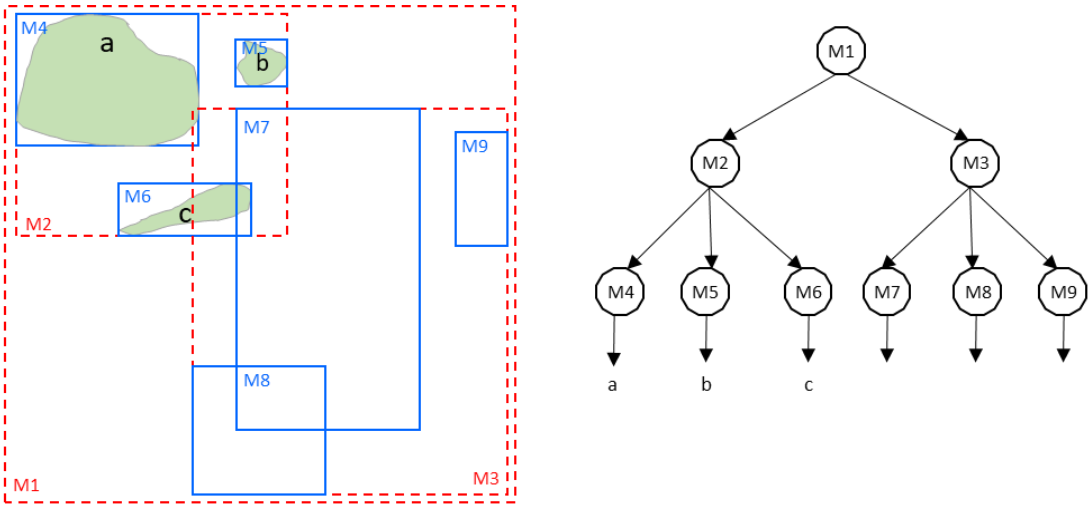


Slika 9: Quadtree – izgradnja indeksa **[23]**

Stablo (slika 9, lijevo) nastaje rekurzivnom podjelom ograničavajućeg pravokutnika nekog prostora na kvadrante (slika 9 desno). Korijen stabla pokriva cijeli prostor nad kojim se indeksiranje provodi, dalje se dijeli na pola po x-osi i po y-osi i tako nastaju četiri kvadranta – četiri djeteta. Otud potječe i ime ovog indeksa. Odluka o daljnjem dijeljenju ovisi o gustoći rasporeda geografskih elemenata u trenutnom kvadrantu. [23]

2.2.2. R-tree

R-tree je hijerarhijska dinamična struktura podataka dizajnirana za učinkovito traženje geometrijskih objekata koje presijeca zadani objekt [24]. Za razliku od Quadtree-a, čija izgradnja je temeljena na podjeli prostora u kojem se nalaze elementi, R-tree nastaje grupiranjem geometrijskih elemenata u prostoru. Elemente predstavljaju njihovi najmanji ograničavajući pravokutnici. Pravokutnici se dalje grupiraju i nastala grupa se zatim ponovo opiše pravokutnikom. Dobivena hijerarhija se pohranjuje u obliku stabla.



Slika 10: R-tree – izgradnja indeksa **[23]**

Struktura je dinamična jer se najmanji ograničavajući pravokutnici mogu promijeniti dodavanjem ili brisanjem nekog objekta iz indeksa.

3. Programsko rješenje za filtriranje geoprostornog toka podataka

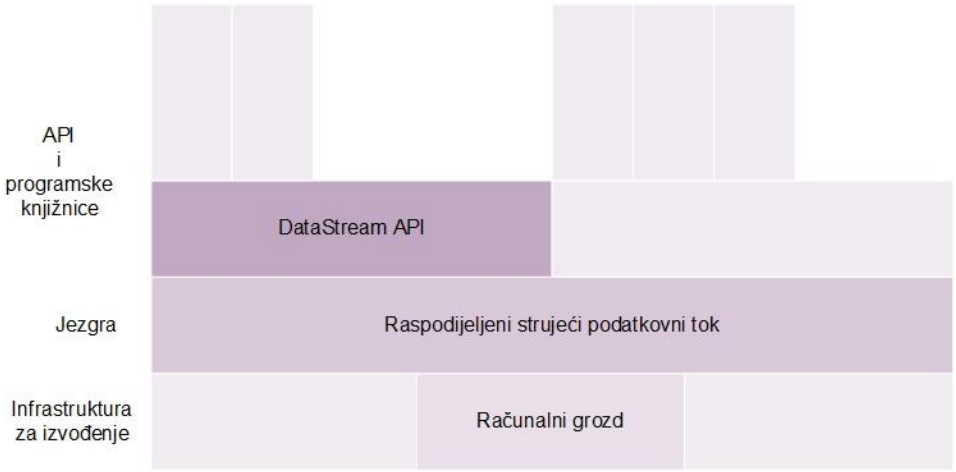
Osnova za izgradnju rješenja filtriranja geoprostornog toka podataka su funkcionalnosti koje nudi platforma Apache Flink u okviru programskog jezika Java. Nadalje, potrebne su dodatne knjižnice za rad s geoprostornim podacima, njihovo učitavanje, organizaciju i izvršavanje operacija. U tu svrhu koriste se knjižnice *GeoSpark*, *GeoTools* i JTS (*Java Topology Suite*).

Filtriranje se provodi u sustavu objava i pretplata koje sadrže geoprostornu komponentu. Objave modeliraju događaj vezan uz neko mjesto na Zemljinoj površini. Primjerice, jedno očitanje senzora temperature zraka koji je smješten na određenoj geografskoj širini i dužini. Pretplate definiraju uvjete pod kojima korisnik želi primiti obavijest o objavi. Uvjeti uključuju i podatak o geografskom području interesa. Na primjer, korisnik se pretplati na objave o izmjerenoj temperaturi zraka većoj od 5°C, na području Zagrebačke županije.

Programsko rješenje prima objave te za svaku pojedinu objavu pronalazi pretplate čije uvjete ona zadovoljava. Sve su pretplate učitane u sustav prije dolaska objava.

3.1. Arhitektura rješenja

Arhitektura Flink platforme opisana je u poglavlju 1.1. Slika 11 prikazuje koje se komponente složaja platforme koriste prilikom izgradnje i izvršavanja aplikacije za filtriranje geoprostornog toka.



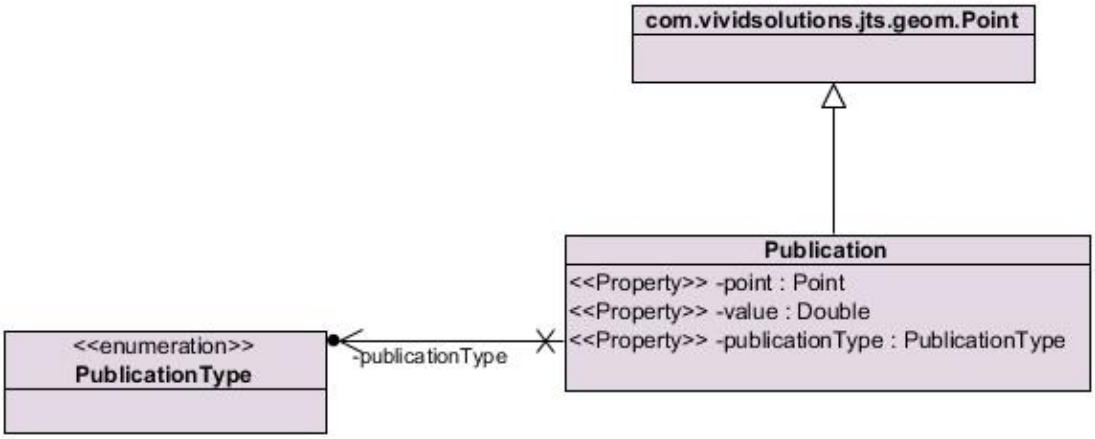
Slika 11: Arhitektura aplikacije – Flink komponentni stog

Infrastruktura za izvođenje je računalni grozd, dok je YARN (*Yet Another Resource Negotiator*) upravitelj resursa. U programskom kôdu koriste se funkcionalnosti koje nudi *DataStream* sučelje za programiranje aplikacija.

3.2. Objave

Objava, koja je geoprostorni podatak, izvedena je unutar aplikacije u obliku razreda koji nasljeđuje razred `Point` (točka) programske knjižnice JTS. Koordinate objave čine geografska širina i dužina točke na Zemljinoj površini. Dodatno, objava ima definiran tip `PublicationType`, kako bi u sustav mogle pristizati objave o različitim vrstama događaja, te brojčani parametar `value` za pohranu mjerene vrijednosti.

Slika 12 prikazuje dijagram razreda `Publication` (objava).

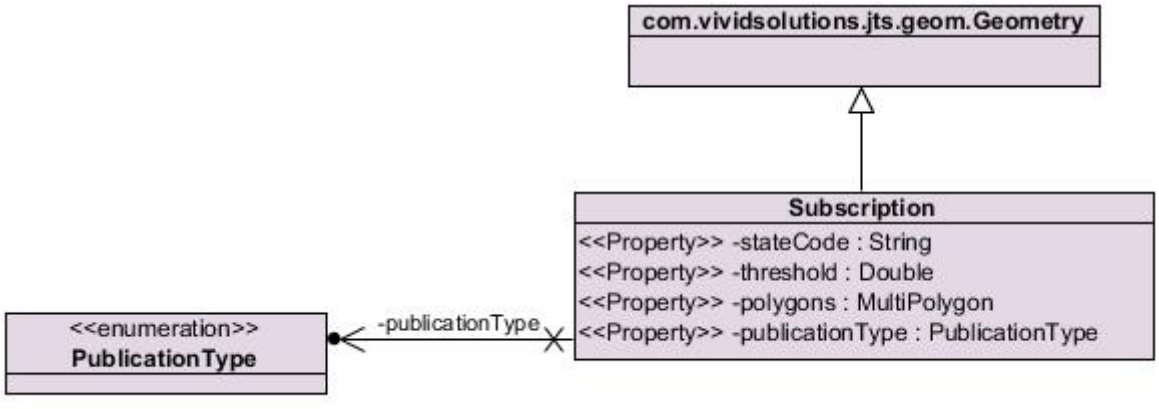


Slika 12: Dijagram razreda objave

3.3. Pretplate

Pretplata, također geoprostorni podatak, u aplikaciji je predstavljena razredom `Subscription` koji nasljeđuje razred `Geometry` programske knjižnice JTS. Pretplata definira područje na Zemljinoj površini na kojem želi pratiti objave. Područje je opisano poligonom (ili poligonima) i u tom obliku pohranjeno unutar atributa `polygons`, a naziv područja nalazi se u atributu `stateCode`. Razred pretplate sadrži i atribute kojima definira vrstu objave koja korisnika zanima (`publicationType`) te minimalnu vrijednost brojčanog parametra koju objava mora sadržavati (`threshold`).

Slika 13 prikazuje dijagram razreda `Subscription` (pretplata).



Slika 13: Dijagram razreda pretplate

3.4. Ulazne i izlazne datoteke

Ulazne datoteke pohranjuju se na HDFS (*Hadoop Distributed File System*). Pretplate i objave učitavaju se u sustav iz tekstualnih datoteka. Dodatno se učitava i *Shapefile* datoteka koja sadrži geometrijske oblike potrebne prilikom učitavanja pretplata. U tablicama 1 i 2 definirani su formati datoteka s objavama i pretplatama. Svaka pojedina objava ili pretplata zapisana je u jedan redak, a njeni parametri odvojeni su zarezom.

Tablica 1: Format datoteke s objavama

Parametar	Tip podatka
Oznaka vrste objave	String
Geografska dužina mjesta gdje je objava nastala	Double
Geografska širina mjesta gdje je objava nastala	Double
Brojčana vrijednost	Double

Tablica 2: Format datoteke s pretplatama

Parametar	Tip podatka
Naziv geografskog područja koje pretplata pokriva	String
Oznaka vrste objave	String
Minimalna vrijednost brojčanog parametra objave	Double

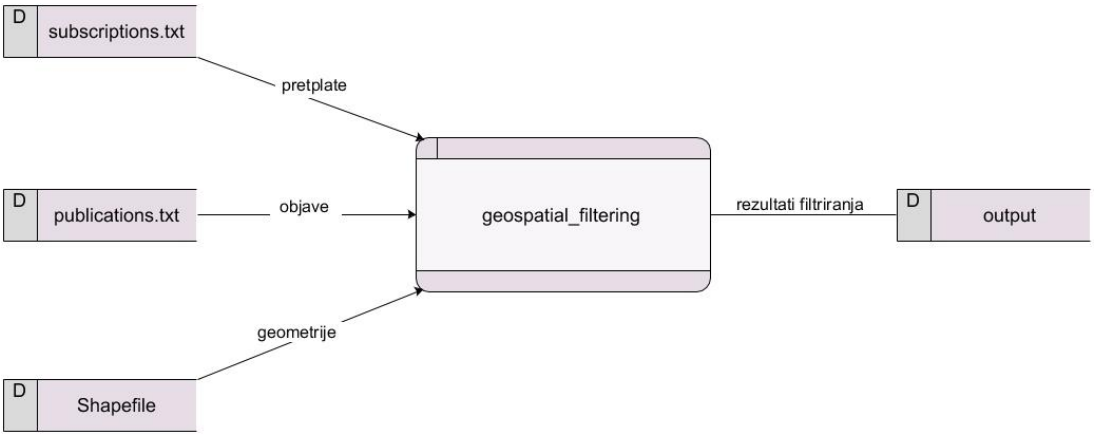
Rezultat izvođenja programa je datoteka koja za svaku objavu ispisuje pretplate dobivene filtriranjem. U jednom retku nalaze se informacije o objavi i zatim niz njoj pridruženih pretplata.

Tablica 3: Format izlazne datoteke

Parametar	Tip podatka
Koordinate objave	Point
Brojčana vrijednost objave	Double
Naziv geografskog područja koje pretplata pokriva	String
Minimalna vrijednost brojčanog parametra objave	Double

3.5. Izvedba rješenja

3.5.1. Dijagram konteksta



Slika 14: Dijagram konteksta rješenja

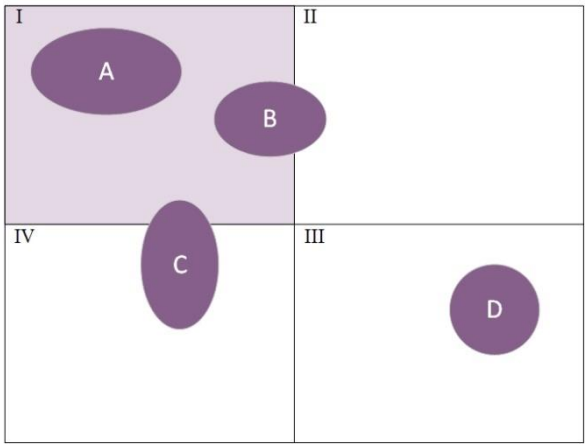
Na slici 14 prikazan je dijagram konteksta programskog rješenja. Vidljiv je sami sustav te njegove ulazne i izlazne datoteke. Sve su datoteke pohranjene, ili će biti pohranjene na HDFS-u. Prvo se učitavaju pretplate, prostorno se razdjeljuju te pohranjuju u geoprostorne indekse. Zatim se objave učitavaju u Flinkov tok podataka i pokreće se obrada. Rezultati obrade ispisuju se u izlaznu datoteku.

Prilikom organizacije pretplata koriste se knjižnice za obradu geoprostornih podataka pobrojane u uvodnom dijelu ovog poglavlja. Prostor koji prekrivaju pretplate razdijeli se na željeni broj dijelova (particija). Nad svakom particijom

izgradi se jedan indeks koji sadrži sve pretplate koje nekim dijelom svoje površine pripadaju toj particiji. Ovim načinom izbjegava se pohrana velikih količina podataka u jedan indeks i ubrzava se obrada.

Razdjeljivanje prostora rade `Partitioning` razredi i `SpatialPartitioner` razred programske knjižnice *GeoSpark* po uzoru na kôd iz izvora [25]. Dodatno, implementiran je i vlastiti `CustomPartitioner` koji nasljeđuje razred `SpatialPartitioner`.

Slika 15 prikazuje prostor podijeljen na četiri dijela i geografske elemente u njemu. Indeks prve particije sadržavat će objekte A, B i C.

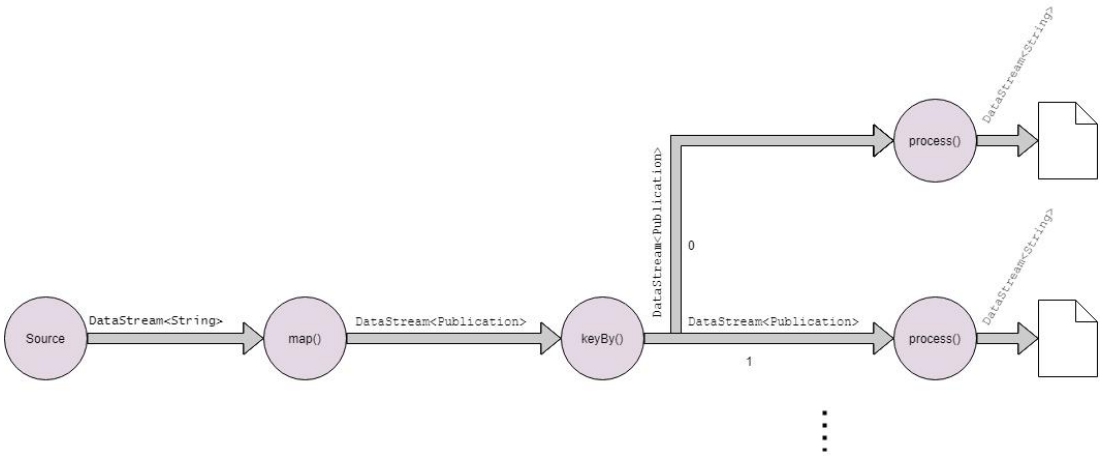


Slika 15: Podjela na particije

Geoprostorni indeksi su objašnjeni u poglavlju 2.2. U rješenju se koriste *Quadtree* i *STRtree* indeksi čije implementacije nudi programska knjižnica JTS [26].

3.5.2. Podatkovni tok

Dijagram na slici 16 prikazuje Flinkovu obradu podatkovnog toka. Bitno je napomenuti da je učitavanje i organiziranje pretplata u indekse provedeno prije pokretanja obrade toka prikazanog na slici.



Slika 16: Podatkovni tok u Flinku

Izvor toka (*Source*) je tekstualna datoteka s objavama opisana tablicom 1. Učitan je tok čiji elementi su tipa `String`. Prvi operator je `map`, koji prima redak po redak ulazne datoteke, formira objave tipa `Publication` i prosljeđuje ih dalje u tok.

Slijedi operator `keyBy` koji dijeli tok na više paralelnih. Svakom elementu se odredi ključ i na temelju njega element pripadne u jedan od paralelnih tokova. Ključ je u ovom slučaju identifikator particije u kojoj se nalazi točka objave. Na dijagramu je dan primjer s ključevima 0 i 1. Svaki se od paralelnih tokova dalje odvojeno obrađuje.

`Process` operator provodi filtriranje. Temeljem ključa objave nad kojom se operator izvodi, utvrđuje se koji je indeks potrebno pretražiti. Objave s ključem 0 pripadaju particiji s identifikatorom 0 te se pretraga pretplata provodi po indeksu izgrađenom nad tom particijom. Sada se za svaku objavu ispituje manje geografsko područje i svaka pretplata koja mora primiti obavijest o određenoj objavi nalazit će se u indeksu koji se pretražuje.

Rezultati pretraživanja se zapisuju u izlaznu datoteku opisanu tablicom 3.

3.5.3. Isječci koda

Na slikama priloženim u nastavku nalaze se isječci kôda koji prikazuju definiranje `SpatialPartitioner`-a, pripremu dovoljnog broja indeksa te dodavanje pretplata u odgovarajuće indekse temeljem particija kojima pripadaju. Slika 17 prikazuje redoslijed izvođenja koraka u glavnom programu.

```
SpatialPartitioner partitioner = subscriptionsProcessing.createPartitioner();

SpatialIndex[] indexes = new SpatialIndex[partitioner.numPartitions()];
subscriptionsProcessing.fillIndexes(partitioner, indexes);
```

Slika 17: Isječak glavnog programa

Na slici 18 je isječak metode `createPartitioner()` koja stvara objekt tipa `SpatialPartitioner`. `QuadtreePartitioning` i `QuadTreePartitioner` su razredi *GeoSpark*-ove programske knjižnice. Na temelju danih uzoraka `samples` prostor se razdjeli na particije.

Zatim se stvaraju i pune geoprostorni indeksi (slika 19). Broj nastalih indeksa jednak je broju particija koje je `partitioner` stvorio; svakoj particiji je pridružen jedan indeks. Punjenje indeksa pretplatama obavlja metoda `addToIndex` prikazana na slici 20. `Partitioner` određuje kojim sve particijama pripada poligon pretplate te se ona dodaje u pripadajuće indekse.

```
QuadtreePartitioning qtp = new QuadtreePartitioning(samples, boundingRectangle, numPartitions);
return new QuadTreePartitioner(qtp.getPartitionTree());
```

Slika 18: Definiranje partitionera

```
public void fillIndexes(SpatialPartitioner partitioner, SpatialIndex[] indexes) throws Exception {

    //initialization of indexes array, length of array is equal to the number of partitions that partitioner created
    int n = partitioner.numPartitions();
    for (int i = 0; i < n; i++) {
        indexes[i] = new Quadtree();
    }

    //Add every subscription from rawData list to a certain index
    for (Subscription s : rawData) {
        if (s != null) {
            addToIndex(s, partitioner, indexes);
        }
    }
}
```

Slika 19: Inicijalizacija indeksa

```
private void addToIndex(Subscription subscription, SpatialPartitioner partitioner, SpatialIndex[] indexes) throws Exception {

    Iterator<Tuple2<Integer, Subscription>> correspondingPartitions = partitioner.placeObject(subscription);

    while (correspondingPartitions.hasNext()) {
        int i = correspondingPartitions.next()._1;

        indexes[i].insert(subscription.getEnvelopeInternal(), subscription);
    }
}
```

Slika 20: Metoda `addToIndex`

4. Usporedba performansi filtriranja na odabranom studijskom slučaju

Razvijeno programsko rješenje primijenjeno je na konkretnom primjeru objava i pretplata. Točnije, objave su točke koje predstavljaju početno mjesto određene vremenske nepogode na području SAD-a, a pretplate su poligoni koji omeđuju pojedine savezne države.

Postoje tri vrste objava: objave o tornadu, o nevremenu s tučom ili o jakom vjetru. Objava definira vrstu nevremena, geografsku širinu i dužinu početne točke nevremena te intenzitet. S druge strane, jedna pretplata pokriva područje jedne savezne države SAD-a, definira jednu vrstu nevremena (objave) o kojoj želi primati obavijesti te njen minimalni intenzitet. Intenzitet tornada određen je prema F ljestvici (za neke objave po EF ljestvici), u slučaju tuče prati se veličina ledenih kapi u inčima, a intenzitet vjetra određuje njegova brzina izražena u čvorovima.

Primjerice, ako dođe objava o tornadu čija početna točka je na 39.1° sjeverne geografske širine i 89.3° zapadne geografske dužine i intenziteta je 3 (po EF ljestvici), tada sve pretplate, koje su zainteresirane za pojavu tornada tog intenziteta na području savezne države Illinois, moraju primiti obavijest.

4.1. Ulazne datoteke

Skup podataka o nepogodama je preuzet sa *stranica Storm Prediction Center*-a [27]. Preuzete su .zip arhive s podacima o tornadima, nevremenima s tučom i jakim vjetrovima u razdoblju od 1955. (za tornado 1950.) do 2018. godine. Prilikom stvaranja datoteke s objavama (tablica 1) koristi se informacija o kojoj nepogodi je riječ, geografskoj širini i dužini početne točke nepogode i njenom intenzitetu. Napomena: intenzitet tornada određen je prema F ljestvici, a za podatke nastale nakon siječnja 2007. godine, po EF ljestvici.

Datoteka s pretplatama stvorena je iz datoteke s objavama s ciljem da područja na kojima se pojavljuju vremenske nepogode budu pokrivena pretplatama (tablica 2). Dodatno, korištena je *Shapefile* datoteka za smještanje

točaka u savezne države i kasnije za dohvat njihovih geometrija. Preuzeta je iz arhive s_11au16.zip sa stranica *National Weather Service*-a [28].

4.2. Opis i pretpostavka eksperimenta

Cilj eksperimenta je proučiti kako odabir podatkovnih struktura, broja particija i paralelizam utječu na vrijeme izvođenja programa. Točnije, mjereno vrijeme obuhvaća postupak učitavanja datoteka, razdjeljivanje prostora pretplata, pohranu pretplata u odgovarajuće geoprostorne indekse, obradu svih nadolazećih objava i ispis rezultata u izlaznu datoteku.

Prilikom mjerenja bira se paralelizam izvođenja, broj particija na koje će prostor pretplata biti podijeljen, struktura za raspodjelu prostora te struktura za indeksiranje pretplata. Nakon uspješno izvršenog programa zabilježeno je vrijeme izvođenja pod oznakom „Job Runtime“.

Mjerenja su provedena na računalnom grozdu. Temeljem zadanog paralelizma Flink je osigurao sve resurse, u obliku *Task Managera* i *Task Slotova*, potrebne za izvođenje programa. Pojedini *Task Manager* raspolagao je memorijom od 1024 MB. Skup ulaznih datoteka sadržavao je 500 objava i 100 pretplata.

Eksperiment pretpostavlja kraće vrijeme izvođenja programa pri paralelnoj obradi u odnosu na neparalelnu. Također, predviđa se brže izvođenje u slučaju odabira *Voronoi* i *R-tree* strukture podataka za podjelu prostora pretplata zbog bolje prilagodbe nad danim skupom podataka.

4.3. Parametri eksperimenta

Paralelizam se zadaje prilikom pokretanja programa na računalnom grozdu i određuje na koliko paralelnih instanci će *zadatak* biti podijeljen. Prilikom mjerenja izmjenjivale su se sljedeće vrijednosti paralelizma: 1, 4, 8, 16.

Geoprostorne strukture u programu imaju dvije uloge: dijele prostor pokriven pretplatama na zadani broj particija i pohranjuju pretplate. U eksperimentu prostor pretplata se dijelio na 4, 8, 16 i 64 particije.

Programske knjižnice korištene prilikom rada s geoprostornim podacima nude više različitih struktura za indeksiranje i razdjeljivanje podataka. JTS knjižnica implementira dvije vrste geoprostornih indeksa koji se u eksperimentu koriste za pohranu pretplata: *Quadtree* i *STRtree*. *GeoSpark* implementira različite `Partitioning` i `Partitioner` razrede čijim se uparivanjem dobivaju razne metode razdjeljivanja podataka.

Indeksi JTS knjižnice kao osnovu imaju indekse objašnjene u poglavlju 2.2., a sljedeće poglavlje ukratko opisuje načine razdjeljivanja korištene u ovom eksperimentu.

4.3.1. Razdjeljivanje prostora pretplata

Nakon učitavanja pretplata stvara se lista uzoraka koja sadrži najmanje ograničavajuće pravokutnike svih pretplata. Razdjeljivanje prostora radi se na temelju pravokutnika u listi uzoraka.

- *Equalgrid*

Prostor se dijeli na pravokutnike – particije jednake veličine.

- *Voronoi*

Voronoijev dijagram gradi poligone nad zadanim skupom točaka S tako da se unutar svakog poligona nađe točno jedna točka. Svaka točka ravnine koja se

nalazi u poligonu najmanje je udaljena točno od one točke skupa S oko koje je taj poligon izgrađen.

- *Quadtree*

Dijeli prostor pretplata izgradnjom *Quadtree* strukture (poglavlje 2.2.1.).

- *Rtree*

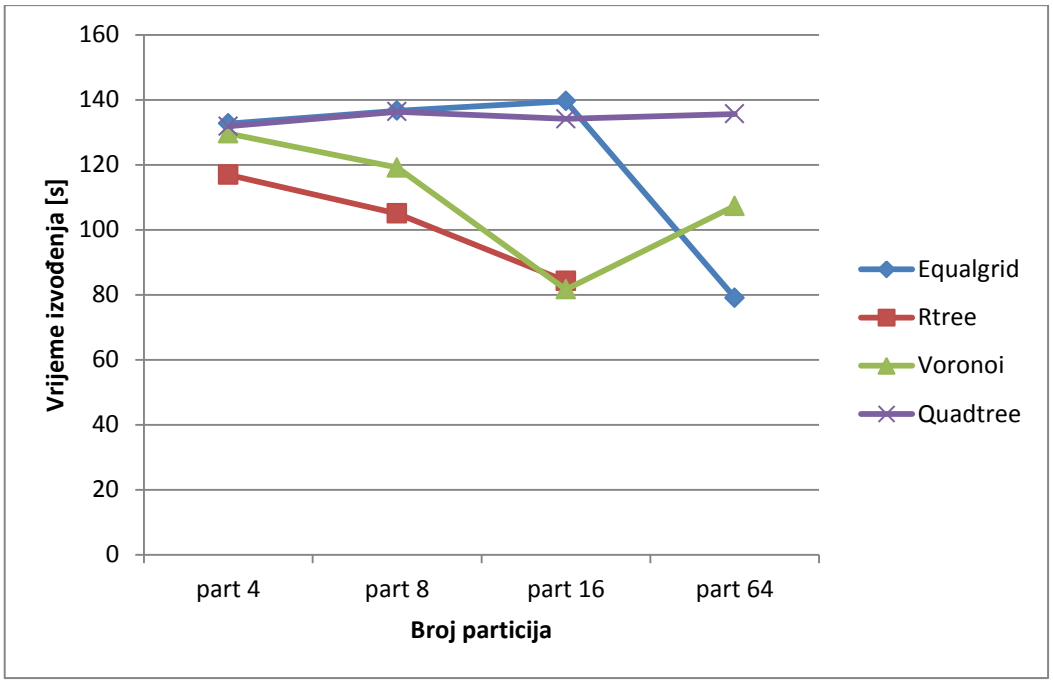
Dijeli prostor pretplata izgradnjom *R-tree* strukture (poglavlje 2.2.2.).

4.4. Rezultati

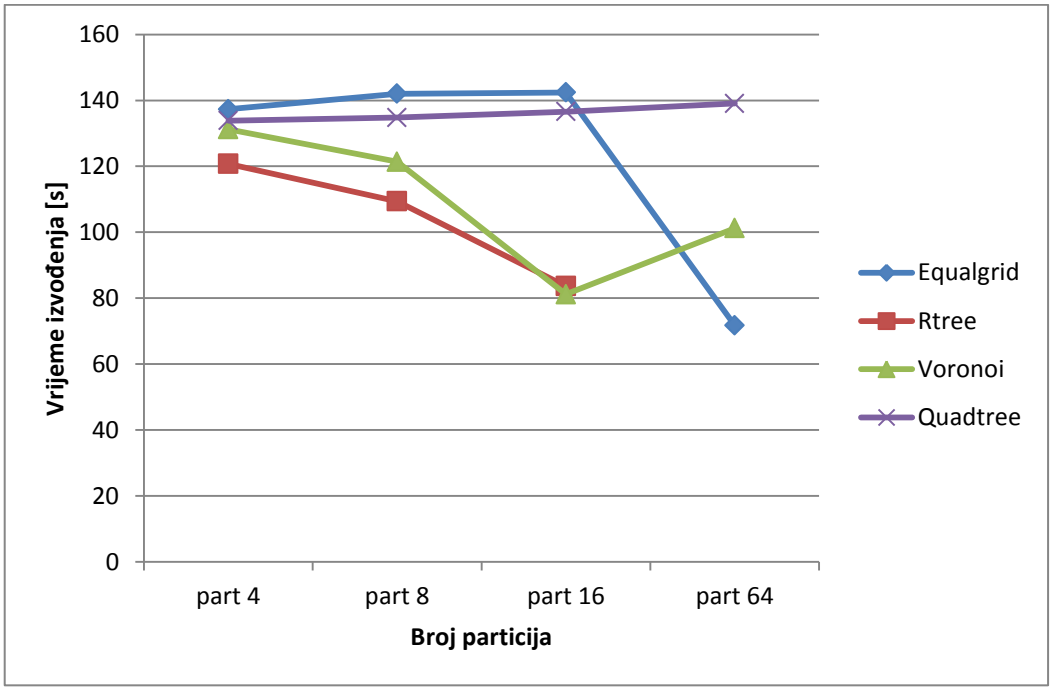
Vrijednosti su mjerene u sekundama i zapisane u tablicu. Grafikoni u nastavku izdvojeni su kao reprezentativni uzorci rezultata dobivenih mjerenjima.

- **Usporedba utjecaja broja particija na vrijeme izvođenja**

Sljedeća dva grafikona prikazuju kako se kreće trajanje izvođenja programa prilikom povećanja broja particija uz konstantan paralelizam od 16 paralelnih instanci izvođenja. Uspoređuju se četiri odabrana načina razdjeljivanja prostora. Na slici 21 je za pohranu pretplata korišten indeks *Quadtree*, a na slici 22 *STRtree*.



Slika 21: Usporedba vremena izvođenja prilikom povećanja broja particija (indeks za pohranu pretplata: *Quadtree*)



Slika 22: Usporedba vremena izvođenja prilikom povećanja broja particija (indeks za pohranu pretplata: *STRtree*)

Podaci prikazani na grafikonima ukazuju da odabir indeksa korištenog za pohranu pretplata na vrlo sličan način utječe na konačno vrijeme izvođenja. Oblici linija i trendovi rasta i pada gotovo se podudaraju na obje slike. Znatnije razlike vidljive su u odabiru strukture za razdjeljivanje podataka. Valja uzeti u obzir činjenicu da raspršenost pretplata u prostoru, i njihov oblik, utječe na performanse pojedine strukture.

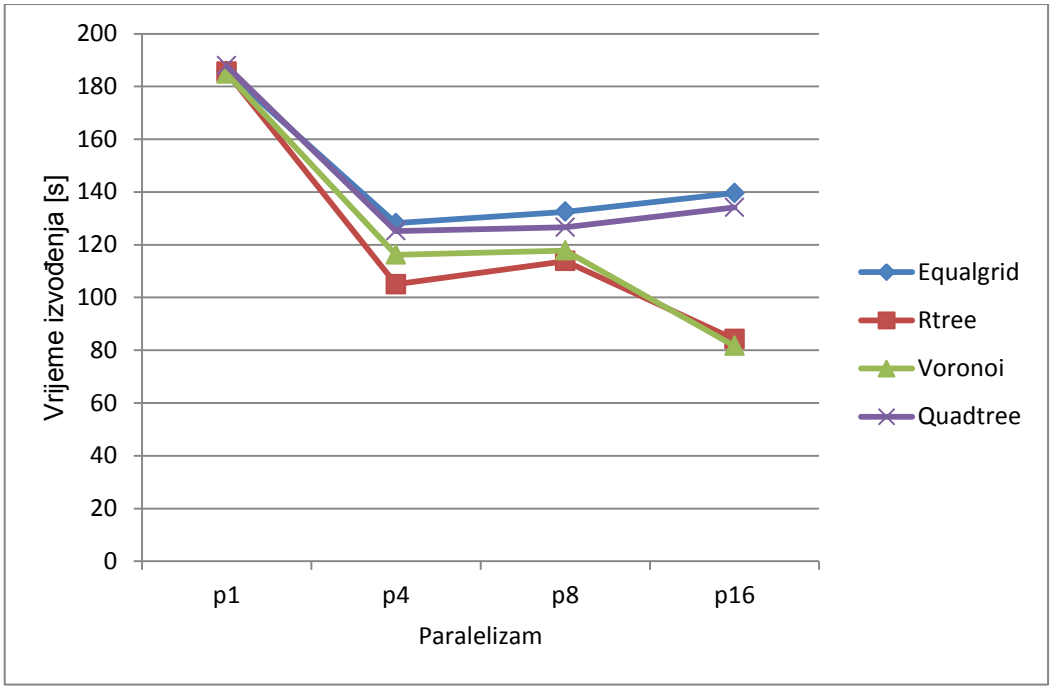
Međusobno slično ponašanje primjetno je prilikom korištenja *Rtree* i *Voronoi* načina razdjeljivanja. Grafikoni prikazuju da se najbolje vrijeme obrade postiže prilikom podjele na 16 particija. Oba načina podjelu prostora prilagođavaju rasporedu elemenata u njemu. Povećanjem broja particija (do 16), sve bolje se opisuju poligoni pretplata. Daljnjim povećanjem broja particija, *Veronoijeva* podjela rezultira rastom vremena obrade, dok u slučaju *Rtree* podjele na 64 particije program nije uspio izgraditi stablo, stoga rezultat nije zabilježen u grafikonu. Izvedena je dodatna provjera postupnim dodavanjem novih pretplata u stari skup dok se stablo nije uspješno izgradilo. Dobiven rezultat pokazao je da vrijeme izvođenja još kraće od onog zabilježenog pri podjeli na 16 particija.

Zanimljivo je značajno smanjenje vremena izvođenja u slučaju *Equalgrid* načina razdjeljivanja pri odabiru 64 particije, postignuto dobrim preklapanjem particija razdijeljenog prostora i zadanog skupa podataka. *Quadtree* podjela povećanjem broja particija ne pokazuje značajne promjene u brzini obrade.

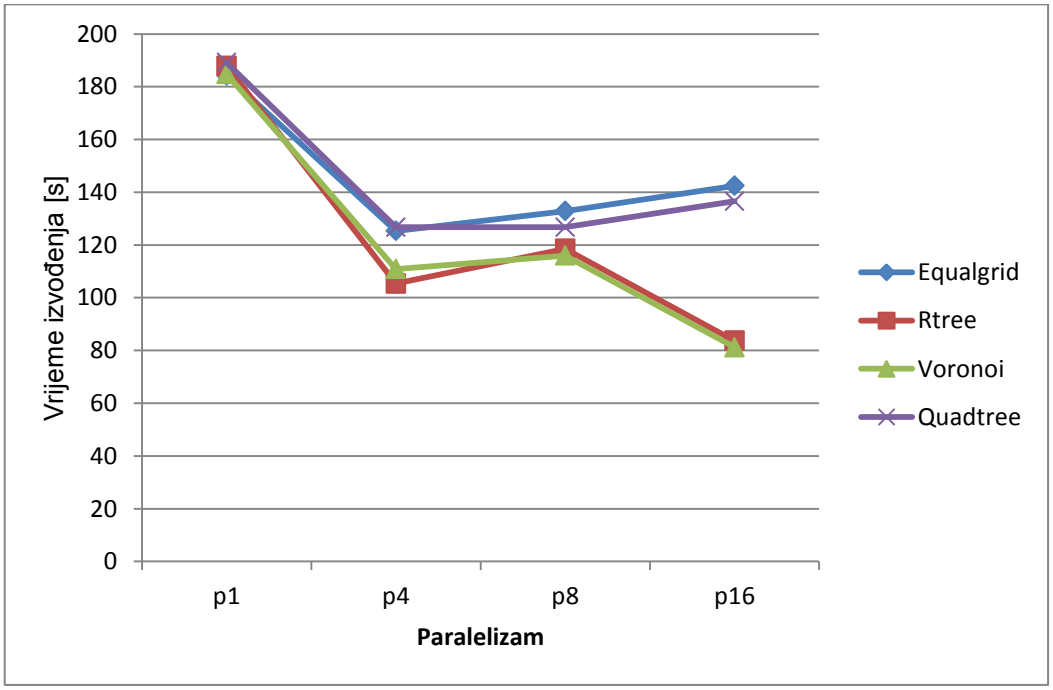
Pokazuje se da je podjela pretplata, odnosno ulaznih podataka, i njihova pohrana, usko vezana uz sam odabir radnog skupa podataka.

- **Usporedba utjecaja paralelizma na vrijeme izvođenja**

Grafikoni na sljedeće dvije slike prikazuju kako se kreće vrijeme izvođenja programa prilikom povećanja paralelizma uz konstantnih 16 particija. Ponovno se uspoređuju četiri odabrana načina razdjeljivanja prostora unutar jednog grafikona. Prilikom mjerenja koje prikazuje slika 23 korišten je *Quadtree* indeks za pohranu pretplata, dok je na slici 24 korišten *STRtree* indeks.



Slika 23: Usporedba vremena izvođenja prilikom povećanja paralelizma (indeks za pohranu pretplata: *Quadtree*)



Slika 24: Usporedba vremena izvođenja prilikom povećanja paralelizma (indeks za pohranu pretplata: STR*tree*)

Grafikoni jasno pokazuju značajno ubrzanje pri raspodijeljenoj obradi u odnosu na centraliziranu. Ubrzanje je prisutno kod svakog načina podjele prostora pretplata.

Daljnjim povećanjem paralelizma, *Equalgrid* i *Quadtree* pokazuju ponovno, blago, produljenje vremena obrade, dok *Voronoi* i *R-tree* strukture imaju znatno poboljšanje u trajanju izvođenja programa pri paralelizmu 16 i istim zadanim brojem particija. U opisanom slučaju svaki od 16 procesa radnika može biti zadužen za jednu od 16 particija koje nisu prazne te se koristan posao može dobro rasporediti.

4.5. Zaključak eksperimenta

Provedena mjerenja potvrđuju pretpostavku eksperimenta o kraćem izvođenju programa pri raspodijeljenoj obradi i korištenju *R-tree* i *Voronoijeve* strukture u svim slučajevima, osim izoliranog slučaja korištenja *Equalgrid* strukture.

Nadalje, rezultati eksperimenta pokazuju jedan stalni uzorak: obrada koja koristi *R-tree* strukturu gotovo uvijek je brža od obrade koja koristi *Quadtree* strukturu. Navedeni uzorak primijećen je i u radu [29].

Najbolje značajke pojedinih struktura za organizaciju geoprostornih podataka, zajedno s vrijednostima paralelizma, dolaze do izražaja u ovisnosti o zadanom skupu podataka. Zbog toga opisani reprezentativni uzorci rezultata daju poticaj daljnjem istraživanju uz proširenje skupa parametara te odabranog skupa podataka.

5. Budući rad

Programsko rješenje razvijeno uz ovaj rad stavlja naglasak na smislenom organiziranju geoprostornih podataka s ciljem ubrzanja filtriranja, otvarajući mogućnosti za daljnje dorade i nadogradnje.

Pretplate su u ovom slučaju statične – učitavaju se prije dolaska objava, organiziraju u indekse i čuvaju u radnoj memoriji. Prijedlog je dodati mogućnost pretplaćivanja i onda kada objave već stižu u sustav. Rješenje bi moglo biti dodavanje pretplata u tok objava. U tom slučaju, prvi zadatak je identificirati je li pridošli element pretplata ili objava. Zatim, razdvojiti tok objava od toka pretplata te ih odvojeno obraditi. Pretplate smještati u odgovarajuće indekse na temelju particija kojima pripadaju, a objave obrađivati na isti način kao i do sada: usmjerujući pretragu na indeks particije u kojoj se objava nalazi.

Nadalje, pretplate i objave u prikazanom programskom rješenju učitavaju se iz tekstualne datoteke. Ukoliko se za izvor toka postavi mrežna pristupna točka (*socket*), sustav može primati pretplate i objave s udaljenih klijenata. Na ovaj način, rješenje je moguće implementirati u veći sustav u kojem će primati objave nastale iz trenutnih mjerenja senzora i pretplate od stvarnih korisnika. Primjerice, senzor izmjeri brzinu vjetra, formira se objava sa svim potrebnim parametrima (identifikator koji označava da je riječ o vjetru, geografske koordinate senzora i izmjerena brzina vjetra) i šalje se sustavu koji dalje pronalazi sve pretplate koje interesira ovakva vrsta objava. Budući da su u ovom slučaju pretplatnici stvarni korisnici, ideja je proslijedit im obavijest. Umjesto ispisa u izlaznu datoteku, korisniku bi se mogla poslati elektronička poruka sa sadržajem objave.

Sustav zasad prima objave s geografskim koordinatama točke na Zemljinoj površini, no mnogi podaci su vezani uz veće područje. Putanju tornada moguće je opisati izlomljenom linijom (*polilinijom*), a nevrijeme s tučom pomoću poligona. Objave je u tom slučaju potrebno proširiti da modeliraju neki drugi geoprostorni tip podatka.

Navedenim promjenama sustav je moguće prilagoditi za primjenu u stvarnoj okolini.

6. Zaključak

Velika količina podataka koja svakodnevno nastaje može se asocirati s nekim geografskim područjem. Takvi podaci nazivaju se geoprostornima jer uz ostale informacije, nose i onu o svom obliku i položaju na Zemljinoj površini. Složenosti podatka uvelike pridonosi složenost geometrije koja ga opisuje. Podaci se mogu modelirati jednostavnim geometrijskim oblicima, primjerice točkama, i složenim, poligonima ili terenima. Kako bi se nad njima efikasno mogle izvoditi operacije, potrebno ih je smisleno organizirati. Programske knjižnice i alati za rad s geoprostornim podacima imaju implementirane različite vrste indeksa, od kojih se često spominju *Quadtree* i *STRtree*. Oni dodatno mogu poslužiti i u mehanizmima podjele podataka u particije.

Nadalje, za rad s velikim količinama podataka potrebna je, uz dobru organizaciju podataka, i robusna platforma na kojoj će se operacije izvršavati. U ovom radu je za tu svrhu korišten Apache Flink koji je u mogućnosti vršiti raspodijeljenu obradu ograničenih i neograničenih tokova podataka uz održavanje stanja. Jedinstven je po tome što istovremeno optimizira obradu i strujećih (*streaming*) i skupnih (*batch*) podataka.

U okviru rada, korištenjem platforme Apache Flink, razvijena je aplikacija za filtriranje geoprostornog toka podataka. Filtriranje je provedeno u sustavu objava i pretplata, gdje su objave elementi toka, a pretplate statični podaci koji čekaju dolazak objava koje ih interesiraju. Korištena je operacija pretvorbe tipa elemenata koji su u toku, operacija odjeljivanja toka podataka po ključu i operacija koja omogućuje obradu elemenata toka na niskoj razini. Za rad s geoprostornim podacima korištene su knjižnice JTS (*Java Topology Suite*), *GeoSpark* i *GeoTools*.

Pomoću programskog rješenja provedeno je istraživanje o brzini obrade geoprostornih podataka (objava i pretplata) u ovisnosti o odabiru struktura za indeksiranje i razdjeljivanje prostora. Dodatni parametri uzeti u obzir prilikom mjerenja su paralelizam izvođenja programa i broj particija na koje se prostor dijeli. Dobiveni rezultati naznačuju važnost pomnog odabira strukture za organizaciju geoprostornih podataka. Potrebno je proučiti i analizirati skup geoprostornih

podataka nad kojim se gradi struktura. Neke strukture, kao što je *R-tree*, su poznate po osiguravanju dobrih performansi, no postoje situacije u kojima i „jednostavnije“ strukture uspiju postići zamjetne rezultate ukoliko im odgovara raspored podataka u prostoru. Također, na odabranom skupu podataka je uočeno da povećanje paralelizma, u odnosu na vrijednost paralelizma 1, značajno ubrzava obradu.

Konačno, proces obrade geoprostornih podataka potrebno je pažljivo analizirati kako bi se postigle što bolje performanse prilikom obrade podataka. Postupci analiziranja kreću se od odabira geometrijske strukture kojom će podatak biti opisan i zapisan u memoriju računala, sve do odabira strukture za organizaciju i indeksiranje skupa geoprostornih podataka. Postoje razne programske knjižnice i platforme za brzu obradu, no samo dobrim razumijevanjem odabranog skupa podataka i pažljivim korištenjem alata moguće je postići najbolje rezultate.

7. Literatura

[1] L. Toma, gis_datamodels.key - gis_datamodels.pdf, 10. 9. 2015. https://www.bowdoin.edu/~ltoma/teaching/cs3225-GIS/fall15/Lectures/gis_datamodels.pdf., [Datum pristupa 28. 5. 2019.].

[2] B. Davies, 5 Best Data Processing Frameworks, 6. 3. 2019., https://www.knowledgehut.com/blog/big-data/5-best-data-processing-frameworks., [Datum pristupa 23. 5. 2019.].

[3] The Apache Software Foundation, What is Apache Flink?, 10. 4. 2018., https://flink.apache.org/flink-architecture.html., [Datum pristupa 27. 5. 2019].

[4] F. Hueske i V. Kalavri, Stream Processing with Apache Flink: Fundamentals, Implementation, and Operation of Streaming Applications, Prvo izdanje ur., Sebastopol: O'Reilly Media, Inc., 2019.

[5] The Apache Software Foundation, Apache Flink 1.8 Documentation: Component Stack, 2019., https://ci.apache.org/projects/flink/flink-docs-release-1.8/internals/components.html., [Datum pristupa 16. 6. 2019.].

[6] The Apache Software Foundation, Apache Flink 1.8 Documentation: Distributed Runtime Environment, 2019., https://ci.apache.org/projects/flink/flink-docs-release-1.8/concepts/runtime.html., [Datum pristupa 1. 6. 2019.].

[7] S. Ewen, F. Hueske i X. Jiang, Apache Flink: Batch as a Special Case of Streaming and Alibaba's contribution of Blink, 13. 2. 2019., https://flink.apache.org/news/2019/02/13/unified-batch-streaming-blink.html., [Datum pristupa 1. 6. 2019].

[8] P. Nowojski, Exactly-Once Processing in Apache Flink (with Apache Kafka, too!), 15. 2. 2018., https://www.ververica.com/blog/end-to-end-exactly-once-processing-apache-flink-apache-kafka., [Datum pristupa 2. 6. 2019.].

[9] The Apache Software Foundation, Apache Flink 1.8 Documentation: Working with State, 2019., https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/stream/state/state.html., [Datum pristupa 26. 5. 2019.].

[10] The Apache Software Foundation, Apache Flink 1.8 Documentation: Operators, 2019., https://ci.apache.org/projects/flink/flink-docs-stable/dev/stream/operators/., [Datum pristupa 1. 6. 2019.].

[11] The Apache Software Foundation, Apache Flink 1.8 Documentation: Process Function (Low-level Operations), 2019., https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/stream/operators/process_function.html., [Datum pristupa 2. 6. 2019.].

[12] The Apache Software Foundation, Apache Flink 1.8 Documentation: Windows, 2019., https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/stream/operators/windows.html., [Datum pristupa 4. 6. 2019.].

[13] Esri, spatial data| Definition - Esri Support GIS Dictionary, 2018., https://support.esri.com/en/other-resources/gis-dictionary/term/d70395b4-f5c4-4375-b053-c2d3b24a3bce., [Datum pristupa 12. 6. 2019.].

[14] ZPR-FER - Zagreb, Microsoft PowerPoint - 4. GeoDB.pptx - 4._GIS.pdf, 10. 2018., https://www.fer.unizg.hr/_download/repository/4._GIS.pdf., [Datum pristupa 12. 6. 2019.].

[15] Esri, topology| Definition - Esri Support GIS Dictionary, 2018., https://support.esri.com/en/other-resources/gis-dictionary/term/b548f294-7f68-4afa-aaee-82f92e54e5a8., [Datum pristupa 12. 6. 2019.].

[16] AnyChart, Airplane Crashes since1970 till 2009, 2019., https://www.anychart.com/products/anymap/gallery/Maps_Point_Maps_(Dot_Maps)/Airplane_Crashes_since_1970_till_2009.php., [Datum pristupa 12. 6. 2019.].

[17] Ordnance Survey, OS Open Rivers, 2016., https://www.ordnancesurvey.co.uk/business-and-government/products/os-open-rivers.html., [Datum pristupa 12. 6. 2019.].

[18] Clker-Free-Vector-Images, Image by Clker-Free-Vector-Images from Pixabay, 2019., https://pixabay.com/vectors/usa-map-united-states-of-america-35713/. [Datum pristupa 18. 6. 2019.].

[19] Total Surveys Ltd, Digital Terrain Modelling (DTM), 2011., http://www.totalsurveys.co.uk/services-provided/digital-terrain-modelling.aspx., [Datum pristupa 12. 6. 2019.].

[20] Library of Congress, ESRI Shapefile, 2019., https://www.loc.gov/preservation/digital/formats/fdd/fdd000280.shtml., [Datum pristupa 14. 6. 2019.].

[21] M. J. de Smith, M. F. Goodchild i P. A. Longley, Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools, Treće izdanje ur., Leicester: The Winchelsea Press, 2011.

[22] Y. Manolopoulos, Y. Theodoridis i V. J. Tsotras, Spatial Indexing Techniques. In: LIU L., ÖZSU M.T. (eds) Encyclopedia od Database Systems, Springer, Boston, MA, 2009.

[23] X. Zang i Z. Du, Spatial Indexing. The Geographic Information Science & Technology Body of Knowledge (4th Quarter 2017 Edition), John P. Wilson (ed), 2017.

[24] S. T. Leutenegger, J. M. Edgington i M. A. Lopez, STR: a simple and efficient algorithm for R-tree packing, *Proceedings 13th International Conference on Data Engineering,* pp. 497-506, 1997.

[25] Data Systems Lab, DataSystemsLab/GeoSpark · GitHub, 2019., https://github.com/DataSystemsLab/GeoSpark/blob/aa9ec969d6c6eacfee6306c7bd4e4985a0d460c7/core/src/main/java/org/datasyslab/geospark/spatialRDD/SpatialRDD.java#L250., [Datum pristupa 10. 5. 2019.].

[26] Locationtech, Interface SpatialIndex, https://locationtech.github.io/jts/javadoc/org/locationtech/jts/index/SpatialIndex.html., [Datum pristupa 15. 6. 2019.].

[27] NCEP/SPC Web Team, Storm Prediction Center WCM Page, 9. 9. 2018., https://www.spc.noaa.gov/wcm/. [Datum pristupa 15. 4. 2019.].

[28] US Department of Commerce, NOAA, National Weather Service, U.S. States and Territories, 16. 5. 2018., https://www.weather.gov/gis/USStates. [Datum pristupa 15. 4. 2019.].

[29] R. Kanth i V. Kothuri, Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data, 1. 2002.

Sažetak

Filtriranje geoprostornog toka podataka korištenjem platforme Apache Flink

Geoprostorni podaci složeni su podaci koji uz ostale svoje parametre sadrže i geografsku komponentu. Modeliraju se geometrijskim oblicima točke, linije i poligona, kao i složenijim strukturama za opisivanje reljefnih površina. Njihova je obrada složena te je potrebno koristiti posebne strukture za bolju organizaciju velike količine ove vrste podataka. Uz dobru organizaciju, bitno je i brzo izvođenje operacija te raspodjela poslova. Razvijeno je rješenje za filtriranje geoprostornog toka podataka i temelji se na platformi Apache Flink koja omogućuje raspodijeljenu obradu ograničenih i neograničenih tokova podataka. Sustav pohranjuje pretplate u obliku poligona, a objave, točke, dolaze u sustav u obliku toka podataka. Cilj je ubrzati pronalazak pretplata koje trebaju primiti obavijest o nekoj objavi. Rješenje je korišteno i za provedbu istraživanja u kojem se uspoređuju vremena obrade prilikom organizacije podataka u različite geoprostorne strukture.

Ključne riječi: geoprostorni podaci, geoprostorni indeks, Apache Flink, tok podataka, stanje, pretplate, objave, razdjeljivanje prostora

Summary

Filtering of Geospatial Data Streams using Apache Flink Platform

Geospatial data is a composite type of data which contains geospatial component, along with other parameters. It is modeled with different geometry shapes, such as point, line and polygon, as well as with some more complex structures for terrain modelling. It is recommended to organize this type of data in special structures to achieve better processing time. For even better performance, some of known big data processing platforms may be used. In this thesis, an application for filtering geospatial streams has been developed based on Apache Flink, a platform for bounded and unbounded stream process. The system stores subscriptions in the shape of polygons, and the publications are brought to the system as a data stream. Publications are points, in geospatial data context. The goal is to determine which subscriptions need to receive a notification about newly arrived publication in the shortest possible time. Finally, a research was conducted using developed solution, in which the processing time was measured when using different data organizing structures.

Key words: geospatial data, spatial index, Apache Flink, data stream, state, subscriptions, publications, partitioning