Raspodijeljeni geoprostorni sustav objavi-pretplati s visokom učestalošću objavljivanja

Livaja, Ivan

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Ivan Livaja

RASPODIJELJENI GEOPROSTORNI SUSTAV OBJAVI-PRETPLATI S VISOKOM UČESTALOŠĆU OBJAVLJIVANJA

DOKTORSKI RAD

Zagreb, 2023.

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Ivan Livaja

RASPODIJELJENI GEOPROSTORNI SUSTAV OBJAVI-PRETPLATI S VISOKOM UČESTALOŠĆU OBJAVLJIVANJA

DOKTORSKI RAD

Mentor: prof. dr. sc. Krešimir Pripužić

Zagreb, 2023.

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Ivan Livaja

DISTRIBUTED GEOSPATIAL PUBLISH/SUBSCRIBE SYSTEM WITH HIGH PUBLICATION FREQUENCY

DOCTORAL THESIS

Supervisor: Professor Krešimir Pripužić, PhD

Zagreb, 2023

Doktorski rad izrađen je na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, na Zavodu za telekomunikacije.

Mentor: prof. dr. sc. Krešimir Pripužić

Doktorski rad ima: 116
stranice

Doktorski rad br.: _____

O mentoru

Krešimir Pripužić je redoviti profesor na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER) gdje predaje kolegije u području raspodijeljenih informacijskih sustava i objektno-orijentiranog programiranja. Diplomirao je u polju elektrotehnike, a doktorirao je u polju računarstva na FER-u 2003., odnosno 2010. godine. Zaposlen je na Zavodu za telekomunikacije FER-a od 2003. godine. Tijekom doktorskog studija proveo je akademsku godinu 2006./2007. kao stipendist Švicarske konfederacije na sveučilištu École Polytechnique Fédérale de Lausanne u laboratoriju Laboratoire de Systèmes d'Information Répartis. U srpnju 2022. godine izabran je u zvanje redovitog profesora. Sudjelovao je u nizu istraživačkih projekata financiranih iz domaćih izvora i fondova EU, a trenutno je voditelj FER-ovog Laboratorija za tokove podataka. Sudjeluje kao istraživač u radu Znanstvenog centra izvrsnosti za znanost o podatcima i kooperativne sustave koji je prvi je nacionalni centar izvrsnosti u području tehničkih znanosti u Hrvatskoj. Objavio je preko 40 znanstvenih radova u časopisima i zbornicima radova u području raspodijeljenih sustava, Interneta stvari i obrade velikih podataka. Član je stručne udruge IEEE.

About the Supervisor

Krešimir Pripužić is a Full Professor at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (FER) where he teaches distributed information systems and object-oriented programming. He received his B.Sc. degree in electrical engineering, and Ph.D. degree in computer science from FER in 2003 and 2010, respectively. He is affiliated with the Department of Telecommunications at FER from 2003. As a part of his Ph.D. studies, he spent academic year 2006-2007 at the Distributed Information Systems Laboratory at École Polytechnique Fédérale de Lausanne, Switzerland, as a scholarship holder of the Swiss Government scholarship for university, fine arts and music schools for foreign students. He was promoted to Full Professor in July 2022. He has participated in a number of research projects funded by national sources and EU funds and is currently leading the Data Streams Laboratory at FER. He is currently participating in the Centre of Research Excellence for Data Science and Advanced Cooperative Systems, which is the first national centre of excellence in the field of technical sciences in Croatia. He has co-authored more than 40 scientific journal and conference papers in the area of large-scale distributed systems, Internet of Things and Big data processing. He is a member of IEEE.

Sažetak

Sustav objavi-pretplati je raspodijeljeni komunikacijski sustav u kojem se proizvođači poruka nazivaju objavljivači te objavljuju poruke (objave) koje žele isporučiti potrošačima poruka, koji se nazivaju pretplatnicima. Pretplatnici se moraju unaprijed pretplatiti na poruke od interesa kako bi ih mogli primiti nakon objavljivanja. Ovaj doktorski rad se bavi posebnom vrstom sustava objavi-pretplati, a to su geoprostorni sustavi objavi-pretplati u kojima i objave i pretplate sadrže geoprostorni objekt. Takav objekt se u objavama koristi za izražavanje informacija o njihovoj lokaciji, a u pretplatama za izražavanje lokacije interesa. Postoji veliki potencijal za korištenje geoprostornih sustava objavi-pretplati u Internetu stvari te na senzorskom Webu. Međutim postojeći geoprostorni sustavi objavi-pretplati nisu primjenjivi u tu svrhu jer su centralizirani i ne mogu se nositi s tokovima geoprostornih podataka koji imaju visoku učestalost objavljivanja. Kako bi se prevladalo ovo ograničenje, u doktorskom radu se predstavlja raspodijeljeni geoprostorni sustav objavi-pretplati za grozd računala koji u stvarnom vremenu učinkovito uspoređuje dolazne objave sa skupom pohranjenih pretplata. Osim toga, predložena su četiri različita algoritma uparivanja objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati. Također su predstavljeni rezultati opsežne eksperimentalne studije u kojoj se uspoređuje propusnost, kašnjenje i potrošnja memorije ovih algoritama. Ovi rezultati jasno pokazuju da su predloženi algoritmi učinkoviti i skalabilni na veće grozdove. Usporedba s najsuvremenijim centraliziranim pristupima pokazuje da su dodatni troškovi obrade predloženih raspodijeljenih algoritama koje uvodi platforma Apache Spark gotovo zanemarivi.

Ključne riječi: geoprostorni podaci, particioniranje, replikacija podataka, veliki skupovi podataka, obrada tokova podataka

Distributed geospatial publish/subscribe system with high publication frequency

In our modern information-oriented world, we are witnessing a dramatic transition from the age of a single connected device per person into the age of the Internet of Things (IoT) where there is a multitude of connected devices per person. Many of these devices are used as sensors which continuously generate a specific type of streaming sensor data which is usually highly frequent, mostly small in size and generally includes a location information. The examples of such devices are IoT sensors, smartphones when used for mobile crowdsourcing applications and static sensors within measurement stations.

There are numerous examples of smart IoT applications in which sensor readings are automatically processed and analyzed by consumer components in real-time. Such a consumer component is usually interested only in a subset of published sensor readings corresponding to a specific geographic location and specific content. For example, it can be interested only in temperature readings within a city district such that air quality readings from the same district or temperature readings from other districts are of no use for it. Therefore, to receive required sensor readings such a consumer should subscribe itself on a geographic location and content of its interest.

A publish/subscribe system (PS system) is typically used in practice when a component needs to be continuously subscribed to a subset of information content published by some other remote components. The most expressive type of PS systems are content-based PS systems (CBPS systems) which support defining the interest of a subscriber as a set of constraints on the content of publications, but usually do not support subscribing to a general location of interest. Therefore, a special type of CBPS systems have to be developed to deal with the problem of location-aware subscriptions to which we refer as the geospatial publish/subscribe system (GeoPS system).

Each publication in a GeoPS system should contain its geographical location. In order to support a general geographic location, we assume that the publication location is a geospatial object, i.e. either a point, polygon, linestring, multipoint, multipolygon, and multilinestring. While it is obvious that a geographic location can be a point since each GPS location is a multidimensional point, on the other hand it is probably not immediately clear how a location can be a polygon or linestring. However, instead of the exact GPS location of a publication, sometimes is only a less precise location information known, such as a city district or road, which can be represented as a polygon or linestring, respectively. Please note that such a less precise location information may be either a consequence of the sensor setup or location privacy protection using an obfuscation-based technique.

Analogous to publications, each subscription in a GeoPS system defines its geographic lo-

cation of interest. It is obvious that such a location will often be a polygon since it is expected that subscribed components would mainly be interested in sensor readings from certain areas. Similarly, it is probably clear that a location of interest can also be a linestring since a component can be interested in sensor readings from a road or river. Furthermore, if a component is interested only in readings of a specific sensor, its subscription location of interest will be a point. Therefore, we additionally assume that the subscription location is also a geospatial object.

Next we have to explain how geospatial objects from a subscription and publication can be matched. Usually, a spatial predicate is used for this purpose, which is a Boolean function that defines a required relationship of two spatial objects. In a GeoPS system, a spatial predicate is thus defined by each subscription to precisely specify in which locations is it interested. Finally, we assume that the spatial predicate of a subscription is one of the following: equals, disjoint, touches, crosses, within, overlaps, contains, intersects, covers, and coveredby.

The matching process between a publication and subscription in a GeoPS system consists of two phases: (1) geospatial objects from the publication and subscription have to be compared spatially using a spatial predicate from the subscription and (2) the content of the publication has to be compared against the content filter in the subscription, which is a set of various constraints on the publication content. When the number of subscriptions is small or when the frequency of incoming publications is low, each incoming publication can be sequentially matched with every subscription in real-time. In other cases, the throughput of the sequential matching will almost certainly be low and thus would increase the latency as publications will have to be queued up before the matching. Therefore, to improve the matching performance of a centralized GeoPS system, subscriptions are usually indexed within a special data structure, namely the spatial index. However, this does not fully solve the problem, but instead just postpones it until the number of subscriptions or the frequency of publications increases further.

The complete solution for the previously mentioned problem is in distributed matching of incoming publications and stored subscriptions, which requires replication or partitioning of subscriptions in addition to their spatial indexing. In this case, the process of matching is divided among worker nodes. If it is scalable, it will be able to efficiently process incoming publications in real-time by scaling itself (i.e., adding or removing worker nodes) to any publication frequency or number of subscriptions. Please note that if we replicate subscriptions over worker nodes we will able to divide the processing load among them, but we will have to keep an exact replica (i.e., a copy) of all subscriptions at every node. On the other hand, if we partition subscriptions over worker nodes we will have to keep only a partition of subscriptions at every node. However, the partitioning is expected to have worse processing performance than the replication since it introduces an additional processing overhead for determining both to which partition a publication belongs as well as which worker is responsible for that partition.

The original scientific contribution that has been achieved in the doctoral dissertation is as follows:

- •Formal model of a distributed geospatial publish-subscribe system (GeoPS)
- •Algorithms for matching publications and subscriptions in a distributed geospatial publish-subscribe system with high publishing frequency
- •Evaluation procedure for the proposed model and developed algorithms with respect to the characteristics of the distributed geospatial publish-subscribe system.

The rest of the dissertation is structured as follows.

An overview of related work is presented in Chapter 2. This dissertation is the first that proposes a distributed GeoPS system and thus we could not find any other paper directly considering this topic. However, other papers are related to our work since they discuss location-aware publish/subscribe systems, distributed PS systems, mobility-aware complex event processing systems, distributed processing of geospatial Big Data and distributed spatial indices. We give an overview of these highly related topics in Chapter 2.

In Chapter 3 we present geospatial objects and the relationships between them. As explained in the introductory part of the doctoral thesis, geospatial objects and their relationships are essential for the definition of geospatial publications and subscriptions.

In Chapter 4 we present spatial indices and partitioning methods we use in our distributed GeoPS system. Our distributed GeoPS system supports different variants of subscription replication and partitioning strategies. For this purpose, it can use three different spatial indices, namely the Quadtree, STR tree and HPR tree, and two different spatial partitioning methods, namely the Quadtree and KDB tree. A spatial index is required to efficiently identify candidates among stored subscriptions for each incoming publication. After identifying the candidate subscriptions, for each of them, we apply its spatial predicate on the incoming publication to check whether there is a match or not. If we did not use a spatial index, we would need to sequentially match each stored subscription with the incoming publication. Therefore, the spatial index reduces the necessary but costly spatial predicate calls. Our distributed GeoPS system requires a spatial partitioning method for an efficient distributed partitioning of subscriptions over worker nodes i.e., Spark executors. Such a method is required to efficiently identify partitions in which are located subscriptions that are potentially interested in a publication, while for the other partitions, we can be sure that they do not contain interested subscriptions. Please note that a spatial partitioning method is not required for an efficient implementation of the centralized GeoPS system.

In Chapter 5 we provide a formal specification of a GeoPS system by using set theory. Let $\boldsymbol{B}$ be a triple $(\boldsymbol{N},\boldsymbol{P},\boldsymbol{S})$, where $\boldsymbol{N}$ is a finite set of nodes, $\boldsymbol{P}$ is a finite set of geospatial publications, and $\boldsymbol{S}$ is a finite set of geospatial subscriptions in a GeoPS system. $\boldsymbol{B}$ gives the *structural view* of a GeoPS system and determines the boundaries of its state space. A node $n \in \boldsymbol{N}$ may publish

geospatial publications from $\boldsymbol{P}$, activate or cancel geospatial subscriptions from $\boldsymbol{S}$, and perform matching between active geospatial subscriptions and publications. Thus, a node assumes one or multiple roles: publisher, subscriber, and processing node. We refer to a node that activated a geospatial subscription as the subscriber (node), and analogously, a node that published a geospatial publication as its publisher (node). Let $\boldsymbol{N}$ be a finite set of nodes, let $\boldsymbol{P}$ be a finite set of geospatial publications, and let $\boldsymbol{S}$ be a finite set of geospatial subscriptions. We define a triple $\boldsymbol{B} \stackrel{\text{def}}{=} (\boldsymbol{N}, \boldsymbol{P}, \boldsymbol{S})$ as a GeoPS system if for each geospatial subscription $s \in \boldsymbol{S}$, this system delivers all matching geospatial publications to the subscriber of s, but each of them exactly once and immediately after its publishing, while it does not deliver a single non-matching geospatial publication to the subscriber of s. The definition specifies two important properties for real-time system specification: liveness and safety. The liveness property asserts that "all matching geospatial publications will eventually be delivered to their subscribers", while the safety property asserts that "matching publications will be delivered only once", while "non-matching geospatial publication will never be delivered".

In Chapter 6 we propose four different replication and partitioning algorithms for managing geospatial subscriptions in our distributed GeoPS system. We present each algorithm with its pseudocode. In addition, for each algorithm, the implementation model is formally verified concerning the logical satisfaction of the specification of the geospatial publish-subscribe system.

In Chapter 7 we propose the architecture of our distributed GeoPS system, four different replication and partitioning algorithms for managing geospatial subscriptions in our distributed GeoPS system and their implementation details. The architecture of our distributed GeoPS system is an Apache Spark Streaming job that runs in an Apache Hadoop YARN cluster. It is a distributed job that employs one of our replication and partitioning algorithms for managing geospatial subscriptions that we propose in Chapter 5. It processes an incoming data stream containing geospatial publications which it ingests from an Apache Kafka topic. For each subscriber, it filters publications from the incoming stream and creates a subscriber stream which contains only publications in which the subscriber is interested. Such a subscriber stream is published to a separate Kafka topic which can easily be consumed by the corresponding subscriber component, possibly an Apache Spark Streaming job. We present each algorithm with a pipeline of Spark DStream, Spark RDD and Java Streams operations which are performed on publications from the incoming geospatial data stream. For each operation, we show the type of incoming and outgoing objects and briefly explain how it is performed. The pipeline of each algorithm consists of two parts, namely the upper pipeline and the lower pipeline. The upper pipeline, in which we find subscriptions that are satisfied for incoming publications, is unique for each of the algorithms, while the lower pipeline is identical for all of them.

In Chapter 8 we present an experimental study to evaluate the throughput, latency and me-

mory consumption of our subscription replication and processing algorithms RIS, SPS, sPIS, RIsPS and RIhPS, as well as to compare them with competing centralized and distributed approaches. There are 20 different variants of our algorithms depending on different spatial indices and/or partitioning methods. The sPS algorithm has 2 variants since it does not index subscriptions. Similarly, RIhPS and RIS algorithms have 3 variants since they do not partition subscriptions. Finally, sPIS and RIsPS algorithms have 6 variants since they both index and partition subscriptions. We experimentally compare these variants to identify those that should be used in practice. The results of this comparison show that RIS_[HT], sPS_{QT}, sPIS_[QT]_{QT}, RIsPS_[HT]_{KT} and RIhPS_[HT] variants are among the best performing and thus we use them as representatives of our algorithms. We present the results of an extensive experimental evaluation in which we compare the throughput, latency, and memory consumption of these algorithms. These results clearly show that they are both efficient and scalable to larger clusters. The comparison with centralized state-of-the-art approaches shows that the processing overhead of our distributed algorithms introduced by the Apache Spark platform is almost negligible.

Finally, we give our conclusions and directions for future work in Chapter 9.

Keywords: geospatial data, partitioning, data replication, big data, data stream processing

Sadržaj

Poglavlje 1

Uvod

Tranzicija suvremenog informacijski usmjerenoga svijeta kojom iz razdoblja samo jednog uređaja povezanog na Internet po osobi prelazi se u razdoblje Interneta stvari (engl. *Internet of Things, IoT*) u kojem postoji mnoštvo povezanih uređaja. Mnogi od ovih uređaja su senzori koji generiraju specifičnu vrstu strujećih podataka koji su najčešće velike frekvencije i uglavnom male veličine, a uključuju geoprostornu komponentu (npr. geografsku širinu i dužinu, liniju, poligon). Napretkom tehnologije se na mnogim područjima svakodnevnog života generiraju velike količine informacija koje dolaze iz raznih izvora, što dovodi do potrebe za bržom i jednostavnijom razmjenom informacija. Brzo pružanje relevantnih informacija vrlo je traženo od strane korisnika kako bi im pomoglo da donesu brze i pravovremene odluke u stvarnom vremenu [1]. Da bi dobili određene informacije, klijenti moraju podnijeti zahtjev tj. upit, a zatim će zaprimiti odgovor koji sadrži informacije upućene od poslužitelja. Različite situacije u različitim okolnostima zahtijevaju prikupljanje podataka iz različitih izvora. Jako često informacije se primaju u obliku događaja na određenoj lokaciji što pruža dodatni izazov u obradi podataka i pružanju relevantnih informacija klijentima. Sam napredak tehnologije vidljiv je i na senzorskim uređajima, koji posjeduju napredne mogućnosti komunikacije s ostalim senzorima te generiraju određenu vrstu strujećih podataka koji su obično vrlo visoke frekvencije objavljivanja, ali uglavnom male veličine. U današnje vrijeme senzorski uređaji su svuda npr. mobilni uređaji koji mogu sadržavati više senzora, kao što su: GPS prijemnik, kamera, mikrofon, kompas, itd.. Mobilni uređaj može u isto vrijeme biti i proizvođač i konzument informacija. Veliki broj senzorskih očitanja ima i geoprostornu komponentu, koja je jako bitna u obradi podataka i pronalaženju zainteresiranih konzumenata.

Sustavi objavi-pretplati učinkovitim algoritmima usporedbe objava s pretplatama omogućuju dostavu relevantnih senzorskih očitanja do raspodijeljenog skupa krajnjih konzumenata u gotovo stvarnom vremenu. Sustav objavi-pretplati se sastoji od niza čvorova koji su raspodijeljeni na Internetu te definira postojanje dvije vrste krajnjih čvorova koji međusobno razmjenjuju poruke preko posrednika (tj. jednog ili više poslužitelja). Prva vrsta čvrova su objavljivači (engl.

publishers) koji stvaraju objave, a druga su pretplatnici (engl. *subscribers*) koji primaju objave od interesa [2]. Važno je naglasiti da objavljivači i pretplatnici nikada ne komuniciraju izravno jedni s drugima i jedni ne znaju za postojanje drugih, već umjesto toga komuniciraju isključivo preko posredničkih poslužitelja. Ovo razdvajanje je poželjna karakteristika za komunikacijski sustav jer su aplikacije neovisne o mogućim komunikacijskim problemima, a izbjegavaju se aspekti poput sinkronizacije ili neposredne komunikacije pretplatnika i objavljivača. Interakcija u sustavu objavi-pretplati odvija se kroz niz sljedećih osnovnih operacija koje se mogu izvršiti od strane objavljivača ili pretplatnika i obrnuto:

- spajanje (engl. *connect*) - spajanja pretplatnika, objavljivača ili posrednika u sustav objavi-pretplati

- odspajanje (engl. *disconnect*) - odspajanje pretplatnika, objavljivača ili posrednika iz sustava objavi-pretplati

- pretpla ćivanje (engl. *subscribe*) - iskazivanje interesa za određenim sadržajem koji će biti objavljen, pokreće ga pretplatnik

- otkazivanje pretplate (engl. *unsubscribe*) - otkazivanje pretplate na koju se pretplatnik prethodno pretplatio

- objavljivanje (engl. *publish*) - objavljivanje podataka u sustavu objavi-pretplati, pokreće ga objavljivač

- otkazivanje objave (engl. *unpublish*) - povlačenje prethodno objavljene objave iz sustava

- obavještavanje pretplatnika (engl. *notify*) - prosljeđivanje obavijesti zainteresiranom pretplatniku

Objavljivač objavljuje informaciju (tj. objavu ili događaj) koja je najčešće strukturirana kao skup parova atribut-vrijednost, gdje svaki atribut ima ime i tip podatka. Na strani pretplatnika, interes za određene objave izražava se putem pretplate. Sama pretplata je filter dijela sadržaja (ili cijele) objave izražen kroz niz ograničenja (na parove atribut-vrijednost u objavama) koja ovise o jeziku koji podržavaju pretplate. Objava odgovara pretplati, ako zadovoljava sva definirana ograničenja njenih atributa. Različiti načini za iskazivanje interesa pretplatnika su doveli do različitih modela pretplata u sustavima objavi-pretplati. Najpopularniji modeli pretplata su: tematski model (engl. *topic-based*), sadržajni model (engl. *content-based*), model

po vrstama (engl. *type-based*) i XML model [3]. U geoprostornim sustavima objavi-pretplati, koji su tema ovog doktorskog rada, pretplate definiraju geoprostornu komponentu i sadržaj koji objave moraju zadovoljavati pa stoga ovi sustavi podržavaju sadržajni model pretplata. U modelu raspodijeljenog sustava objavi-pretplati su identificirana 4 logička sloja: sloj mrežnih protokola (engl. *network protocols*), sloj prekrivajuće mreže (engl. *overlay infrastructure*), sloj za usmjeravanje (engl. *event routing*) i sloj za usporedbu (engl. *matching*) [3, 4, 5]. Ovaj doktorski rad prvenstveno se fokusira na sloj za usporedbu unutar kojeg se vrši usporedba između geoprostornih objava i pretplata s ciljem identificiranja objava koje u potpunosti zadovoljavaju određenu pretplatu.

1.1 Geoprostorni sustav objavi-pretplati

Senzorska očitanja uključuju i geoprostorni objekt (npr. točka, linija ili poligon) koji predstavlja zemljopisni položaj senzora. Pored geoprostornog objekta, svako očitanje senzora također najčešće uključuje sljedeće podatke: vremensku oznaku, ID senzora, vrstu senzora, vrstu čitanja i vrijednost čitanja. Postoje mnogobrojni primjeri aplikacija u kojima procesi, koji su konzumenti senzorskih očitanja, donose zaključke i odluke na osnovu primljenih senzorskih očitanja. Geoprostorna pretplata, koja u ovom doktorskom radu predstavlja iskaz interesa konzumenta senzorskih očitanja, se sastoji od geoprostornog objekta koji predstavlja geografsko područje interesa te dodatnih uvjeta koje senzorsko očitanje mora zadovoljavati (npr. vrsta očitanja). Na osnovu geografskog područja interesa i dodatnih uvjeta se vrši kontinuirana usporedba geoprostornih pretplata s nadolazećim geoprostornim objavama.

Geoprostorni događaji mogu biti korisna apstrakcija fenomena na nekoj geografskoj lokaciji ili dinamičke promjene stanja u geografskom prostoru. Npr. lokacija vozila, mjesto požara, meteorološko motrenje i vremensko širenje šumskih požara su neki primjeri dinamičkih geoprostornih informacija koje se mogu prikazati kao geoprostorni događaji. U kontekstu geoprostornih sustava objavi-pretplati, objavljivači su oni klijenti koji su izvor geoprostornih događaja, uključujući i geoprostorne promjene stanja tj. oni objavljuju geoprostorne objave koje bi mogle biti od interesa za druge klijente. Pretplatnici su oni klijenti koji se registriraju iskazujući svoje interese i žele biti obaviješteni o određenim geoprostornim objavama. Kako geoprostorne obavijesti sadrže atribute i prostorne podatake, pretplatnici izražavaju svoje interese, ne samo na sadržaj opisanih atributa, već i na prostor unutar kojeg se objavljuju geoprostorne obavijesti. Ova vrsta pretplate naziva se geoprostorna pretplata. Za definiranje takvih pretplata jezik mora podržavati izraze koji se koriste za dodjeljivanje prostornih ograničenja objavljenih geoprostornih obavijesti. Kao što je prethodno spomenuto, u geoprostornom sustavu objavi-pretplati jedna od najvažnijih informacija je lokacija, bilo da se radi o objavama ili pretplatama tj. svakoj pretplati i objavi pridružen je geoprostorni objekt, koji predstavlja područje na koje se odnosi

pretplata ili objava. Pri tome se geoprostorne objave i pretplate sastoje od dvije zasebne podatkovne komponente: prostorne komponente i komponente atributa. U tradicionalnom sustavu objavi-pretplati se koristi samo komponenta atributa te se usporedba između pretplata i objava vrši samo u atributnom prostoru. U geoprostornom sustavu objavi-preplati usporedbu je potrebno vršiti u oba prostora, tj. ne samo u atributnom prostoru, već i u geoprostoru. Objava će odgovarati pretplati ukoliko zadovoljava njene uvjete u oba prostora. Npr. pretplaćivanje na određene objave bez definiranja prostorne komponente, npr. o novim događajima iz svijeta sporta, predstavlja primjer pretplaćivanja u tradicionalnom sustavu objavi-pretplati jer se pri usporedbi provjeravaju samo vrijednosti atributa koji predstavljaju sadržaj objave. Ako se pretplatom želi, uz sadržaj od interesa, definirati i geoprostorno područje interesa, npr. pretplatiti se na nove događaje iz svijeta sporta na području Europe, tada se radi o geoprostornom sustavu objavi-pretplati. Pri tome se zadržavaju sve prednosti tradicionalnog sustava objavi-pretplati, ali se omogućuju i dodatne mogućnosti pretplaćivanja na geografsko područje od interesa.

Postoji više vrsta geoprostornih pretplata: pretplata na točku, na liniju i na područje. Točka reprezentira geometrijski aspekt objekta za koji je relevantan samo njegov položaj u prostoru, ali ne i njegova veličina. Linija, odnosno niz povezanih linija temeljna je apstrakcija za niz povezanih objekata u prostoru, dok je poligon apstrakcija za objekte koji se prostiru u dvodimenzionalnom prostoru na nekom području [6, 7, 8]. Ovaj doktorski rad fokusira se na geoprostorne sustave objavi-pretplati koji se temelje na spomenutoj usporedbi geoprostornih objava i pretplata.

1.2 Problematika istraživanja

Istraživanje se provodilo u četiri faze:

- *Modeliranje geoprostornih objekata i analiziranje postojećih centraliziranih algoritama usporedbe geoprostornih objekata i struktura podataka za njihovo indeksiranje:* U prvoj fazi istraživanja proučena su povezana istraživanja s naglaskom na vrste geoprostornih objekata i odnose između njih. Nakon toga formalno su modelirani geoprostorni objekti i njihovi odnosi te su analizirani centralizirani algoritmi usporedbe geoprostornih objekata. Osim toga, proučene su i analizirane prostorne strukture podataka koje su primjenjive na modelirane geoprostorne objekte i njihove odnose.

- *Formalni model geoprostornog sustava objavi-pretplati:* Na temelju rezultata prve faze istraživanja, definiran je formalni model geoprostornog sustava objavi-pretplati jer je za kasnije istraživanje bilo neophodno opisati entitete u geoprostornom sustavu objavi-pretplati te njihove međusobne odnose i interakciju. Osim toga, bilo je potrebno formalno

specificirati uvjete pod kojima objava mora biti isporučena pretplatniku te slučajeve u kojime ona to ne smije biti. Korištenjem teorije skupova definiran je formalni model koji uključuje entitete (pretplatnike i objavljivače), poruke (objave i pretplate) i njihov sadržaj te pravila rada sustava.

- *Algoritmi usporedbe u raspodijeljenom geoprostornom sustavu objavi-pretplati:* U trećoj fazi istraživanja istraženi su centralizirani algoritmi usporedbe u geoprostornim sustavima objavi-pretplati koji vremenski učinkovito pronalaze pretplate koje odgovaraju objavi, a temelje se na odgovarajućim prostornim strukturama podataka u kojima su indeksirane pretplate. Nakon toga razvijeni su algoritmi usporedbe za raspodijeljeni geoprostorni sustav objavi-pretplati s visokim intenzitetom objavljivanja u računalnom grozdu. Ovi algoritmi se temelje na tehnikama indeksiranja, replikacije i/ili particioniranja pretplata čime se ostvaruje raspodjela opterećenja ulaznog prometa (tj. nadolazećih objava) te se postiže vremenska učinkovitost i skalabilnost.

- *Postupak vrednovanja predloženog modela i eksperimentalna evaluacija razvijenih algoritama:* U četvrtoj fazi istraživanja vrednovan je predloženi model geoprostornog sustava objavi-pretplati te su eksperimentalno evaluirani centralizirani algoritmi usporedbe i razvijeni raspodijeljeni algoritmi usporedbe na stvarnim podacima iz odabranog studijskog slučaja. Kako različiti studijski slučajevi koriste različite geoprostorne objekte i odnose među njima, odabran je studijski slučaj koji pokriva većinu modeliranih geoprostornih objekata i njihovih odnosa. Centralizirani algoritmi su eksperimentalno evaluirani na odabranom poslužitelju pri čemu je varirana frekvencija objava, broj pretplata te različiti parametri algoritma. Tijekom evaluacije ponavljan je eksperiment radi mjerenja prosječne propusnosti, kašnjenja i memorijskog zauzeća. Raspodijeljeni algoritmi eksperimentalno su evaluirani u računalnom grozdu pri čemu je također varirana frekvencija objava, broj pretplata te različiti parametri razvijenog algoritma, kao i broj čvorova u računalnom grozdu radi evaluacije skalabilnosti. Tijekom vrednovanja također su ponavljani eksperimenti radi mjerenja prosječne propusnosti, kašnjenja i memorijskog zauzeća.

1.3 Ciljevi doktorskog rada

Doktorski rad obuhvaća istraživanja u području raspodijeljenih sustava objavi-pretplati, s naglaskom na geoprostorne objekte i odnose između njih. Zbog visokog intenziteta objavljivanja izazovno je usporediti dolazne geoprostorne objave s pohranjenim geoprostornim pretplatama u stvarnom vremenu. Zbog toga je nužno minimizirati vremensku složenost algoritama usporedbe u sljedećim koracima: pronalaženje (pretplata) kandidata u indeksu pretplata, usporedba

geoprostornog objekta dolazne objave s kandidatima te usporedba njenog preostalog sadržaja s odgovarajućim kandidatima tj. njihovim filterima. U doktorskom radu se analiziraju algoritmi usporedbe geoprostornih objekata i prostorne strukture podataka koje se koriste za njihovu pohranu. Predlaže se formalni model geoprostornog sustava objavi-pretplati. Razvijeni su vremenski učinkoviti raspodijeljeni algoritmi usporedbe u geoprostornim sustavima objavi-pretplati. Na odabranom studijskom slučaju verificiran je predloženi model i eksperimentalno su evaluirani razvijeni algoritmi usporedbe. Cilj istraživanja je bio formalno modelirati geoprostorni sustav objavi-pretplati te razviti i na studijskom slučaju eksperimentalno evaluirati centralizirane i raspodijeljene algoritme usporedbe u geoprostornom sustavu objavi-pretplati. Eksperimentalna evaluacija je pokazala da su raspodijeljeni algoritmi usporedbe skalabilni i učinkovitije obrađuju dolazni promet od centraliziranih algoritama.

1.4 Znanstveni doprinosi doktorskog rada

Znanstveni doprinosi doktorskog rada su:

- Formalni model raspodijeljenog geoprostornog sustava objavi-pretplati
- Algoritmi usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati s visokim intenzitetom objavljivanja
- Postupak vrednovanja predloženog modela i razvijenih algoritama s obzirom na obilježja raspodijeljenog geoprostornog sustava objavi-pretplati

1.5 Struktura doktorskog rada

Nakon uvodnog poglavlja koje opisuje motivaciju rada i definira ga pomoću postavljenih istraživačkih pitanja, u drugom poglavlju prikazuje se pregled srodnih istraživanja koji su povezani s temom istraživanja raspodijeljenih geoprostornih sustava objavi-pretplati te su isti predstavljeni u pet tematskih cjelina.

Treće poglavlje obuhvaća pregled geoprostornih objekata i njihovih odnosa. Opisani su tipični geoprostorni objekti, kao što su točka, linija i poligon te njihovi prostorni odnosi. Predstavljeni su prostorni odnosi i različiti modeli presjeka geoprostornih objekata.

Četvrto poglavlje predstavlja prostorne indekse i metode njihovog raspodjeljivanja koje koriste različite varijante predloženih algoritama usporedbe objava i pretplata. Koriste se tri različita prostorna indeksa (Quadtree, STR Tree i HPR Tree) te šest različitih metoda prostornog raspodjeljivanja (Hilbertovo particioniranje, particioniranje stablom STR Tree, particioniranje

stablom Quadtree, particioniranje stablom KDB Tree, particioniranje pravilnom rešetkom te particioniranje Voronoijevim dijagramima).

U petom poglavlju modeliran je raspodijeljeni geoprostorni sustav objavi-pretplati korištenjem teorije skupova.

U šestom poglavlju predložena su četiri različita algoritma usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu te je predstavljen njihov pseudokod.

U sedmom poglavlju opisana je implementacija raspodijeljenog geoprostornog sustava objavi-pretplati te su opisana četiri predložena algoritma usporedbe dolaznih objava i prijavljenih pretplata. Algoritmi su opisani na način kako su programski ostvareni za izvođenje na sustavu Apache Spark i srodnim tehnologijama.

U osmom poglavlju opisana je provedba eksperimentalnog vrednovanja propusnosti, kašnjenja i potrošnje memorije predloženih algoritama usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati. Prikazani su i analizirani dobiveni rezultati. Također, u ovom poglavlju su eksperimentalno vrednovane različite inačice predloženih algoritama s obzirom na vrstu prostornog indeksa i metodu prostornog raspodjeljivanja pretplata te su identificirane one s najvećim potencijalom za primjenu u praksi.

U devetom poglavlju donose se ukupni zaključci provedenog istraživanja te se predlažu koraci i smjernice za nastavak budućih istraživanja u području raspodijeljenih geoprostornih sustava objavi-pretplati s visokom učestalošću objavljivanja.

Poglavlje 2

Pregled srodnih istraživanja

U literaturi nisu pronađeni radovi koji se izravno bave raspodijeljenim geoprostornim sustavima objavi-pretplati, koji su tema ovog doktorskog rada, ali su pronađeni srodni radovi koji su povezani s ovom temom istraživanja. Ti radovi predstavljeni su u ovom poglavlju. Srodni radovi su podijeljeni u sljedećih pet tematskih cjelina: sustavi objavi-pretplati koji su svjesni lokacije, raspodijeljeni sustavi objavi-pretplati, sustavi za obradu složenih događaja koji su svjesni pokretljivosti, raspodijeljena obrada velikih geoprostornih podataka te raspodijeljeni prostorni indeksi.

2.1 Sustav objavi-pretplati koji su svjesni lokacije

Tradicionalni sustavi objavi-pretplati koji se temelje na sadržaju (engl. *Content Based Publish/Subscribe systems, CBPS systems*) općenito ne uzimaju u obzir geoprostorne informacije. Međutim, postoji nekoliko takvih sustava koji su svjesni lokacije [43, 82, 83, 115, 116, 117, 118]. Sustav objavi-pretplati predstavljen u [82] povremeno prima informacije o točnoj lokaciji objavljivača i pretplatnika koje se koriste pri usporedbi objava s pretplatama pomoću lokacijskog ograničenja (engl. *location constraint*) u obliku geografske lokacije i minimalne udaljenosti od interesa. Sustav objavi-pretplati iz [115] podržava tri različita filtra objava u pretplatama tj. prema temi, sadržaju i blizini. Filter blizine (engl. *proximity filter*) definira geoprostorno područje unutar kojega su objavljene objave validne, a može ga postaviti objavljivač ili pretplatnik. U [116] je uveden lokacijski ovisan filter (engl. *location-dependent filter*) koji se koristi za pretplaćivanje na geoprostornu lokaciju ili područje. Poseban jezik za izražavanje filtra se koristi u [117] za definiranje pretplata koje podržavaju filtriranje objava prema kontekstu njihove lokacije. Dodavanje informacija o lokaciji za objave i pretplate je predloženo u [118] gdje se lokacija može izraziti fizički (po geografskim koordinatama ili području interesa) ili logički korištenjem simboličkih izraza. Prostorni sustav objavi-pretplati predstavljen u [83] podržava specificiranje lokacije unutar objave te izražavanje područja interesa za pret-

plate. Pri tome jezik za definiranje pretplata podržava dva prostorna predikata: unutar (engl. *within*) - za iskazivanje interesa za određeno područje i udaljenost (engl. *distance*) - za iskazivanje interesa za područje u krugu oko neke geografske lokacije. Međutim svi navedeni sustavi objavi-pretplati svjesni lokacije ne podržavaju općenite geoprostorne objekte kao što su poligoni ili polilinije. U radovima [43, 46] predstavljen je raspodijeljeni sustav objavi-pretplati koji je svjestan lokacije, a automatski aktivira/deaktivira pokretne objavljivače senzorskih očitanja kako bi se istovremeno zadovoljili zahtjevi pokrivenosti senzorima i kvalitete opažanja s ciljem smanjenja ukupne potrošnje energije objavljivača i pretplatnika. Međutim, ovaj sustav također ne podržava općenite geoprostorne objekte u objavama i pretplatama već samo točkaste objave te pretplate koje definiraju ćelije od interesa u sustavu MGRS [42].

Postoji nekoliko centraliziranih sustava objavi-pretplati koji koriste geoprostorne objekte za izražavanje lokacije objava i interesa pretplata [4, 17, 19, 30, 31, 48, 119, 120]. U [119] i [30] grupa autora predstavlja centralizirani geoprostorni sustav objavi-pretplati za dojavu požara u stvarnom vremenu. Prototipna implementacija u [119] koristi bazu podataka za pohranu pretplata dok se višerazinska rešetka koristi kao prostorni indeks u [30]. Model AHS (engl. *Aggregated Hierarchical Spatial*) je predstavljen u [4] i [17] kako bi se učinkovito odredilo odgovaraju li senzorska očitanja skupu unaprijed definiranih upita. Dok [4] predstavlja centralizirani sustav objavi-pretplati, [4] uvodi pristup za web koji se temelji na povlačenju (engl. *pull*) podataka. U [4] varijanta stabla Quadtree naziva se stablo LOST (engl. *loading spatio-temporal*) i koristi se za indeksiranje pretplata. Stablo Quadtree se koristi kao prostorni indeks pretplata i u [17] gdje je vremenska učinkovitost ovog indeksa dokazana eksperimentalnom evaluacijom. U [120] predstavljen je centralizirani geoprostorni sustav objavi-pretplati za praćenje izljeva nafte, ali autori ne navode vrstu prostornog indeksa koji koriste u implementaciji. Ista skupina autora je objavila i rad [19] u kojem koriste bazu podataka za indeksiranje pretplata, ali to čini njihov pristup neupotrebljivim za primjene u stvarnom vremenu.

2.2 Raspodijeljeni sustavi objavi-pretplati

Glavni istraživački problem vezan za raspodijeljene sustave objavi-pretplati je kako učinkovito usmjeriti poruke (tj. objave, pretplate, itd.) između čvorova. Ovi se sustavi mogu kategorizirati u tri kategorije ovisno o vrsti usmjeravanja poruka: preplavljivanje (engl. *flooding*) te selektivno (engl. *selective*) i epidemijsko (engl. *gossiping*) usmjeravanje [58].

Preplavljivanje je najjednostavniji tip usmjeravanja poruka, a rezultira time da su čvorovi preplavljeni objavama ili pretplatama dok se uspoređivanje događa kod izvornog, odnosno odredišnog čvora. Budući da preplavljivanje objavama nije skalabilno zbog broja razmijenjenih poruka, ono uglavnom služi kao referentna strategija usmjeravanja u eksperimentalnim evaluacijama, npr. [121]. Preplavljivanje pretplata rezultira velikim brojem razmijenjenih poruka u

slučaju čestih izmjena pretplata [122] te stoga nije široko prihvaćeno.

Selektivno usmjeravanje uključuje sljedeća dva pristupa usmjeravanju poruka: usmjeravanje filtriranjem i usporedba treće strane (engl. *rendezvous*). U oba pristupa pretplata se pohranjuje samo na nekom podskupu čvorova mreže, a isto vrijedi i za slanje svake objave. Ovaj pristup ima smisla jedino u slučaju kad je na objavu pretplaćen samo mali dio čvorova mreže. U slučaju da svaku objavu prima većina čvorova u mreži puno bolje rješenje je preplavljivanje objavama. Kod usmjeravanja filtriranjem objave se šalju samo onim čvorovima u mreži koji se nalaze na izravnom putu do nekog pretplatnika. Prvi put je predstavljen u [123] i nakon toga je primijenjen u mnogim drugim raspodijeljenim sustavima objavi-pretplati. Na ovaj način se vrlo brzo zaustavlja širenje objava koje nisu zanimljive većini čvorova u mreži. Može se reći da u ovom slučaju postoji povratni put od svakog pretplatnika do svakog potencijalnog objavljivača. Ukoliko su svi čvorovi u mreži potencijalni objavljivači, tada dolazi do preplavljivanja mreže pretplatama, ali s tom razlikom da čvorovi nemaju znanje o cjelokupnoj mreži, već samo o svojim susjedima. Ova vrsta usmjeravanja se najčešće primjenjuje u mreži s posrednicima koja je aciklična. Performanse ovog pristupa direktno ovise o topologiji mreže s posrednicima. Kod usmjeravanja s usporedbom treće strane postoje dvije funkcije kojima se pretplate ili objave preslikavaju na jedan ili više čvorova u mreži. Na ovaj način se istim čvorovima u mreži šalju pretplate i objave koje im potencijalno odgovaraju, a tada svaki takav čvor vrši usporedbu i slanje objava pretplatnicima. Ovaj pristup se najčešće primjenjuje u strukturiranoj mreži posrednika. Najveći nedostatak ovog pristupa je ograničena izražajnost pretplata uzrokovana navedenim funkcijama preslikavanja. Predložen je u [124] i od tada se koristi u mnogim drugim raspodijeljenim sustavima objavi-pretplati.

Epidemijsko usmjeravanje (engl. *gossiping*) je ne-deterministički tip usmjeravanja, ali je vrlo popularan u znanstvenoj zajednici zbog visoke pouzdanosti, robusnosti i samo-stabilizacije [58]. U ovom doktorskom radu predlažu se deterministički algoritmi particioniranja i replikacije pretplata koji nisu vezani za epidemijsko usmjeravanje.

2.3 Sustavi za obradu složenih događaja koji su svjesni pokretljivosti

Sustavi za obradu složenih događaja (engl. *Complex event processing, CEP)* razmatraju objekte dolaznog toka podataka kao događaje niske razine interesa koji se moraju filtrirati i kombinirati kako bi se otkrili određeni obrasci koji predstavljaju složene događaje više razine interesa. Takav složeni događaj više razine rezultat je obrade skupa događaja koji su kombinirani sa statističkim, logičkim ili vremenskim operatorima. U praksi je tok događaja predstavljen grafom (operatora) koji povezuje operatore jedne s drugima. Iako su se sustavi CEP razvili iz sustava objavi-pretplati oni su različiti od njih budući da sustavi objavi-pretplati tijekom procesa

usporedbe obično razmatraju svaku objavu odvojeno od ostalih. Bez obzira na ovu razliku, raspodijeljeni sustav objavi-pretplati i sustavi CEP imaju i sličnosti jer se sastoje od mreže čvorova koji međusobno komuniciraju kako bi zainteresiranim potrošačima pružili informacije od interesa. U sustavima CEP koji su svjesni poketljivosti interes potrošača mijenja se ovisno o njegovoj lokaciji te stoga ovi sustavi moraju podržavati jednog ili više operatera koji su svjesni pokretljivosti. Pregled izazova u sustavima CEP koji su svjesni pokretljivosti te rješenja ovih izazova su dana u [36]. Međutim, samo se nekoliko znanstvenih radova bavi sustavima CEP koji su svjesni pokretljivosti, a većina njih su rezultat rada iste istraživačke skupine. U sustavu CEP koji je svjestan pokretljivosti iz [34, 37] čvorovi su organizirani u prostorno-particioniranoj hijerarhiji na skalabilnom raspodijeljenom R-stablu (SD-Rtree), koje je predstavljeno u [35], tako da se svakom čvoru dodjeljuje prostorna regija za koju je zadužen kako bi se učinkovito obrađivali pokretni prostorni upiti raspona (engl. *moving spatial range queries*) potrošača u regiji. Međutim ovi se radovi uglavnom bave učinkovitom izvedbom re-konfiguracije dinamičkog grafa operatora nastalom uslijed promjene lokacije potrošača, dok se ovaj doktorski rad fokusira na učinkovitu usporedbu između prostornih objekata koji predstavljaju objave i pretplate.

2.4 Raspodijeljena obrada velikih geoprostornih podataka

Veliki geoprostorni podaci odnose se na skupove prostornih podataka koji premašuju kapacitet trenutnih računalnih sustava [125]. U [71] sugerira se da su volumen (engl. *volume*), raznolikost (engl. *variety*) i brzina (engl. *velocity*) tri dimenzije izazova u upravljanju velikim podacima. Kasnije je IBM dodao vjerodostojnost (engl. *veracity*) kao četvrtu dimenziju [126]. U posljednjem desetljeću razvijene su mnoge različite platforme otvorenog koda za rješavanje spomenutih izazova korištenjem grozda računala kao što su Apache Hadoop YARN (MapReduce2), Apache Spark, Apache Flink, Apache Storm, H20 itd. Usporedba ovih platformi dana je u [70]. Među navedenim platformama, platforme Apache Hadoop YARN [21] i Apache Spark [22, 23] su vjerojatno najpopularnije i to je glavni razlog zašto su privukle toliku pozornost vezanu uz geoprostornu obradu velikih podataka. Hadoop-GIS [66] bio je prvi okvir za upravljanje velikim geoprostornim podacima, a vrlo brzo su ga slijedili SpatialHadoop [69], GeoSpark (Apache Sedona) [67, 68], SpatialSpark [65], LocationSpark [63] i mnogi drugi. Ti su okviri privukli veliku pozornost među različitim istraživačkim skupinama [62]. Ipak većina literature smatra volumen najvažnijim izazovom pri upravljanju velikim geoprostornim podacima, budući da skupovi geoprostornih podataka mogu biti vrlo veliki. Ovaj doktorski rad se prvenstveno bavi brzinom (tj. izazovom obrade velikih geoprostornih podataka s visokom učestalošću objavljivanja u stvarnom vremenu) što je također prepoznato kao istraživački problem u nekim drugim radovima [52, 59, 60, 61].

2.5 Raspodijeljeni geoprostorni indeksi

U [20] su predložena dva raspodijeljena algoritma prostornog indeksiranja: centralizirani globalni indeks i hijerarhijski indeks na dvije razine. Centralizirani globalni indeks ne pohranjuje prostorne podatke već samo osnovne informacije potrebne za lociranje tih podatka koji su raspodijeljeni na čvorovima. Raspodijeljeni prostorni indeks koji se temelji na platformi Apache Storm predložen je u [52], ali je dizajniran za pokretne prostorne objekte dok se ovaj doktorski rad bavi statičkim pretplatama. Metoda za izgradnju raspodijeljenog Hilbertovog R-stabla predložena je u [50], dok [51] predlaže shemu višedimenzionalnog indeksiranja podataka koja koristi hijerarhijske strukture indeksiranja, pri čemu su obje metode namijenjene učinkovitoj obradi upita raspona (engl. *range queries*). U [49] predlaže se dvoslojni raspodijeljeni prostorni indeks koji koristi metodu Geohash za transformaciju geoprostorne lokacije u jednodimenzionalni *hashcode* koji se zatim koristi za particioniranje podataka u disjunktne pod-pravokutnike koji se dodaju u R-stablo. Međutim metoda Geohash se ne može koristiti za složenije geoprostorne objekte koji su tema ovog doktorskog rada. Arhitektura temeljena na više agenata za obradu kontinuiranih geoprostornih upita pokretnih objekata prikazana je u [44]. U praksi se ova arhitektura koristi za kontinuirano slanje upita višestrukim poslužiteljima geoprostornih baza podataka na kojima se pohranjuju podaci, ali je dizajnirana za pokretne prostorne upite (engl. *moving spatial queries*), dok se ovaj doktorski rad bavi statičkim pretplatama. U [35] i [33] su predložene implementacije raspodijeljenih stabala R-Tree i Quadtree, koje su dizajnirane za pohranu i postavljanje upita nad velikim skupovima prostornih podataka koji su pohranjeni na mnogo različitih čvorova. Ovi radovi se prvenstveno bave dinamičkim ažuriranjem stabala nakon umetanja i brisanja podataka. Međutim, pri tome više čvorova mora međusobno razmjenjivati poruke kako bi pronašli odgovor na postavljeni prostorni upit raspona (engl. *spatial range query*), što nije primjenjivo u kontekstu obrade u stvarnom vremenu kojom se bavi ovaj doktorski rad.

Poglavlje 3

Geoprostorni objekti i njihovi odnosi

U ovom poglavlju se definiraju geoprostorni objekti i odnosi između njih. Kao što je objašnjeno u uvodnom dijelu doktorskog rada, geoprostorni objekti i njihovi odnosi su bitni za definiranje geoprostornih objava i pretplata. One će biti formalno definirane u poglavlju 5 ovog doktorskog rada.

3.1 Geoprostorni objekti

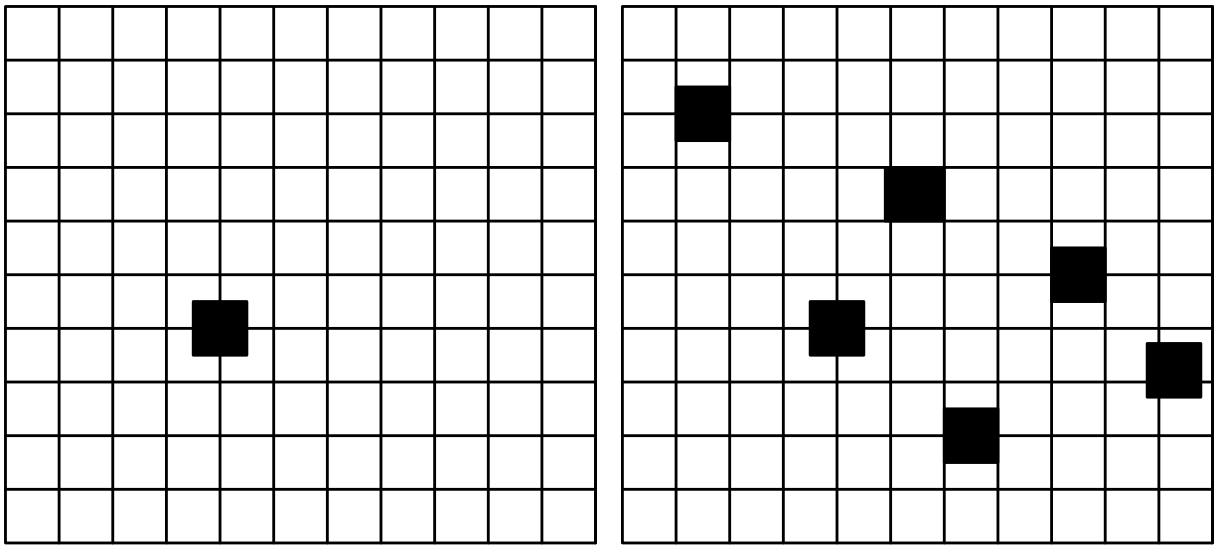
Pojam geoprostornog objekta označava područje u dvodimenzionalnom koordinatnom sustavu, koji predstavlja dio zemljine površine tj. područje interesa pretplate ili područje na koje se odnosi objava u geoprostornom sustavu objavi-pretplati. Geoprostorni objekti u prostoru definiraju se kao:

•To čkasti: točka (engl. *Point*) i više točaka (engl. *MultiPoint*),
•Linijski: linija (engl. *LineString*) i više linija (engl. *MultiLineString*) i
•Poligonski objekti: poligon (engl. *Polygon*) i više poligona (engl. *MultiPolygon*).

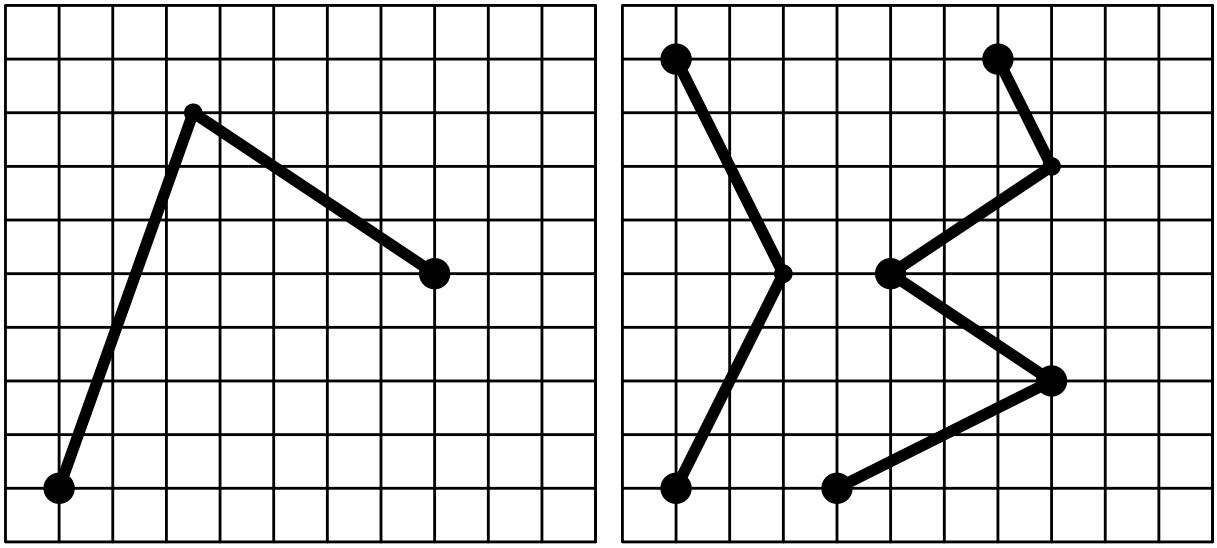
Kao što je prikazano na slici 3.1, točka (objekt tipa *Point*) je u prostoru određena s jednom koordinatom, odnosno s jednim uređenim parom x, y, dok se objekt tipa *MultiPoint* sastoji od jedne ili više zasebnih točaka koje su određene svaka svojim koordinatama x, y.

Kod skupa točaka međusobno povezanih dužinama (objekt tipa LineString) svaka točka tvori dužinu s prethodnom, a definirane točke ne zatvaraju prostor pa stoga ne čine poligon. Objekt tipa MultiLineString predstavlja skup više zasebnih objekata tipa LineString. Ove dvije vrste objekata su prikazana na slici 3.2.

Objekt tipa Polygon predstavlja ravninu omeđenu nizom dužina koje su pak kao i kod objekta tipa LineString određene nizom točaka. Pri tome je nužno da su prva i zadnja točka jednake kako bi ravnina bila omeđena. Pri definiranju poligona s rupom kao na slici 3.3 po-



Slika 3.1: Točkasti objekti

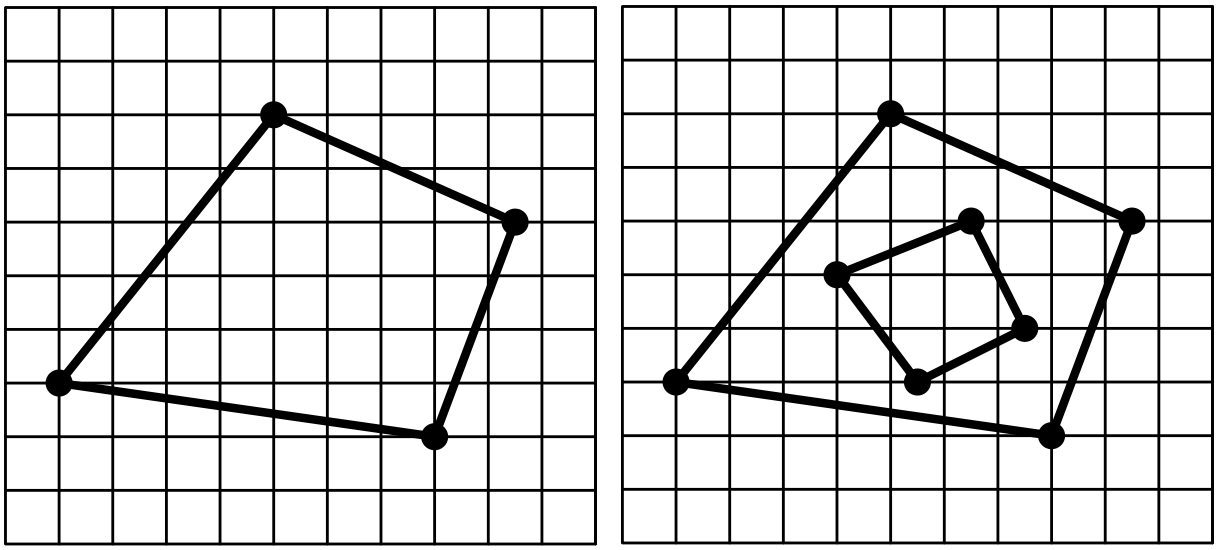


Slika 3.2: Linijski objekti

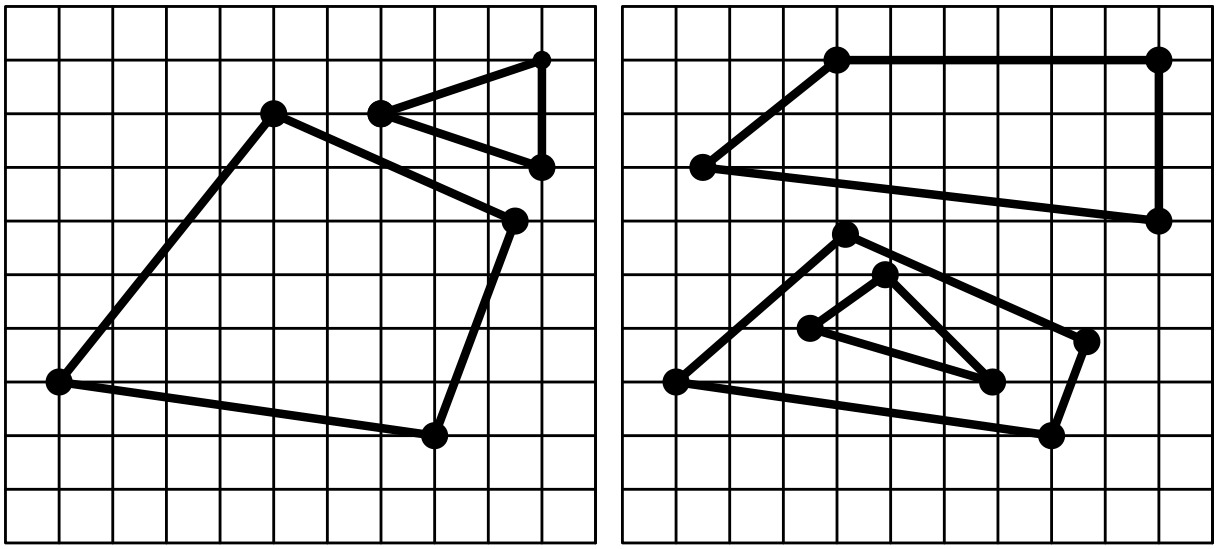
trebno je navesti dva polja koordinata od kojih jedno definira poligon, a drugo rupu u njemu. Objekt tipa MultiPolygon predstavlja skup više zasebnih poligona, kao što je prikazano na slici 3.4.

3.2 Prostorni odnosi i modeli presjeka geoprostornih objekata

Model geoprostornih odnosa se temelji na pojmovima topologije skupa točaka – unutrašnjosti, vanjština i granica. Prostorni model podataka, na osnovu kojeg se definiraju topološke relacije rabi algebarsku topologiju i zasnovan je na osnovnim geometrijskim objektima koji su definirani za različite dimenzije prostora [9]:



Slika 3.3: Poligonski objekti



Slika 3.4: Skup više poligona

- 0- ćelija je čvor (minimalni 0-dimenzionalni objekt)
- 1- ćelija je veza među dvije različite 0-ćelije
- 2- ćelija je površina opisana zatvorenim nizom od tri 1-ćelije koje se ne presijecaju

Pomoću prostornih predikata (engl. *spatial predicates*) moguće je utvrditi postojanje specifičnih prostornih odnosa između geoprostornih objekata. Temeljni pristup za usporedbu dva geoprostorna objekta čine testovi presjeka unutrašnjosti (engl. *interiors*), granica (engl. *boundaries*) i vanjština (engl. *exteriors*) te klasificiranje odnosa među njima. Klasificiranje odnosa među geoprostornim objektima postiže se na temelju vrijednosti rezultirajuće matrice presjeka [8, 10].

Postoji više modela koji se koriste za definiranje odnosa geoprostornih objekata [10]:

•4IM (engl. *Four Intersection Model*)

•9IM (engl. *Nine Intersection Model*)

•SNE-9IM (engl. *Separation Number Extended Nine Intersection Model*)

•DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*)

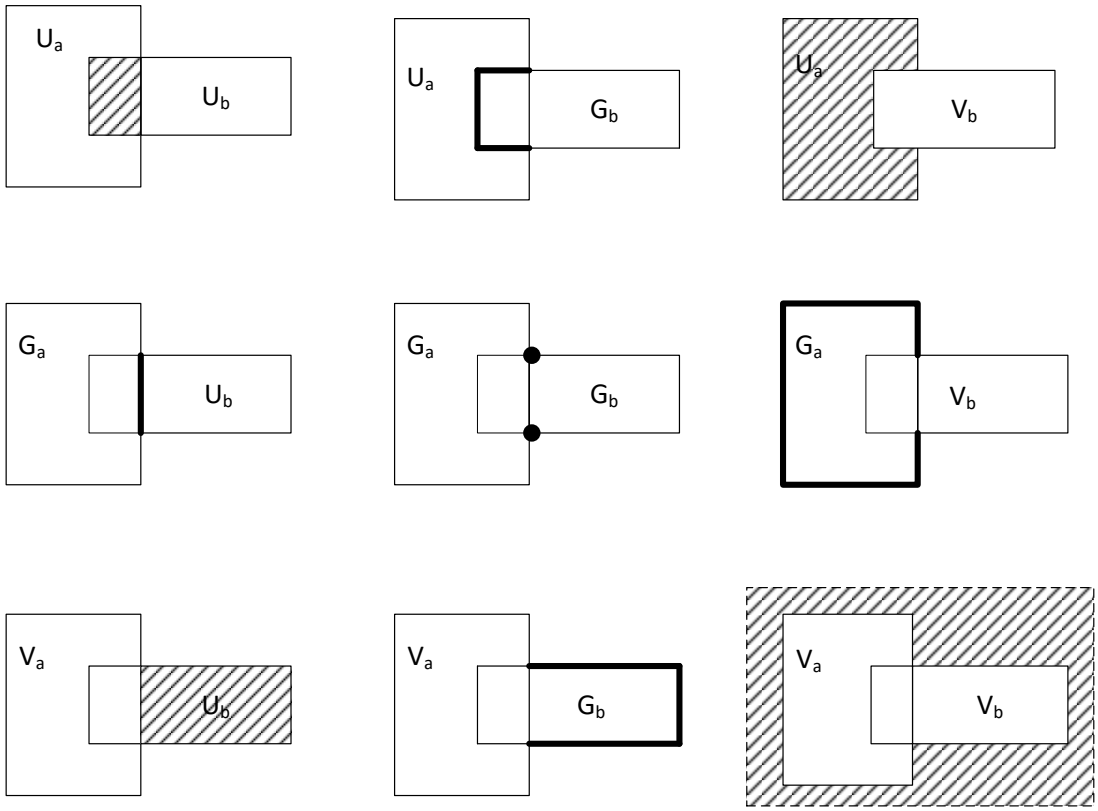
Svaki od navedenih modela presjeka temelji se na prihvaćenim definicijama granica, unutrašnjosti i vanjštine za osnovne tipove geometrijskih objekata koji se razmatraju. Stoga je prvi korak definiranje unutrašnjosti, granice i vanjštine uključenih tipova geometrije. Domena geometrijskih objekata koji se razmatraju su oni koji su topološki zatvoreni.

•Granica geometrijskog objekta je skup geometrija sljede će donje dimenzije.

•Unutrašnjost geometrijskog objekta sastoji se od onih to čaka koje su preostale (unutra) kad se uklone granične točke.

•Vanjština geometrijskog objekta sastoji se od to čaka koje nisu u unutrašnjosti ili granici.

Svaki od ovih modela opisuje odnos dva objekta na temelju matrice presjeka. 4IM (engl. *Four Intersection Model*) je boolov skup operacija koji u obzir uzima presjek između granice i vanjštine. Vrijednost u matrici može biti 0 ili 1, gdje 0 znači da se objekti ne križaju, a 1 znači da se objekti križaju. 9IM (engl. *Nine Intersection Model* uzima u obzir vanjštinu, unutrašnjost i granice geoprostornog objekta, gdje vrijednost 0 u matrici znači da se objekti ne križaju, a 1 znači da se objekti križaju. SNE-9IM (engl. *Separation Number Extended Nine Intersection Model*) također uzima u obzir vanjštinu, unutrašnjost i granice geoprostornog objekta. Vrijednost u matrici je 0 ili cijeli broj, gdje 0 znači da se objekti ne križaju, a cijeli broj predstavlja na koliko mjesta se objekti križaju (dodiruju, presijecaju). DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) uzima u obzir vanjštinu, unutrašnjost i granice geoprostornog objekta te pruža više detalja. Dodatni detalji pružaju ključne informacije u definiranju prostornih odnosa, stoga je u nastavku opisan DE-9IM model i prostorni predikati tj. odnose geoprostornih objekata. Usporedbom unutrašnjosti, granica i vanjštine objekta A, i unutrašnjosti, granica i vanjštine objekta B moguće je kombinirati tako da stvaraju devet fundamentalnih opisa topoloških relacija između dva objekta. To su presjeci:



Slika 3.5: Grafički prikaz rezultata matrice presjeka geoprostornih objekata A i B

1.unutrašnjost objekta A (U_a) s unutrašnjosti objekta B (U_b)

2.unutrašnjost objekta A (U_a) s granicom objekta B (G_b)

3.unutrašnjost objekta A (U_a) s vanjštinom objekta B (V_b)

4.granica objekta A (G_a) s unutrašnjosti objekta B (U_b)

5.granica objekta A (G_a) s granicom objekta B (G_b)

6.granica objekta A (G_a) s vanjštinom objekta B (V_b)

7.vanjština objekta A (V_a) s unutrašnjosti objekta B (U_b)

8.vanjština objekta A (V_a) s granicom objekta B (G_b)

9.vanjština objekta A (V_a) s vanjštinom objekta B (V_b)

Matrica presjeka prikazana na slici 3.5 sadrži skup od devet vrijednosti presjeka p. Vrijed-

nosti presjeka *p* može se izraziti kao skup čiji su elementi T, F, *, 0, 1, 2. Za vrijednost skupa *p* = 0 definira se nulta dimenzija tj. dva geoprostorna objekta se presjecaju, a presjek *p* predstavlja točku u prostoru. Za vrijednost skupa *p* = 1 definira se jednodimenzionalni rezultat tj. dva geoprostorna objekta se presjecaju, a presjek *p* predstavlja liniju u prostoru. Za vrijednost skupa *p*=2 definira se dvodimenzionalni rezultat tj. dva geoprostorna objekta se presjecaju, a presjek *p* predstavlja plohu u prostoru. Vrijednost skupa *p* = F (engl. *False*) znači da se dva objekta ne presijecaju u području koje se promatra u matrici presjeka, dok za *p* = T (engl. *True*) definira se da presjek *p* predstavlja rezultat različit od praznog skupa u matrici presjeka. Za definiranje prostornog odnosa i što učinkovitijeg rezultata obrade nije potrebna detaljna analiza za neki navedeni odnos u matrici presjeka jer je na određenim odnosima iz matrice presjeka odmah vidljivo u kakvom su prostornom odnosu dva geoprostorna objekta, pa se u matrici presjeka za presjeke koji nisu potrebni za definiranje odnosa koristiti *p*=*.

Prostorni predikati osiguravaju kompletno pokrivanje svih geoprostornih odnosa objekata i moguće ih je primijeniti nad objektima istih ili različitih dimenzija. Također, svaki predikat može biti izražen svojom matricom presjeka. Prostorne odnose između geoprostornih objekata predstavljaju sljedeći predikati [6, 7, 8]:

•Jednak (engl. *Equals*)

•Disjunktan (engl. *Disjoint*)

•Dodiruje (engl. *Touches*)

•Križa (engl. *Crosses*)

•Sadržan (engl. *Within*)

•Preklapa (engl. *Overlaps*)

•Sadrži (engl. *Contains*)

•Presijeca (engl. *Intersects*)

•Prekriva (engl. *Covers*)

•Prekriven (engl. *Covered By*)

U nastavku su navedene definicije za svaki od ovih devet predikata te su prikazani i u matrici modela DE-9IM.

3.2.1 Jednak

Kod prostornog predikata Jednak (engl. *Equals*) opisan je samo jedan slučaj tj. da su dva geoprostorna objekta jednaka ukoliko vrijedi:

$$Equals(A,B) \stackrel{\text{def}}{=}$$
$$(U_a \cap U_b) \neq \emptyset \wedge (U_a \cap V_b) = \emptyset \wedge (G_a \cap V_b) = \emptyset \wedge (V_a \cap U_b) = \emptyset \wedge (V_a \cap G_b) = \emptyset \quad (3.1)$$

Matrica presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za prostorni predikat Jednak definirana je u Tablici 3.1.

Tablica 3.1: Matrica presjeka za prostorni predikat Jednak

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	F
Granica objekta A	*	*	F
Vanjština objekta A	F	F	*

3.2.2 Disjunktan

Kod prostornog predikata Disjunktan (engl. *Disjoint*) može se javiti jedinstveni slučaj kada dva objekta nemaju nikakvih dodirnih točaka tj. dva geoprostorna objekta se razdvajaju ukoliko vrijedi da:

$$Disjoint(A,B) \stackrel{\text{def}}{=} (U_a \cap U_b) = \emptyset \wedge (U_a \cap G_b) = \emptyset \wedge (G_a \cap U_b) = \emptyset \wedge (G_a \cap B_b) = \emptyset \quad (3.2)$$

Matrica presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za prostorni predikat Disjunktan definirana je u Tablici 3.2.

3.2.3 Dodiruje

Kod prostornog predikata Dodiruje (engl. *Touches*) opisano je više slučajeva ovisno o kakvom tipu geoprostornog objekta se radi. Objekti imaju barem jednu graničnu točku zajedničku, ali

Tablica 3.2: Matrica presjeka za prostorni predikat Disjunktan

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	F	F	*
Granica objekta A	F	F	*
Vanjština objekta A	*	*	*

nemaju unutarnje točke zajedničke, pa se razlikuju tri tipa odnosa među geoprostornim objektima.

$$\begin{aligned} Touches(A,B) \stackrel{\text{def}}{=} & [(U_a \cap U_b) = \emptyset \wedge (G_a \cap U_b) \neq \emptyset] \vee \\ & [(U_a \cap U_b) = \emptyset \wedge (U_a \cap G_b) \neq \emptyset] \vee \\ & [(U_a \cap U_b) = \emptyset \wedge (G_a \cap G_b) \neq \emptyset] \end{aligned} \tag{3.3}$$

Matrice presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za tri tipa odnosa među geoprostornim objektima za prostorni predikat Dodiruje su definirana u Tablicama 3.3,3.4 i 3.5.

Tablica 3.3: Matrica presjeka za 1. tip odnosa kod prostornog predikata Dodiruje

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	F	*	*
Granica objekta A	T	*	*
Vanjština objekta A	*	*	*

Tablica 3.4: Matrica presjeka za 2. tip odnosa kod prostornog predikata Dodiruje

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	F	T	*
Granica objekta A	*	*	*
Vanjština objekta A	*	*	*

Tablica 3.5: Matrica presjeka za 3. tip odnosa kod prostornog predikata Dodiruje

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	F	*	*
Granica objekta A	*	T	*
Vanjština objekta A	*	*	*

3.2.4 Križa

Kod prostornog predikata Križa (engl. *Crosses*) opisana su dva slučaja ovisno o kakvom tipu geoprostornih objekta se radi.

$$Crosses(A,B) \stackrel{\text{def}}{=} [((U_a \cap U_b) \neq \emptyset \wedge (U_a \cap V_b) \neq \emptyset] \vee [U_a \cap U_b) \neq \emptyset \wedge (V_a \cap U_b) \neq \emptyset] \quad (3.4)$$

Matrice presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za dva tipa odnosa među geoprostornim objektima za prostorni predikat Križa su definirana u Tablicama 3.6 i 3.7.

Tablica 3.6: Matrica presjeka za 1. tip odnosa kod prostornog predikata Križa

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	T
Granica objekta A	*	*	*
Vanjština objekta A	*	*	*

Tablica 3.7: Matrica presjeka za 2. tip odnosa kod prostornog predikata Križa

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	*
Granica objekta A	*	*	*
Vanjština objekta A	T	*	*

3.2.5 Sadržan

Kod prostornog predikata Sadržan (engl. *Within*) opisani je jedinstveni slučaj kada je jedan objekt unutar drugog objekta. Geoprostorni objekt nalazi se unutar drugog geoprostornog

objekta ukoliko vrijedi da:

$$Within(A,B) \stackrel{\text{def}}{=} (U_a \cap U_b) \neq \emptyset \wedge (U_a \cap V_b) = \emptyset \wedge (G_a \cap V_b) = \emptyset \tag{3.5}$$

Matrica presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za prostorni predikat Sadržan definirana je u Tablici 3.8.

Tablica 3.8: Matrica presjeka za prostorni predikat Sadržan

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	F
Granica objekta A	*	*	F
Vanjština objekta A	*	*	*

3.2.6 Preklapa

Prostorni predikat Preklapa (engl. *Overlaps*) opisan je pomoću dva slučaja ovisno o kakvom tipu geoprostornih objekata se radi. Geoprostorni objekt preklapa drugi geoprostorni objekt ukoliko vrijedi da:

$$\begin{aligned} Overlaps(A,B) \stackrel{\text{def}}{=} &[(U_a \cap U_b) \neq \emptyset \wedge (U_a \cap V_b) \neq \emptyset \wedge (V_a \cap U_b) \neq \emptyset] \vee \\ &[(U_a \cap U_b) = 1 \wedge (U_a \cap V_b) \neq \emptyset \wedge (V_a \cap U_b) \neq \emptyset] \end{aligned} \tag{3.6}$$

Matrice presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za dva tipa odnosa među geoprostornim objektima za prostorni predikat Preklapa su definirana u Tablicama 3.9 i 3.10.

Tablica 3.9: Matrica presjeka za 1. tip odnosa kod prostornog predikata Preklapa

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	T
Granica objekta A	*	*	*
Vanjština objekta A	T	*	*

Tablica 3.10: Matrica presjeka za 2. tip odnosa kod prostornog predikata Preklapa

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	1	*	T
Granica objekta A	*	*	*
Vanjština objekta A	T	*	*

3.2.7 Sadrži

Prostorni predikat Sadrži (engl. *Contains*) opisan je jedinstveni slučajem kada jedan objekt sadrži drugi objekt. Primjećuje se sličnost sa prostornim predikatom Unutar (engl. *Within*) tj. da se radi o inverznom slučaju.

$$Contains(A,B) \stackrel{\text{def}}{=} (U_a \cap U_b) \neq \emptyset \wedge (V_a \cap U_b) = \emptyset \wedge (V_a \cap G_b) = \emptyset \tag{3.7}$$

Matrica presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za prostorni predikat Sadrži definirana je u Tablici 3.11.

Tablica 3.11: Matrica presjeka za prostorni predikat Sadrži

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	*
Granica objekta A	*	*	*
Vanjština objekta A	F	F	*

3.2.8 Presijeca

Prostorni predikat Presijeca (engl. *Intersects*) opisan je sa više slučajeva ovisno o kakvom tipu geoprostornog objekta se radi

$$Intersect(A,B) \stackrel{\text{def}}{=} [(U_a \cap U_b) \neq \emptyset] \vee [(U_a \cap G_b) \neq \emptyset] \vee [(G_a \cap U_b) \neq \emptyset] \vee [(G_a \cap G_b) \neq \emptyset] \tag{3.8}$$

Matrice presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za tipove odnosa među geoprostornim objektima za prostorni predikat Presijeca su definirana u Tablicama 3.12, 3.13, 3.14 i 3.15.

Tablica 3.12: Matrica presjeka za 1. tip odnosa kod prostornog predikata Presijeca

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	*
Granica objekta A	*	*	*
Vanjština objekta A	*	*	*

Tablica 3.13: Matrica presjeka za 2. tip odnosa kod prostornog predikata Presijeca

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	T	*
Granica objekta A	*	*	*
Vanjština objekta A	*	*	*

3.2.9 Prekriva

Prostorni predikata Prekriva (engl. *Covers*) opisan je sa više slučajeva ovisno o kakvom tipu geoprostornog objekta se radi.

$$\begin{aligned} Covers(A,B) \stackrel{\text{def}}{=} & [(U_a \cap U_b) \neq \emptyset \wedge (V_a \cap U_b) = \emptyset \wedge (V_a \cap G_b) = \emptyset] \vee \\ & [(U_a \cap G_b) \neq \emptyset \wedge (V_a \cap U_b) = \emptyset \wedge (V_a \cap G_b) = \emptyset] \vee \\ & [(G_a \cap G_b) \neq \emptyset \wedge (V_a \cap U_b) = \emptyset \wedge (V_a \cap G_b) = \emptyset] \vee \\ & [(G_a \cap G_b) \neq \emptyset \wedge (V_a \cap U_b) = \emptyset \wedge (V_a \cap G_b) = \emptyset] \end{aligned} \tag{3.9}$$

Matrice presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za tipove odnosa među geoprostornim objektima za prostorni predikat Prekriva su definirana u Tablicama 3.16, 3.17, 3.18 i 3.19.

Tablica 3.14: Matrica presjeka za 3. tip odnosa kod prostornog predikata Presijeca

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	*	*
Granica objekta A	T	*	*
Vanjština objekta A	*	*	*

Tablica 3.15: Matrica presjeka za 4. tip odnosa kod prostornog predikata Presijeca

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	*	*
Granica objekta A	*	T	*
Vanjština objekta A	*	*	*

Tablica 3.16: Matrica presjeka za 1. tip odnosa kod prostornog predikata Prekriva

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	*
Granica objekta A	*	*	*
Vanjština objekta A	F	F	*

3.2.10 Prekriven

Kod prostornog predikata Prekriven (engl. *Covered By*) opisano je više slučajeva ovisno o tipu geoprostornih objekta.

$$\begin{aligned} \textit{Covered by}(A,B) \stackrel{\text{def}}{=} & [(U_a \cap U_b) \neq \emptyset \wedge (U_a \cap V_b) = \emptyset \wedge (G_a \cap V_b) = \emptyset] \vee \\ & [(U_a \cap G_b) \neq \emptyset \wedge (U_a \cap V_b) = \emptyset \wedge (G_a \cap V_b) = \emptyset] \vee \\ & [(G_a \cap U_b) \neq \emptyset \wedge (U_a \cap V_b) = \emptyset \wedge (G_a \cap V_b) = \emptyset] \vee \\ & [(G_a \cap G_b) \neq \emptyset \wedge (U_a \cap V_b) = \emptyset \wedge (G_a \cap V_b) = \emptyset] \end{aligned} \tag{3.10}$$

Matrice presjeka modela DE-9IM (engl. *Dimensionally Extended Nine Intersection Model*) za dva tipa odnosa među geoprostornim objektima za prostorni predikat Prekriven su definirane u Tablicama 3.20 i 3.21, 3.22 i 3.23.

Tablica 3.17: Matrica presjeka za 2. tip odnosa kod prostornog predikata Prekriva

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	T	*
Granica objekta A	*	*	*
Vanjština objekta A	F	F	*

Tablica 3.18: Matrica presjeka za 3. tip odnosa kod prostornog predikata Prekriva

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	*	*
Granica objekta A	T	*	*
Vanjština objekta A	F	F	*

Tablica 3.19: Matrica presjeka za 4. tip odnosa kod prostornog predikata Prekriva

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	*	*
Granica objekta A	*	T	*
Vanjština objekta A	F	F	*

Tablica 3.20: Matrica presjeka za 1. tip odnosa kod prostornog predikata Prekriven

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	T	*	F
Granica objekta A	*	*	F
Vanjština objekta A	*	*	*

Tablica 3.21: Matrica presjeka za 2. tip odnosa kod prostornog predikata Prekriven

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	T	F
Granica objekta A	*	*	F
Vanjština objekta A	*	*	*

Tablica 3.22: Matrica presjeka za 3. tip odnosa kod prostornog predikata Prekriven

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	*	F
Granica objekta A	T	*	F
Vanjština objekta A	*	*	*

Tablica 3.23: Matrica presjeka za 4. tip odnosa kod prostornog predikata Prekriven

	Unutrašnjost objekta B	Granica objekta B	Vanjština objekta B
Unutrašnjost objekta A	*	*	F
Granica objekta A	*	T	F
Vanjština objekta A	*	*	*

Poglavlje 4

Prostorni indeksi i metode particioniranja prostornih podataka

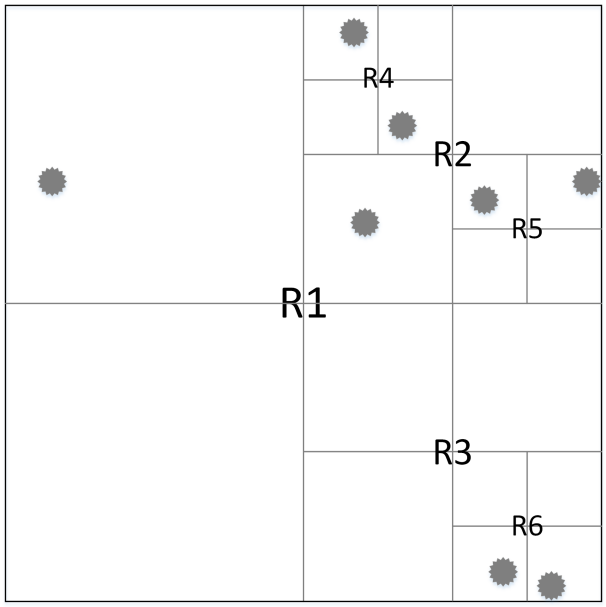
Raspodijeljeni geoprostorni sustav objavi-pretplati (engl. *Geospatial Publish/Pubscribe system*) koji je tema ovog doktorskog rada, podržava različite algoritme usporedbe za vremenski učinkovito pronalaženje pretplata koje odgovaraju objavi. Ovi algoritmi će biti predstavljeni u poglavlju 6 ovog doktorskog rada. Ovisno o korištenom prostornom indeksu i metodi particioniranja, razlikujemo različite varijante algoritama usporedbe. U eksperimentalnoj evaluaciji varijanti algoritama usporedbe u poglavlju 8 korišteni su prostorni indeksi QuadTree, STR Tree i HPR Tree te metode prostornog particioniranja QuadTree, STR Tree, KDB Tree, Hilbertova, Voronoijeva i pravilna rešetka (engl. *equal grid*). U nastavku ovog poglavlja predstavljeni su navedeni prostorni indeksi i metode particioniranja.

4.1 Prostorni indeksi

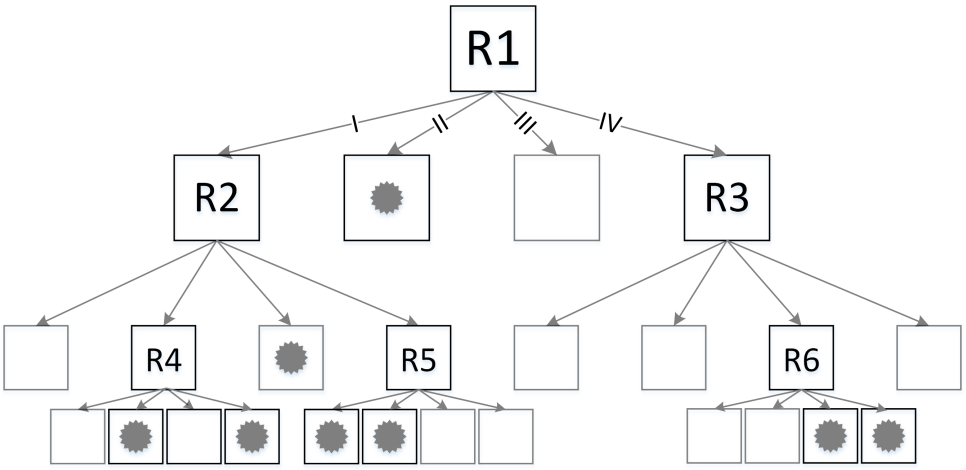
Prostorni indeks je potreban za učinkovito identificiranje kandidata među pohranjenim pretplatama za svaku dolaznu objavu. Nakon identificiranja pretplata kandidata za svaku od njih uzastopno se primjenjuje prostorni predikat na dolaznu objavu kako bi se provjerilo postoji li podudarnost ili ne. Ukoliko se ne bi koristio prostorni indeks trebalo bi se uzastopno provjeravati podudaranje za svaku pohranjenu pretplatu s dolaznom objavom. Stoga prostorni indeks smanjuje broj potrebnih poziva prostornih predikata čime se postiže učinkovitija obrada dolaznih objava.

Stablo QuadTree je hijerarhijska struktura podataka [11]. Ključna ideja ovog stabla je rekurzivno dekomponirati bilo koji dimenzionalni prostor, sve dok svaki prostorni objekt koji želimo pohraniti (npr. točka) ne dosegne ćeliju QuadTree-a, kao što je prikazano na slici 4.1. Iako se QuadTree može generalizirati na bilo koju dimenziju, u dvodimenzionalnom slučaju ono dijeli prostor na četiri kvadranta ili ćelije. QuadTree se rekurzivno gradi iz korijenske ćelije, tako da

prvo dijeli svoj prostor na četiri pod-ćelije, a zatim pohranjuje pokazivače na te podćelije, kao što je prikazana na slici 4.2. Isti postupak dijeljenja ćelije u 4 pod-ćelije ponavlja se rekurzivno sve dok se svi prostorni objekti ne mogu pohraniti u listove stabla. Upiti nad pohranjenim prostornim objektima u stablu QuadTree se izvršavaju rekurzivnim postupkom prelaska ćelija stabla koje se sijeku s predmetnim prostornim objektom. Postupak počinje na razini korijena stabla, a zatim se rekurzivno ponavlja na podrazine sve dok se ne dosegnu pohranjeni prostorni objekti u listovima stabla.



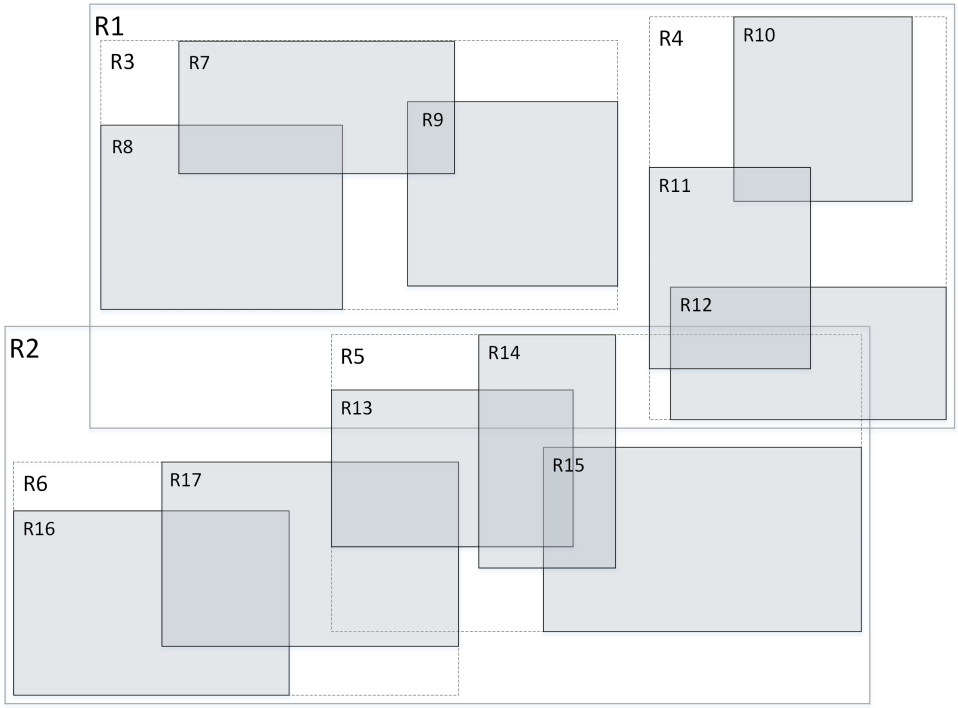
Slika 4.1: Primjer pohrane točaka u stablu QuadTree



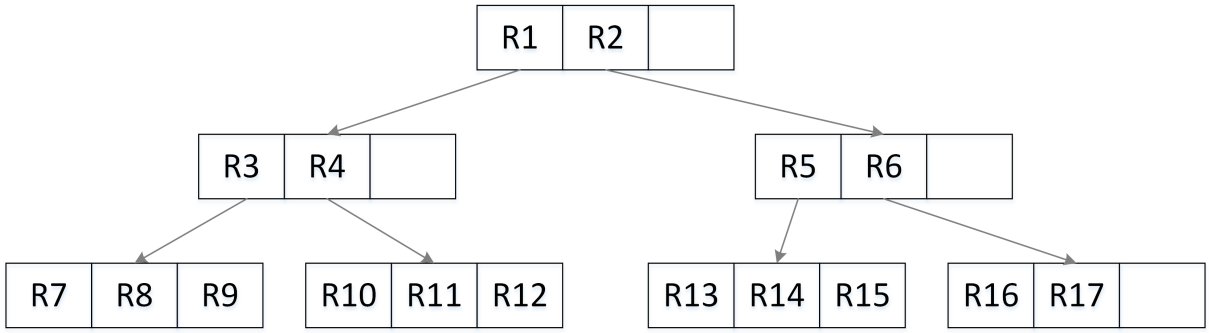
Slika 4.2: Primjer strukture stabla QuadTree

Stablo R-Tree je hijerarhijska struktura podataka koja se također koristi za prostorno indeksiranje [12]. Ključna ideja je grupirati obližnje prostorne objekte i predstaviti ih minimalnim

graničnim pravokutnikom (engl. *Minimum Boundary Rectangle, MBR*) u sljedećoj razini stabla, kao što je prikazana na slikama 4.3 i 4.4. Stablo R-Tree se također može generalizirati na bilo koju dimenziju. Rekurzivno se gradi iz korijenske ćelije, tako da grupira obližnje prostorne objekte i pohranjuje pokazivače na njihove MBR-ove. Isti postupak grupiranja prostornih objekata ponavlja se rekurzivno sve dok se svi prostorni objekti ne pohrane u stablo. Upiti nad pohranjenim prostornim objektima u stablu R-Tree se izvršavaju rekurzivnim postupkom prelaska MBR-ova koji se sijeku s predmetnim prostornim objektom. Postupak počinje na razini korijena i rekurzivno se ponavlja na podrazine dok se ne postigne najniža razina stabla. U raspodijeljenom geoprostornom sustavu objavi-pretplati koriste se sljedeće dvije verzije stabla R-Tree: STR Tree (engl. *Sort-Tile-Recursive Tree*) i HPR Tree (engl. *Hilbert-Packed R-Tree*). Obje su statične verzije stabla R-Tree koje ne podržavaju nova umetanja i brisanja nakon početnog učitavanja velikih količina podataka (engl. *bulk loading*). Međutim one pokušavaju konstruirati optimalniju strukturu stabla R-Tree s boljom iskorištenošću pohrane i manje preklapanja što rezultira poboljšanom izvedbom obrade upita. Ova stabla su konstruirana algoritmom pakiranja koji prvo sortira MBR-ove prostornih objekata s velikim opterećenjem, a zatim ih grupira prema redoslijedu sortiranja. Razlikuju se po načinu razvrstavanja MBR-ova. HPR Tree koristi Hilbertovu krivulju ispunjavanja prostora za sortiranje i grupiranje MBR-ova, dok STR Tree rekurzivno sortira MBR-ove po svakoj dimenziji, a zatim ih grupira u dijelove jednake veličine. Eksperimentalno su uspoređeni u radu [13], ali je zaključeno da niti jedno od ova dva stabla nije općenito bolje od drugog pa u praksi, za svaki studijski slučaj, treba provjeriti koje je od ovih stabala bolje .



Slika 4.3: Primjer MBR-ova u stablu R-Tree



Slika 4.4: Primjer strukture stabla R-Tree

Na kraju ovog potpoglavlja se definira općeniti prostorni indeks koji uključuje oba prethodno navedena prostorna indeksa.

Definicija 1 (Prostorni indeks)**.** Definira se prostorni indeks $\mathcal{I}$ koji se koristi za prostorno indeksiranje objekata $(o_1, o_2, \dots)$ njima pridijeljenim geoprostornim komponentama $(g_{o_1}, g_{o_2}, \dots)$ kao struktura podataka koja na upit postavljen s nekim geoprostornim objektom g učinkovito vraća skup $\mathcal{I}(g)$ objekata koji zadovoljava sljedeći uvjet:

$$\mathcal{I}(g) \subseteq \mathcal{I} : \nexists o_i \in \mathcal{I} \setminus \mathcal{I}(g) \wedge \mathit{Disjoint}(g, g_{o_i}) = \bot. \tag{4.1}$$

Drugim riječima, u razlici skupova $\mathcal{I} \setminus \mathcal{I}(g)$, koja sadrži objekate koji se nalaze u indeksu $\mathcal{I}$, a nisu vraćeni u skupu $\mathcal{I}(g)$, nema objekata koji nisu disjunktni s g. Stoga, skup $\mathcal{I}(g)$ sadrži sve objekte iz $\mathcal{I}$ čije geoprostorne komponente nisu disjunktne s g, ali uz njih može sadržavati

i neke objekte iz $\mathcal{I}$ čije geoprostorne komponente jesu disjunktne s g.

4.2 Metode particioniranja prostornih podataka

Raspodijeljeni geoprostorni sustav objavi-pretplati zahtijeva metodu prostornog particioniranja za učinkovito raspodijeljeno particioniranje pretplata na radnim čvorovima. Takva metoda je potrebna za učinkovito identificiranje particija u kojima se nalaze pretplate koje su potencijalno zainteresirane za objavu, dok za ostale particije možemo biti sigurni da ne sadrže zainteresirane pretplate. Prostorno particioniranje nije potrebno za učinkovitu implementaciju centraliziranog geoprostorog sustava objavi-pretplati.

Metode particioniranja QuadTree i STR Tree te Hilbertova metoda particioniranja se temelje na prethodno spomenutim prostornim indeksima QuadTree, KDB Tree i HPR Tree.

Metoda particioniranja KDB Tree se temelji na stabu KDB Tree (engl. *k-dimensional B-tree*) koje je hijerarhijska struktura podataka, a sastoji se od stranica. Služi za particioniranje k-dimenzionalnog prostora u svrhu optimizacije prostornih upita. Pri tome je to balansirano stablo s jednolikom udaljenošću svakog lista od korijena stabla. Svaki čvor stabla je zapravo stranica (engl. *page*) stoga se pristup čvorovima djeci može obaviti učinkovito tehnikom straničenja (engl. *paging*) [14]. Postoje dva tipa stranica:

- Stranice regije – unutarnji čvorovi stabla, sadrže pokazivače na druge stranice
- Stranice to čaka – listovi stabla, sadrže pokazivače na podatke

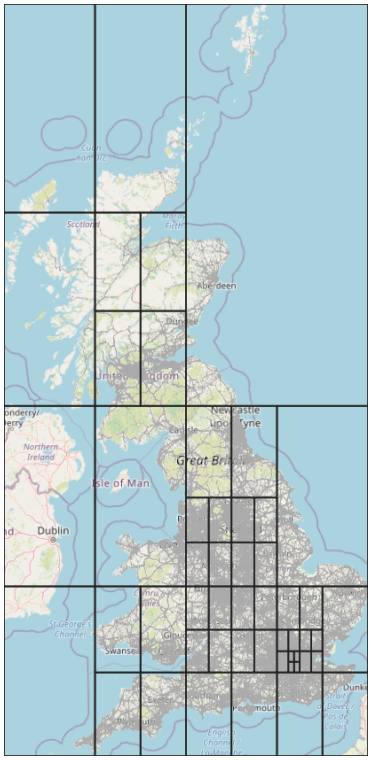
Stablo KDB Tree je slično stablu QuadTree jer također pohranjuje prostorne objekte samo u svojim listovima. Nadalje, također se rekurzivno gradi od korijenske stranice, ali dok QuadTree dijeli prostor na ćelije jednake veličine, KDB Tree umjesto toga dijeli prostor svake stranice koju presiječe neka ravnina na točno dvije pod-stranice.

Voronoijeva metoda prostornog particioniranja dijeli prostor na regije gdje je svako područje određeno središnjom točkom. Za svaku geoprostornu točku izračunava se udaljenost do svih središta, a objekt se zatim dodjeljuje najbližoj regiji.

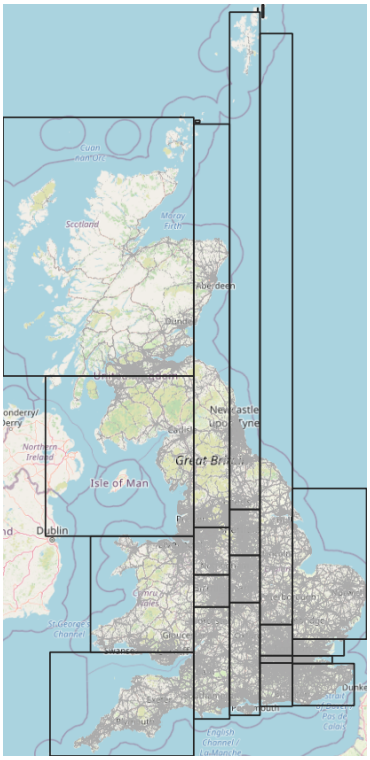
Metoda particioniranja pravilnom rešetkom (engl. *equal grid*) je jednostavnija od prethodno spomenutih metoda particioniranja, a prostor dijeli na pravokutnike jednake veličine.

Na slikama od 4.5 do 4.10 particioniran je skup podataka koji se kasnije koristi u eksperimentalnoj evaluaciji u poglavlju 8. Prostorne točke u ovom skupu podataka prikazane su sivom bojom, dok crni pravokutnici predstavljaju granice particija. Hilbertova i Voronoijeva metoda particioniranja proizvode particije koje se preklapaju, dok druge metode particioniranja imaju disjunktne particije.

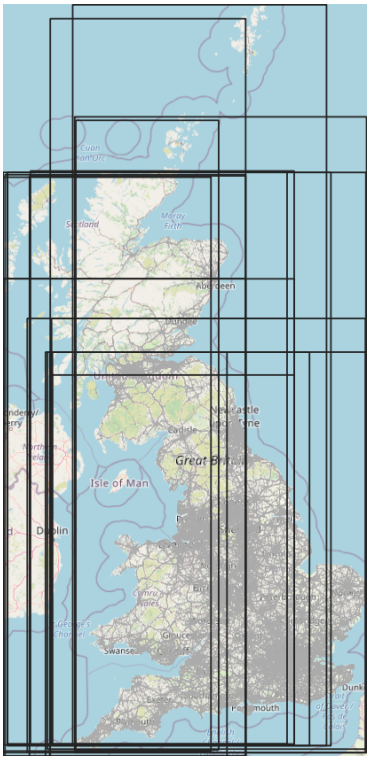
Na kraju ovog potpoglavlja se definira općenita metoda prostornog particioniranja koja



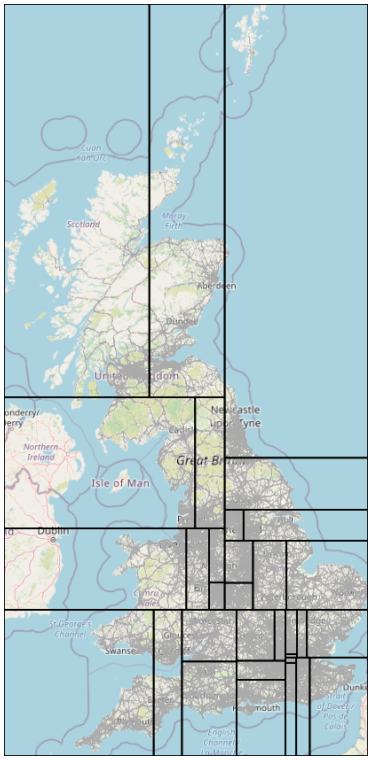
Slika 4.5: Metoda particioniranja Quad Tree



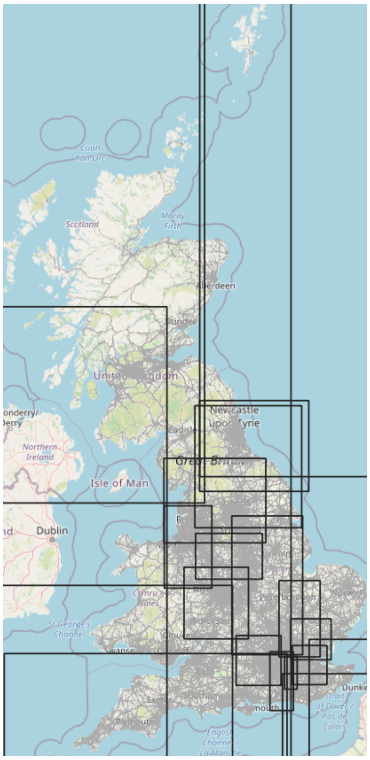
Slika 4.6: Metoda particioniranja STR Tree



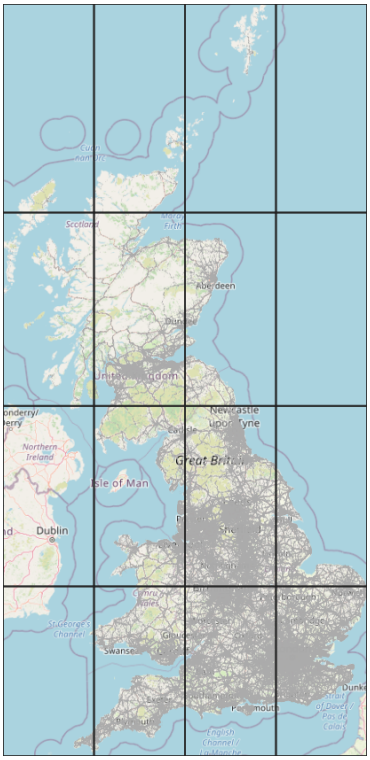
Slika 4.7: Hilbertova metoda particioniranja



Slika 4.8: Metoda particioniranja KDB Tree



Slika 4.9: Voronoijeva metoda particioniranja



Slika 4.10: Metoda particioniranja pravilnom rešetkom

uključuje svih šest prethodno navedenih metoda particioniranja.

Definicija 2 (Metoda prostornog particioniranja)**.** Neka je $\mathcal{A}$ neko geoprostorno područje od interesa predstavljeno geoprostornim objektom $g_{\mathcal{A}}$. Definira se metoda prostornog particioniranja $\mathcal{F}$ koja područje $\mathcal{A}$ particionira na skup $\boldsymbol{R} = \{R_1, R_2, \ldots, R_n\}$ od n particija i koja za neki geoprostorni objekt g vraća skup particija $\mathcal{F}(g) \subseteq \boldsymbol{R}$ kojima g pripada kako slijedi:

$$\mathcal{F}(g) \stackrel{\text{def}}{=} \{R_i \in \boldsymbol{R} : \mathit{Disjoint}(g, g_{R_i}) = \bot\} \text{ i} \tag{4.2}$$

$$\mathit{Covers}(\bigcup_{i=1}^{n} g_{R_i}, \mathcal{A}) = \top, \tag{4.3}$$

gdje je g_{R_i} geoprostorni objekt particije R_i koji predstavlja geoprostorno područje koje ona obuhvaća.

Bitno je napomenuti da izraz (4.2) definira da neki geoprostorni objekt g pripada particijama čiji geoprostorni objekti g_{R_i} nisu disjunktni s njim, dok izraz (4.3) definira da skup particija $\boldsymbol{R}$ potpuno prekriva područje $\mathcal{A}$ tako da ne postoji niti jedan dio ovog područja kojeg ne prekriva barem jadna particija iz $\boldsymbol{R}$.

Poglavlje 5

Model raspodijeljenog geoprostornog sustava objavi-pretplati

U ovom poglavlju formalno se specificira raspodijeljeni geoprostorni sustav objavi-pretplati korištenjem teorije skupova. Neka je $\boldsymbol{B}$ = ($\boldsymbol{N}$, $\boldsymbol{P}$, $\boldsymbol{S}$) uređena trojka u kojoj je N konačan skup čvorova, P je konačni skup geoprostornih objava i S je konačni skup geoprostornih pretplata u geoprostornom sustavu objavi-pretplati. $\boldsymbol{B}$ daje strukturni pogled na geoprostorni sustav objavi-pretplati i određuje njegove granice. Čvor $n \in \boldsymbol{N}$ može objaviti geoprostornu objavu iz $\boldsymbol{P}$, aktivirati ili otkazati geoprostornu pretplatu iz $\boldsymbol{S}$ te provesti usporedbu (engl. *matching*) između aktivnih geoprostornih pretplata i objava. Dakle, svaki čvor može imati jednu ili više sljedećih uloga: objavljivač, pretplatnik ili čvor za obradu. Referirat ćemo se na čvor koji je aktivirao neku geoprostornu pretplatu kao (čvor) pretplatnik i analogno, na čvor koji je objavio geoprostornu objavu kao (čvor) objavljivač.

5.1 Geoprostorna objava

Definicija 3 (Geoprostorna objava)**.** Neka je $n_p \in \boldsymbol{N}$ neki čvor, neka je t_p^A neki trenutak, neka je c_p neki geoprostorni objekti neka je c_p neki sadržaj objave. Definira se skup $p \stackrel{\text{def}}{=} (c_p, g_p, n_p, t_p^A) \in \boldsymbol{P}$ kao geoprostornu objavu sa sadržajem c_p i geoprostornim objektom g_p koju je objavio čvor n_p i koja ima pridruženo vrijeme objavljivanja t_p^A.

Pri tome se pretpostavlja da je geoprostorni objekt g_p jedno od sljedećeg: točka (engl. *Point*), poligon (engl. *Polygon*), linija (engl. *LineString*), više točaka (engl. *MultiPoint*), više poligona (engl. *MultiPolygon*) ili više linija (engl. *MultiLineString*). Vrijeme objavljivanja t_p^A označava trenutak kada se geoprostorna objava p pojavljuje u sustavu. Pretpostavlja se da je sadržaj geoprostorne objave statičan pa se njezin sadržaj i vrijeme objavljivanja ne mijenjaju nakon njezina ulaska u sustav. Nadalje se pretpostavlja da geoprostorne objave imaju dodijeljeno jedinstveno vrijeme t_p^A prilikom ulasku u sustav tako da se sve geoprostorne objave mogu

sortirati prema svom vremenu objavljivanja. Ova pretpostavka sigurno vrijedi pri obradi na jednom čvoru koji u tom slučaju može dodjeljivati jedinstvene vremenske oznake dolaznim geoprostorim objavama, dok ona ne vrijedi u raspodijeljenom okruženju. Ipak, u raspodijeljenom okruženju može se pretpostaviti da svaki čvor za obradu može dodijeliti jedinstvenu (implicitnu) vremensku oznaku svojim ulaznim geoprostornim objavama, dok vremenske oznake ne moraju biti jedinstvene u cijelom sustavu.

5.2 Geoprostorna pretplata

Definicija 4 (Geoprostorna pretplata)**.** Neka $n_s \in \boldsymbol{N}$ neki čvor, neka su t_s^A i t_s^C dva trenutka za koja vrijedi $t_s^A < t_s^C$, neka je $fltr_s$ neka funkcija filtra sadržaja, neka je g_s neki geoprostorni objekt te neka je $pred_s$ neki prostorni predikat. Definira se skup $s \stackrel{\text{def}}{=} (fltr_s, g_s, pred_s, n_s, t_s^A, t_s^C)$ kao geoprostrornu pretplatu s filterom sadržaja $fltr_s$, geoprostornim objektom g_s i prostornim predikatom $pred_s$ koju je aktivirao čvor n_s i koja je aktivna unutar perioda $(t_s^A, t_s^C]$.

Analogno geoprostornim objavama, u slučaju geoprostorne pretplate također se pretpostavlja da je geoprostorni objekt g_s točka, poligon, linija, više točaka, više linija ili više poligona [7] [15] [16]. Nadalje pretpostavlja se da je prostorni predikat $pred_s$ jedan od sljedećih predikata: jednak (engl. *equals*), dodiruje (engl. *touches*), križa (engl. *crosses*), sadržan (engl. *within*), preklapa (engl. *overlaps*), sadrži (engl. *contains*), presijeca (engl. *intersects*), prekriva (engl. *covers*) i prekriven (engl. *covered by*). Vrijeme aktiviranja t_s^A i vrijeme otkazivanja t_s^C označavaju trenutke u kojima je geoprostorna pretplata aktivirana i otkazana u sustavu. Stoga je geoprostorna pretplata aktivna unutar perioda $(t_s^A, t_s^C]$. Analogno geoprostornim objavama pretpostavlja se da su t_s^A i t_s^C implicitne vremenske oznake koje su dodijeljene od strane čvorova za obradu. Naravno, prijenos geoprostorne pretplate i objave na obradu čvorovima uvodi određena kašnjenja u odnosu na vrijeme stvaranja na izvornom čvoru. Ipak moguće je pružiti jamstva za ispravnost usporedbe geoprostornih objava toleriranjem ograničenog vremena širenja geoprostrorne pretplate i geoprostorne objave što je uobičajena praksa u raspodijeljenim sustavima. Dodatno, pretpostavlja se da su funkcije filtra sadržaja $fltr_s$ i prostornog predikata $pred_s$ vremenski neovisne Booleove funkcije takve da je prva funkcija geoprostornog sadržaja objave (tj. $fltr_s(c_p)$)), dok je druga funkcija geoprostornih objekata g_p i g_s (tj. $pred_s(g_s, g_p)$).

5.3 Usporedba objava s pretplatom

Definicija 5 (Usporedba objava s pretplatom)**.** Neka je $s \in \boldsymbol{S}$ geoprostorna pretplata i neka je t trenutak u vremenu kad je s aktivna, tj. $t_s^A < t \leq t_s^C$. Definira se skup geoprostornih objava koje zadovoljavaju pretplatu s kao $M_s(t) \subseteq \boldsymbol{P}$ kako slijedi:

$$M_s(t) \stackrel{\text{def}}{=} \{p : (p \in \boldsymbol{P}) \wedge (t_s^A < t_p^A \leq t \leq t_s^C) \wedge [match_s(p) = \top], \tag{5.1}$$

gdje je $match_s(p) = \{[fltr_s(c_p) = \top] \wedge [pred_s(g_s, g_p) = \top]\}$ funkcija usporedbe pretplate s.

Funkcija usporedbe $match_s$ vraća $\top$ za geoprostorne objave koje sadržajem zadovoljavaju pretplatu s, dok skup M_s objava koje zadovoljavaju pretplatu s u trenutku t obuhvaća sve objave koje zadovoljavaju s, a objavljene su između trenutka aktiviranja pretplate s i trenutka t. Pretpostavlja se da geoprostorne pretplate predstavljaju iskaz interesa za buduće geoprostorne objave, tj. one koje se pojavljuju nakon aktiviranja geoprostorne pretplate, i ne mogu predstavljati iskaz interesa na prošle geoprostorne objave, tj. one koje su objavljene prije aktiviranja geoprostorne pretplate. U nastavku se definira geoprostorni sustav objavi-pretplati.

5.4 Geoprostorni sustav objavi-pretplati

Definicija 6 (Geoprostorni sustav objavi-pretplati)**.** Neka je $\boldsymbol{N}$ konačni skup čvorova, neka je $\boldsymbol{P}$ konačni skup geoprostornih objava, i neka je $\boldsymbol{S}$ konačni skup geoprostornih pretplata. Definira se skup $\boldsymbol{B} = (\boldsymbol{N}, \boldsymbol{P}, \boldsymbol{S})$ kao geoprostorni sustav objavi-pretplati ako za svaku geoprostornu pretplatu $s \in \boldsymbol{S}$, ovaj sustav isporučuje pretplatnicima sve odgovarajuće geoprostorne objave s, ali svaku od njih točno jednom, dok ne isporučuje pretplatniku s niti jednu geoprostornu objavu koja ne zadovoljava njegovu pretplatu.

Ova definicija navodi tri važna svojstva za specifikaciju stvarnovremenskih sustava [17, 18, 19]: životnost (engl. *liveness*) i dvije sigurnosti (engl. *safety*). Životnost uvjetuje da će „sve geoprostorne objave koje zadovoljavaju pretplatu na kraju biti dostavljene pretplatniku". Dva sigurnosna svojstva zabranjuju isporuku objava koje ne bi trebale biti dostavljene pretplatniku, ali ništa ne pretpostavljaju o slučajevima kada je takva dostava potrebna. Prvo sigurnosno svojstvo je legalnost (engl. *legality*) koje uvjetuje da "geoprostorna objava koja ne zadovoljava pretplatu nikada neće biti isporučena pretplatniku". Drugo sigurnosno svojstvo je jedinstvenost (engl. *uniqueness*) koje uvjetuje da će "geoprostorna objava biti isporučena pretplatniku točno jednom".

Poglavlje 6

Algoritmi usporedbe u raspodijeljenom geoprostornom sustavu objavi-pretplati

U ovom poglavlju predlažu se četiri različita algoritma usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati te se predstavlja njihov pseudokod. Osim toga, za svaki algoritam se formalno verificira model implementacije u cilju provjere logičkog zadovoljavanja specifikacije geoprostornog sustava objavi-pretplati, koja je definirana u Definiciji 6. Zbog jasnoće prezentacije se u ovom poglavlju koriste sljedeće pretpostavke:

- skup aktivnih pretplata S^A je konstantan tijekom rada sustava, tj. pretplate se ne dodaju niti brišu iz ovog skupa,
- metode (prostornog i hash) particioniranja se ne mijenjanju (tj. dinamički prilagođavaju raspodjeli dolaznih objava) tijekom rada sustava i
- skup radnih čvorova N^W je konstantan tijekom rada sustava, tj. novi radni čvorovi ne dolaze u sustav niti ga postojeći napuštaju.

6.1 Replicirani indeks i pretplate

Prvi algoritam koji se predstavlja je **Replicirani indeks i pretplate** (engl. *Replicated Index and Subscriptions, RIS*). Ovo je osnovni algoritam koji replicira prostorni indeks koji sadrži pretplate na sve izvršitelje u grozdu. Stoga navedeni algoritam pripada kategoriji preplavljivanja pretplatama u raspodijeljenim sustavima objavi-pretplati. Ovaj algoritam je vrlo sličan centraliziranom pristupu u kojem je cijeli prostorni indeks pohranjen u memoriji na jednom čvoru. Očigledno ovaj algoritam nije prostorno učinkovit, jer pohranjuje cijeli prostorni indeks na svim radnim čvorovima. Kod ovog algoritma se svaka particija dolaznog toka podataka obrađuje kod drugog radnog čvora zato što se dolazni tok podataka particionira na radne čvorove. Stoga je to u osnovi raspodijeljena verzija paraleliziranog prostornog indeksa.

Algoritam 6.1 predstavlja pseudokod algoritma RIS. U inicijalizacijskom dijelu ovog algo-

ritma se prvo stvara prostorni indeks $\mathcal{I}$ (2. linija) i u njega dodaju sve pretplate $s_i \in \boldsymbol{S}^A$ (3. i 4. linija). U praksi se kao prostorni indeks koristi neki od prostornih indeksa predstavljenih u poglavlju 4.1. Neki od ovih prostornih indeksa se moraju izgraditi nakon što se u njih dodaju pretplate (5. linija). Nakon toga se prostorni indeks replicira na sve radne čvorove (6. i 7. linija).

```
1  inicijalizacija:
2      stvori prostorni indeks I u kojem će biti indeksirane pretplate
3      za svaku pretplatu s_i ∈ S^A
4          u indeks I indeksiraj s_i korištenjem njenog geoprostornog
               objekta g_{s_i}
5      izgradi indeks I ako je potrebno
6      za svaki radni čvor n_i ∈ N^W
7          repliciraj indeks I na čvor n_i
8  algoritam:
9      za svaku novoobjavljenu objavu p_i ∈ P
10         odaberi radni čvor n_i ∈ N^W koji će obraditi objavu p_i
11         proslijedi objavu p_i na čvor n_i
12         na čvoru n_i
13             u indeksu I pronađi skup S^C(p_i) pretplata kandidata za p_i
14             za svakog kandidata s_i ∈ S^C(p_i)
15                 ako objava p_i zadovoljava kandidata s_i
16                     proslijedi objavu p_i pretplatniku kandidata s_i
```

Algoritam 6.1: Pseudokod algoritma RIS

Nakon provedbe inicijalizacijskog dijela ovog algoritma, mogu se početi obrađivati dolazne objave (9. linija). Za svaku novoobjavljenu objavu p_i odabire se radni čvor n_i koji će ju obraditi (10. linija). Nakon što se objava p_i proslijedi na odabrani radni čvor n_i (11. linija), na njemu se postavlja upit nad repliciranim prostornim indeksom $\mathcal{I}$ koji pronalazi skup $\boldsymbol{S}^C(p_i) = \mathcal{I}(g_{p_i})$ pretplata kandidata za objavu p_i (13. linija), koji je definiran izrazom (4.2). Zatim se za svakog kandidata $s_i \in \boldsymbol{S}^C(p_i)$ provjerava zadovoljava li ga objava p_i (14. i 15. linija). Ako objava p_i zadovoljava kandidata s_i (tj. $match_{s_i}(p_i) = \top$) tada se p_i prosljeđuje pretplatniku kandidata s_i (16. linija).

Teorem 1. *Pseudokod algoritma RIS, predstavljen kao Algoritam 6.1, zadovoljava specifikaciju geoprostornog sustava objavi-pretplati iz Definicije 6.*

Dokaz. Potrebno je dokazati da Algoritam 6.1 zadovoljava svojstva životnosti, legalnosti i jedinstvenosti. Za dokazivanje svojstva životnosti pretpostavimo da postoji objava p_i koja nije isporučena pretplatniku pretplate s_i iako ju zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije

isporučena pretplatniku pretplate s_i. Da algoritam RIS ne bi isporučio pretplatniku od s_i objavu p_i, pretplata s_i se ne smije nalaziti u skupu pretplata kandidata $\boldsymbol{S}^C(p_i)$ za p_i koji se dohvaća i koristi u 13. i 14. liniji Algoritma 6.1. Međutim, indeks $\mathcal{I}$ po Definiciji 1 će pri pronalaženju skupa pretplata kandidata $\boldsymbol{S}^C(p_i) = \mathcal{I}(g_{p_i})$ za p_i u njemu vratiti svaku pretplatu koja je kandidat za p_i (tj. čiji geoprostorni objekt nije disjunktan s geoprostornim objektom od p_i) pa time i pretplatu s_i. Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva legalnosti pretpostavimo da postoji objava p_i isporučena pretplatniku pretplate s_i iako ju ne zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i. Međutim, iz 15. i 16. linije u Algoritmu 6.1 se vidi da za svaku objavu p_i koja je isporučenu pretplatniku pretplate s_i vrijedi $match_{s_i}(p_i) = \top$. Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva jedinstvenosti pretpostavimo da $\exists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput. Da bi algoritam RIS isporučio pretplatniku od s_i objavu p_i više od jednom, pretplata s_i se mora ponavljati u skupu pretplata kandidata $\boldsymbol{S}^C(p_i)$ za p_i u 13. i 14. liniji Algoritma 6.1 ili se objava p_i mora obrađivati na više od jednog radnog čvora $n_i \in \boldsymbol{N}^W$ u 12.-17. liniji Algoritma 6.1. Međutim, skup pretplata kandidata $\boldsymbol{S}^C(p_i)$ po svojoj definiciji ne sadrži duplikate, a u 10. i 11. liniji Algoritma 6.1 se vidi da on odabire samo jedan radni čvor n_i na kojem će se obraditi objava p_i. Stoga $\nexists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput što je u kontradikciji s pretpostavljenim. ■

6.2 Prostorno particionirane pretplate

Prvi predloženi algoritam particioniranja pretplata je **Prostorno particionirane pretplate** (engl. *spatially Partitioned Subscriptions, sPS*). To je čisti algoritam particioniranja i jedini predloženi algoritam koji prostorno ne indeksira pretplate. Za svaku nadolazeću objavu ovaj algoritam najprije identificira prostorne particije kojima objava pripada i zatim je šalje na radne čvorove koji su odgovorni za te particije. Konačno dolazna objava se uspoređuje s pretplatama pohranjenim na datim čvorovima. Važno je napomenuti da je svaki takav čvor zapravo posrednik za dolaznu objavu i stoga ovaj algoritam pripada kategoriji usmjeravanja usporedbom treće strane (engl. *rendezvous*) u raspodijeljenim sustavima objavi-pretplati. Očito je ovaj algoritam memorijski učinkovitiji od algoritma RIS zbog particioniranja pretplata, ali je istovremeno i manje vremenski učinkovit jer za svaku particiju ide slijedno kroz sve pretplate unutar particije i provjerava koje od njih su zadovoljene objavom. Stoga je sličan raspodijeljenoj verziji paraleliziranog sekvencijalnog algoritma, ali je vremenski ipak učinkovitiji jer zanemaruje pretplate u particijama kojima objava ne pripada.

Algoritam 6.2 predstavlja pseudokod algoritma sPS. U inicijalizacijskom dijelu ovog algoritma se prvo odredi metoda prostornog particioniranja $\mathcal{F}$, koja geoprostorne objekte partici-

onira u skup mogućih prostornih particija $\boldsymbol{R}$ (2. linija). Zatim se ona pošalje svakom radnom čvoru $n_i \in \boldsymbol{N}^W$ (3. i 4. linija). U praksi se kao metode prostornog particioniranje koriste neke od metoda particioniranja prostornih podataka predstavljenih u poglavlju 4.2. Potom se za svaku pretplatu $s_i \in \boldsymbol{S}^A$ korištenjem metode $\mathcal{F}$ odredi skup particija $\boldsymbol{R}(g_{s_i}) \subseteq \boldsymbol{R}$ kojima pripada geoprostorni objekt g_{s_i} pretplate s_i te se za svaku particiju $R_i \in \boldsymbol{R}(g_{s_i})$ doda pretplata s_i u skup pretplata $\boldsymbol{S}(R_i)$ te particije (5.-7. linija). Nakon toga se za svaku prostornu particiju $R_i \in \boldsymbol{R}$ odabere radni čvor $n_{R_i} \in \boldsymbol{N}^W$ koji će biti zadužen za particiju R_i te mu se pošalje skup pretplata $\boldsymbol{S}(R_i)$ te particije (8. i 9. linija).

```
1  inicijalizacija:
2      odredi metodu prostornog particioniranja F koja prostorne
           objekte particionira u skup mogućih prostornih particija R
3      za svaki radni čvor n_i ∈ N^W
4          proslijedi metodu F svakom čvoru n_i
5      za svaku pretplatu s_i ∈ S^A
6          korištenjem metode F odredi skup particija F(g_{s_i}) ⊆ R kojima
               pripada geoprostorni objekt g_{s_i} pretplate s_i
7          za svaku particiju R_i ∈ F(g_{s_i}) dodaj pretplatu s_i u skup
               pretplata S(R_i) te particije
8      za svaku prostornu particiju R_i ∈ R
9          odaberi radni čvor n_{R_i} ∈ N^W koji će biti zadužen za particiju
               R_i te mu proslijedi skup pretplata S(R_i) te particije
10 algoritam:
11     za svaku novoobjavljenu objavu p_i ∈ P
12         korištenjem metode F odredi skup particija F(g_{p_i}) ⊆ R kojima
               pripada geoprostorni objekt g_{p_i} objave p_i
13         za svaku prostornu particiju R_i ∈ F(g_{p_i})
14             proslijedi objavu p_i na čvor n_{R_i} koji je zadužen za
                   particiju R_i
15             na čvoru n_{R_i}
16                 za svaku pretplatu s_i ∈ S(R_i)
17                     ako objava p_i zadovoljava pretplatu s_i
18                         stavi pretplatu s_i u skup pretplata S(p_i) koje
                               zadovoljava objava p_i
19         za svaku pretplatu s_i ∈ S(p_i) proslijedi objavu p_i pretplatniku
               pretplate s_i
```

Algoritam 6.2: Pseudokod algoritma sPS

Nakon provedbe inicijalizacijskog dijela ovog algoritma, mogu se početi obrađivati dolazne objave (11. linija). Za svaku novoobjavljenu objavu $p_i \in \boldsymbol{P}$ korištenjem metode prostornog

particioniranja $\mathcal{F}$ odredi se skup particija $\mathcal{F}(g_{p_i}) \subseteq \boldsymbol{R}$ kojima pripada geoprostorni objekt g_{p_i} objave p_i (12. linija). Potom se za svaku particiju $R_i \in \mathcal{F}(g_{p_i})$ proslijedi objava p_i na čvor n_{R_i} koji je zadužen za particiju R_i (13. i 14. linija). Nakon toga se na čvoru n_{R_i} za svaku pretplatu $s_i \in \boldsymbol{S}(R_i)$ provjeri zadovoljava li objava p_i pretplatu s_i (15. i 16. linija). Ako objava p_i zadovoljava pretplatu s_i, pretplata s_i se stavlja u skup pretplata $\boldsymbol{S}(p_i)$ koje zadovoljava objava p_i (18. linija). Potom se za svaku pretplatu $s_i \in \boldsymbol{S}(p_i)$ proslijedi objava p_i pretplatniku pretplate s_i (19. linija).

Teorem 2. *Pseudokod algoritma sPS (tj. Algoritam 6.2) zadovoljava specifikaciju geoprostornog sustava objavi-pretplati iz Definicije 6.*

Dokaz. Potrebno je dokazati da Algoritam 6.2 zadovoljava svojstva životnosti, legalnosti i jedinstvenosti. Za dokazivanje svojstva životnosti pretpostavimo da postoji objava p_i koja nije isporučena pretplatniku pretplate s_i iako ju zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i. Da algoritam sPS ne bi isporučio pretplatniku od s_i objavu p_i, pretplata s_i se ne smije nalaziti u skupu $\boldsymbol{S}(R_i)$ niti jedne particije $R_i \in \mathcal{F}(g_{p_i}) \subseteq \boldsymbol{R}$ kojima pripada geoprostorni objekt g_{p_i} objave p_i, što je vidljivo iz 12.-18. linije Algoritma 6.2. Formalno to možemo izraziti na sljedeći način: $\nexists R_i \in \boldsymbol{R}(g_{p_i}) \subseteq \boldsymbol{R} : s_i \in \boldsymbol{S}(R_i)$. Međutim, kako stoji da $match_{s_i}(p_i) = \top$ slijedi da $Disjoint(g_{p_i}, g_{s_i}) = \bot$ pa stoga s_i i p_i imaju barem jednu dodirnu točku, a kako particije $\boldsymbol{R}$ po Definiciji 2 potpuno prekrivaju geoprostorno područje od interesa, ta točka mora pripadati nekoj particiji R_i s kojom nisu disjunktni ni s_i niti p_i pa $\exists R_i \in \boldsymbol{R} : Disjoint(g_{p_i}, g_{R_i}) = Disjoint(g_{s_i}, g_{R_i}) = \bot$. Zbog toga $\exists R_i \in \boldsymbol{R}(g_{p_i}) \subseteq \boldsymbol{R} : s_i \in \boldsymbol{S}(R_i)$ jer je pretplata s_i za particiju R_i dodana u skup pretplata $\boldsymbol{S}(R_i)$ te particije (5.-7. linija Algoritma 6.2). Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva legalnosti pretpostavimo da postoji objava p_i isporučena pretplatniku pretplate s_i iako ju ne zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i. Međutim, iz 17. i 18. linije Algoritma 6.2 se vidi da za svaku objavu p_i koja je isporučenu pretplatniku pretplate s_i vrijedi $match_{s_i}(p_i) = \top$. Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva jedinstvenosti pretpostavimo da $\exists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput. Da bi algoritam sPS isporučio pretplatniku od s_i objavu p_i više od jednom, pretplata s_i se mora ponavljati u skupu pretplata $\boldsymbol{S}(p_i)$ koje zadovoljava p_i u 18. i 19. liniji Algoritma 6.2. Međutim, skup pretplata $\boldsymbol{S}(p_i)$ po svojoj definiciji ne sadrži duplikate. Stoga $\nexists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput što je u kontradikciji s pretpostavljenim. ■

6.3 Prostorno particionirani indeks i pretplate

Drugi predloženi algoritam particioniranja je **Prostorno particionirani indeks i pretplate** (engl. *spatially Partitioned Index and Subscriptions, sPIS*). Sličan je algoritmu sPS, ali je vremenski učinkovitiji od njega i ne provjerava sekvencijalno sve pretplate unutar svake particije kojoj objava pripada, već umjesto toga za svaku takvu particiju postavlja upit nad prostornim indeksom pretplata unutar particije kako bi učinkovito pronašao kandidate među njima. Analogno algoritmu sPS, ovaj algoritam također pripada kategoriji usmjeravanja usporedbom treće strane (engl. *rendezvous*) u raspodijeljenim sustavima objavi-pretplati. Algoritam sPIS je vremenski učinkovitiji od algoritma sPS jer ne prolazi sekvencijalno kroz sve pretplate unutar neke particije već provjerava samo podskup pretplata koje vrati prostorni indeks particije. S druge strane malo je manje memorijski učinkovit, jer je prostorni indeks pretplata ipak nešto veći od odgovarajućeg popisa pretplata.

Algoritam 6.3 predstavlja pseudokod algoritma sPIS. U inicijalizacijskom dijelu ovog algoritma se prvo odredi metoda prostornog particioniranja $\mathcal{F}$ koja prostorne objekte particionira u skup mogućih prostornih particija $\boldsymbol{R}$ (2. linija) te se za svaki radni čvor $n_i \in \boldsymbol{N}^W$ pošalje metoda prostornog particioniranja $\mathcal{F}$ (3. i 4. linija). U praksi se kao metode prostornog particioniranje koriste neke od metoda particioniranja prostornih podataka predstavljenih u poglavlju 4.2. Potom se za svaku pretplatu $s_i \in \boldsymbol{S}^A$ korištenjem metode $\mathcal{F}$ odredi skup particija $\mathcal{F}(g_{s_i}) \subseteq \boldsymbol{R}$ kojima pripada geoprostorni objekt g_{s_i} pretplate s_i (5. i 6. linija). Potom se za svaku particiju $R_i \in \mathcal{F}(g_{s_i})$ dodaje pretplata s_i u indeks pretplata $\mathcal{I}_{R_i}$ te particije (7. i 8. linija). Kao što je prethodno spomenuto u poglavlju 6.1 kao prostorni indeks koriste se neki od prostornih indeksa predstavljenih u poglavlju 4.1. Nakon toga se za svaku prostornu particiju $R_i \in \boldsymbol{R}$ odabere radni čvor $n_{R_i} \in \boldsymbol{N}^W$ koji će biti zadužen za particiju R_i te mu se pošalje indeks pretplata $\mathcal{I}_{R_i}$ te particije (9. i 10. linija).

```
1  inicijalizacija:
2      odredi metodu prostornog particioniranja F koja prostorne
           objekte particionira u skup mogućih prostornih particija R
3      za svaki radni čvor n_i ∈ N^W
4          proslijedi metodu F svakom čvoru n_i
5      za svaku pretplatu s_i ∈ S^A
6          korištenjem metode F odredi skup particija F(g_{s_i}) ⊆ R kojima
               pripada geoprostorni objekt g_{s_i} pretplate s_i
7          za svaku particiju R_i ∈ F(g_{s_i})
8              indeksiraj s_i korištenjem njenog geoprostornog objekta g_{s_i}
                   u indeks pretplata I_{R_i} te particije
9      za svaku prostornu particiju R_i ∈ R
10         odaberi radni čvor n_{R_i} ∈ N^W koji će biti zadužen za particiju
```

```
            R_i te mu proslijedi indeks pretplata I_{R_i} te particije
11  algoritam:
12      za svaku novoobjavljenu objavu p_i ∈ P
13          korištenjem metode F odredi skup particija F(g_{p_i}) ⊆ R kojima
                pripada geoprostorni objekt g_{p_i} objave p_i
14          za svaku prostornu particiju R_i ∈ F(g_{p_i})
15              proslijedi objavu p_i na čvor n_{R_i} koji je zadužen za
                    particiju R_i
16              na čvoru n_{R_i}
17                  u indeksu I_{R_i} pronađi pretplate kandidate S^C(p_i) za p_i
18                  za svakog kandidata s_i ∈ S^C(p_i)
19                          ako objava p_i zadovoljava kandidata s_i
20                              stavi pretplatu s_i u skup pretplata S(p_i)
                                    koje zadovoljava objava p_i
21          za svaku pretplatu s_i ∈ S(p_i) proslijedi objavu p_i pretplatniku
                pretplate s_i
```

Algoritam 6.3: Pseudokod algoritma sPIS

Nakon provedbe inicijalizacijskog dijela ovog algoritma, mogu se početi obrađivati dolazne objave (12. linija). Za svaku novoobjavljenu objavu $p_i \in \boldsymbol{P}$ korištenjem metode prostornog particioniranja $\mathcal{F}$ odredi se skup particija $\mathcal{F}(g_{p_i}) \subseteq \boldsymbol{R}$ kojima pripada geoprostorni objekt g_{p_i} objave p_i (13. linija). Potom se za svaku particiju $R_i \in \boldsymbol{R}(g_{p_i})$ proslijedi objava p_i na čvor n_{R_i} koji je zadužen za particiju R_i (14. i 15. linija). Nakon toga se na čvoru n_{R_i} postavlja upit nad prostornim indeksom $\mathcal{I}_{R_i}$ koji pronalazi skup $\boldsymbol{S}^C(p_i) = \mathcal{I}_{R_i}(g_{p_i})$ pretplata kandidata za objavu p_i (16. i 17. linija), te se za svakog kandidata $s_i \in \boldsymbol{S}^C(p_i)$ provjeri zadovoljava li objava p_i kandidata s_i (18. i 19. linija). Ukoliko objava p_i zadovoljava kandidata s_i, tada se pretplata s_i stavi u skup pretplata $\boldsymbol{S}(p_i)$ koje zadovoljava objava p_i (20. linija). Konačno se za svaku pretplatu $s_i \in \boldsymbol{S}(p_i)$ proslijedi objavu p_i pretplatniku pretplate s_i (21. linija)

Teorem 3. *Pseudokod algoritma sPIS (tj. Algoritam 6.3) zadovoljava specifikaciju geoprostornog sustava objavi-pretplati iz Definicije 6.*

Dokaz. Potrebno je dokazati da Algoritam 6.3 zadovoljava svojstva životnosti, legalnosti i jedinstvenosti. Za dokazivanje svojstva životnosti pretpostavimo da postoji objava p_i koja nije isporučena pretplatniku pretplate s_i iako ju zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i. Da algoritam sPIS ne bi isporučio pretplatniku od s_i objavu p_i, pretplata s_i se ne smije nalaziti u indeksu $\mathcal{I}_{R_i}$ niti jedne particije $R_i \in \mathcal{F}(g_{p_i}) \subseteq \boldsymbol{R}$ kojima pripada geoprostorni objekt g_{p_i} objave p_i jer će ju indeks $\mathcal{I}_{R_i}$ po Definiciji 1 sigurno vratiti u skupu $\boldsymbol{S}^C(p_i) = \mathcal{I}_{R_i}(g_{p_i})$, što je vidljivo iz 13.-20. linije Algoritma 6.3. Formalno to možemo izraziti na sljedeći način: $\nexists R_i \in \boldsymbol{R}(g_{p_i}) \subseteq \boldsymbol{R} : s_i \in \mathcal{I}_{R_i}$. Međutim, kako stoji da

$match_{s_i}(p_i) = \top$ slijedi da $Disjoint(g_{p_i}, g_{s_i}) = \bot$ pa stoga s_i i p_i imaju barem jednu dodirnu točku, a kako particije $\boldsymbol{R}$ po Definiciji 2 potpuno prekrivaju geoprostorno područje od interesa, ta točka mora pripadati nekoj particiji R_i s kojom nisu disjunktni ni s_i niti p_i pa $\exists R_i \in \boldsymbol{R}$: $Disjoint(g_{p_i}, g_{R_i}) = Disjoint(g_{s_i}, g_{R_i}) = \bot$. Zbog toga $\exists R_i \in \boldsymbol{R}(g_{p_i}) \subseteq \boldsymbol{R} : s_i \in \mathcal{I}_{R_i}$ jer je pretplata s_i za particiju R_i indeksirana u indeksu $\mathcal{I}_{R_i}$ pretplata te particije (5.-8. linija Algoritma 6.3). Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva legalnosti pretpostavimo da postoji objava p_i isporučena pretplatniku pretplate s_i iako ju ne zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i. Međutim, iz 19. i 20. linije Algoritma 6.3 se vidi da za svaku objavu p_i koja je isporučenu pretplatniku pretplate s_i vrijedi $match_{s_i}(p_i) = \top$. Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva jedinstvenosti pretpostavimo da $\exists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput. Da bi algoritam sPIS isporučio pretplatniku od s_i objavu p_i više od jednom, pretplata s_i se mora ponavljati u skupu pretplata $\boldsymbol{S}(p_i)$ koje zadovoljava p_i u 20. i 21. liniji Algoritma 6.3. Međutim, skup pretplata $\boldsymbol{S}(p_i)$ po svojoj definiciji ne sadrži duplikate. Stoga $\nexists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput što je u kontradikciji s pretpostavljenim. ■

6.4 Replicirani indeks i particionirane pretplate

Treći predloženi algoritam particioniranja je **Replicirani indeks i particionirane pretplate** (engl. *Replicated Index Partitioned Subscriptions, RIPS)*. Analizom vremenske učinkovitosti algoritma sPIS došlo se do ovog algoritma. Algoritam sPIS uvijek je sporiji od algoritma RIS jer se objava i njezine odgovarajuće pretplate mogu nalaziti u nekoliko različitih particija pa je stoga potrebno nekoliko puta izvršiti identičnu, ali zahtjevnu primjenu prostornog predikata te provjeriti odgovara li pretplata kandidat objavi. Kako bi se prevladao ovaj nedostatak algoritam RIPS u isto vrijeme koristi replicirani prostorni indeks i particioniranje pretplata. Međutim pohranjivanje cijelih pretplata u prostorni indeks (kao u algoritmu RIS) bilo bi u suprotnosti s ciljem da se pretplate particioniraju. Stoga je zaključeno da algoritam RIPS treba u prostornom indeksu pohraniti samo parove "ID particije - ID pretplate" kako bi se drastično smanjilo memorijsko zauzeće što je pristup sličan globalnom indeksu iz [20]. Međutim kod algoritma RIPS indeks nije centraliziran, već je umjesto toga repliciran na svim radnim čvorovima te uz to pohranjuje pretplate (tj. upite), a ne geoprostorne objekte (tj. objave). Za svaku dolaznu objavu algoritam RIPS najprije identificira ID-jeve pretplata kandidata. Zatim šalje dolaznu objavu i ID-jeve pretplata kandidata radnim čvorovima koji su odgovorni za te particije. Konačno dolazna objava se uspoređuje s pretplatama koje odgovaraju ovim ID-jevima i koje su pohranjene na ovim čvorovima. Analogno algoritmima sPS i sPIS, algoritam RISP također pripada kate-

goriji usmjeravanja usporedbom treće strane (engl. *rendezvous*) u raspodijeljenim sustavima objavi-pretplati. Za razliku od algoritama sPS i sPIS, u algoritmu RIPS metoda particioniranja koristi se samo za particioniranje pretplata na izvršitelje, a ne za identifikaciju particija kojima objava pripada. Stoga se kod algoritma RIPS mogu koristiti i neprostorne metode particioniranja pretplata, dok kod algoritama sPS i sPIS to isključivo mora biti metoda prostornog particioniranja. Dakle, predložene su dvije različite verzije algoritma RISP: RIhPS (engl. *Replicated Index hash Partitioned Subscriptions*) i RIsPS (engl. *Replicated Index spatially Partitioned Subscriptions*). Za particioniranje pretplata algoritam RihPS koristi metodu particioniranja s raspršenim adresiranjem (*hash partitioning*), dok algoritam RIsPS koristi metodu prostornog particioniranja (*spatial partitioning*). Međutim takva metoda prostornog particioniranja je modificirana verzija standardne metode prostornog particioniranja koja se koristi u algoritmima sPs i sPIS jer ona uvijek vraća samo jedan ID particije za svaku pretplatu, dok standardna metoda particioniranja vraća jedan ili više ID-jeva particija za svaku pretplatu. Ova funkcionalnost lako se postiže vraćanjem najmanjeg ID-ja particije među ID-jevima particija koje vraća standardna metoda prostornog particioniranja. Algoritam RIPS najbolje iskorištava repliciranje prostornog indeksa i particioniranje pretplata. Iako je njegov cjevovod najkompleksniji među predloženim algoritmima, ovaj algoritam najbolje balansira između prostorne i vremenske učinkovitosti što je prikazano u sljedećem poglavlju.

Algoritam 6.4 predstavlja pseudokod algoritma RIPS. U inicijalizacijskom dijelu ovog algoritma se prvo stvori novi prostorni indeks $\mathcal{I}$ u kojem će biti indeksirane pretplate te pohranjeni samo ID-jevi pretplate i particije kojoj ona pripada (2. linija). Zatim se odredi metoda (prostornog ili hash) particioniranja $\mathcal{F}$ koja svaku pretplatu particionira u točno jednu particiju iz skupa particija $\boldsymbol{R}$ (3. linija). Nakon toga, za svaku particiju $R_i \in \boldsymbol{R}$ se stvara mapu pretplata $\boldsymbol{M}(R_i)$ te particije iz koje će se korištenjem ID-ja pretplate kao ključa efikasno moći dohvatiti pretplata (4. i 5. linija). Potom se za svaku pretplatu $s_i \in \boldsymbol{S}^A$ (6. linija) napravi sljedeće:

1. korištenjem metode $\mathcal{F}$ se odredi particija $\mathcal{F}(s_i) = R_{s_i} \in \boldsymbol{R}$ kojoj s_i pripada (7. linija),
2. mapira se ključ $ID(s_i)$ na vrijednost s_i u mapi pretplata $\boldsymbol{M}(R_{s_i})$ te particije (8. linija) i
3. indeksira se par $\langle ID(s_i), ID(R_{s_i})\rangle$ u prostorni indeks $\mathcal{I}$ korištenjem geoprostornog objekta g_{s_i} pretplate s_i (9. linija).

U praksi se kao prostorni indeks koristi neki od prostornih indeksa predstavljenih u poglavlju 4.1. Neki od ovih prostornih indeksa se moraju izgraditi nakon što se u njih dodaju pretplate (10. linija). Nakon toga se stvara pomoćna mapa $\boldsymbol{M}(\boldsymbol{R})$ iz koje će se korištenjem ID-ja particije kao ključa efikasno moći dohvaćatiti radni čvor koji je zadužen za tu particiju (11. linija). Zatim se za svaku prostornu particiju $R_i \in \boldsymbol{R}$ (12. linija) odabere radni čvor $n_{R_i} \in \boldsymbol{N}^W$ koji će biti zadužen za particiju R_i te mu se pošalje mapa pretplata $\boldsymbol{M}(R_i)$ te particije (13. linija). Osim toga, se u mapi particija $\boldsymbol{M}(\boldsymbol{R})$ mapira ključ $ID(R_i)$ na radni čvor n_{R_i} (14. linija). Konačno, prostorni indeks $\mathcal{I}$ i mapa particija $\boldsymbol{M}(\boldsymbol{R})$ se repliciraju na sve radne čvorove (15. i 16. linija).

```
1  inicijalizacija:
2      stvori novi prostorni indeks 𝓘 u kojem će biti indeksirani parovi
           ID pretplate – ID particije kojoj pretplata pripada
3      odredi metodu particioniranja ℱ koja svaku pretplatu s_i
           particionira u točno jednu particiju R_{s_i} iz skupa particija R
4      za svaku particiju R_i ∈ R
5          stvori mapu pretplata M(R_i) te particije
6      za svaku pretplatu s_i ∈ S^A
7          korištenjem metode ℱ odredi particiju ℱ(s_i) = R_{s_i} ∈ R kojoj
               pripada pretplata s_i
8          mapiraj ključ ID(s_i) na vrijednost s_i u mapi pretplata M(R_{s_i}) te
               particije
9          indeksiraj par ⟨ID(s_i), ID(R_{s_i})⟩ u 𝓘 korištenjem geoprostornog
               objekta g_{s_i} pretplate s_i
10     izgradi indeks 𝓘 ako je potrebno
11     stvori mapu particija M(R)
12     za svaku particiju R_i ∈ R
13         odaberi radni čvor n_{R_i} ∈ N^W koji će biti zadužen za particiju
               R_i te mu proslijedi mapu pretplata M(R_i) te particije
14         mapiraj ključ ID(R_i) na radni čvor n_{R_i} u mapi particija M(R)
15     za svaki radni čvor n_i ∈ N^W
16         repliciraj indeks 𝓘 i mapu particija M(R) na čvor n_i
17 algoritam:
18     za svaku novoobjavljenu objavu p_i ∈ P
19         odaberi radni čvor n_i ∈ N^W koji će obraditi objavu p_i
20         proslijedi objavu p_i na čvor n_i
21         na čvoru n_i
22                 u indeksu 𝓘 pronađi parove ⟨ID(s_i), ID(R_{s_i})⟩ pretplata
                       kandidata s_i za p_i
23                 za svaki par ⟨ID(s_i), ID(R_{s_i})⟩
24                     dohvati radni čvor n_{R_{s_i}} iz mape particija M(R)
                           korištenjem ključa ID(R_{s_i}) te mu proslijedi
                           objavu p_i
25                     na čvoru n_{R_{s_i}}
26                             dohvati kandidata s_i iz mape pretplata
                                   M(R_{s_i}) korištenjem ključa ID(s_i)
27                             ako objava p_i zadovoljava kandidata s_i
28                                     stavi pretplatu s_i u skup
                                           pretplata S(p_i) koje
```

```
                                                    zadovoljava objava p_i
29          za svaku pretplatu s_i ∈ S(p_i) proslijedi objavu p_i pretplatniku
                pretplate s_i
```

Algoritam 6.4: Pseudokod algoritma RIPS

Nakon provedbe inicijalizacijskog dijela ovog algoritma, mogu se početi obrađivati dolazne objave (18. linija). Za svaku novoobjavljenu objavu $p_i \in \boldsymbol{P}$ odabere se radni čvor $n_i \in \boldsymbol{N}^W$ koji će obraditi objavu p_i i proslijedi se objava p_i na čvor n_i (19. i 20. linija). Zatim se na čvoru n_i postavlja upit nad prostornim indeksom $\mathcal{I}$ koji pronalazi skup parova $\mathcal{I}(g_{p_i}) = \{\ldots, \langle ID(s_i), ID(R_{s_i})\rangle, \ldots\}$ koji predstavljaju ID-jeve pretplate kandidata s_i i particije R_{s_i} kojoj ona pripada (21. i 22. linija). Potom se za svaki par $\langle ID(s_i), ID(R_{s_i})\rangle$ iz mape particija $\boldsymbol{M}(\boldsymbol{R})$ dohvati radni čvor $n_{R_{s_i}}$ koji je zadužen za particiju R_{s_i} kojoj kandidat s_i pripada (23. i 24. linija). Nakon što se proslijedi objava p_i na čvoru $n_{R_{s_i}}$ pronađe se kandidat s_i u mapi pretplata $\boldsymbol{M}(R_{s_i})$ (25. i 26. linija). Zatim se provjeri zadovoljava li objava p_i kandidata s_i (27. linija). Ukoliko objava p_i zadovoljava kandidata s_i, tada se pretplata s_i stavi u skup pretplata $\boldsymbol{S}(p_i)$ koje zadovoljava objava p_i (28. linija). Za svaku pretplatu $s_i \in \boldsymbol{S}(p_i)$ proslijedi se objava p_i pretplatniku pretplate s_i (29. linija).

Teorem 4. *Pseudokod algoritma sPS (tj. Algoritam 6.4) zadovoljava specifikaciju geoprostornog sustava objavi-pretplati iz Definicije 6.*

Dokaz. Potrebno je dokazati da Algoritam 6.4 zadovoljava svojstva životnosti, legalnosti i jedinstvenosti. Za dokazivanje svojstva životnosti pretpostavimo da postoji objava p_i koja nije isporučena pretplatniku pretplate s_i iako ju zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i. Da algoritam RIPS ne bi isporučio pretplatniku od s_i objavu p_i, par $\langle ID(s_i), ID(R_{s_i})\rangle$ se ne smije nalaziti u indeksu $\mathcal{I}$ ili se pretplata s_i ne smije nalaziti u mapi $\boldsymbol{M}(R_{s_i})$ particije $\mathcal{F}(s_i) = R_{s_i} \in \boldsymbol{R}$. Ako se par $\langle ID(s_i), ID(R_{s_i})\rangle$ nalazi u indeksu $\mathcal{I}$, on će ga po Definiciji 1 sigurno vratiti u skupu $\mathcal{I}(g_{p_i}) = \{\ldots, \langle ID(s_i), ID(R_{s_i})\rangle, \ldots\}$, što je vidljivo iz 18.-22. linije Algoritma 6.4. Analogno, ako je pretplata s_i mapirana u mapi $\boldsymbol{M}(R_{s_i})$ ona će sigurno biti dohvaćena iz te mape, što je vidljivo iz 26. linije Algoritma 6.4. Formalno to možemo izraziti na sljedeći način: $[\nexists R_i \in \boldsymbol{R}(g_{p_i}) \subseteq \boldsymbol{R} : \langle ID(s_i), ID(R_{s_i})\rangle \in \mathcal{I}] \vee [s_i \notin \boldsymbol{M}(R_{s_i})]$. Međutim, kako stoji da $match_{s_i}(p_i) = \top$ slijedi da $Disjoint(g_{p_i}, g_{s_i}) = \bot$ pa stoga s_i i p_i imaju barem jednu dodirnu točku, a kako particije $\boldsymbol{R}$ po Definiciji 2 potpuno prekrivaju geoprostorno područje od interesa, ta točka mora pripadati nekoj particiji R_i s kojom nisu disjunktni ni s_i niti p_i pa $\exists R_i \in \boldsymbol{R}$: $Disjoint(g_{p_i}, g_{R_i}) = Disjoint(g_{s_i}, g_{R_i}) = \bot$. Zbog toga $\exists R_i \in \boldsymbol{R}(g_{p_i}) \subseteq \boldsymbol{R} : \langle ID(s_i), ID(R_{s_i})\rangle \in \mathcal{I}$ jer je par $\langle ID(s_i), ID(R_{s_i})\rangle$ indeksiran u indeksu $\mathcal{I}$ (6.-9. linija Algoritma 6.4). Osim toga, svaka pretplata s_i je mapirana u mapi $\boldsymbol{M}(R_{s_i})$ svoje particije, što se vidi iz linija 11.-14. Algoritma 6.4. Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \top$ koja nije isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva legalnosti pretpostavimo da postoji objava

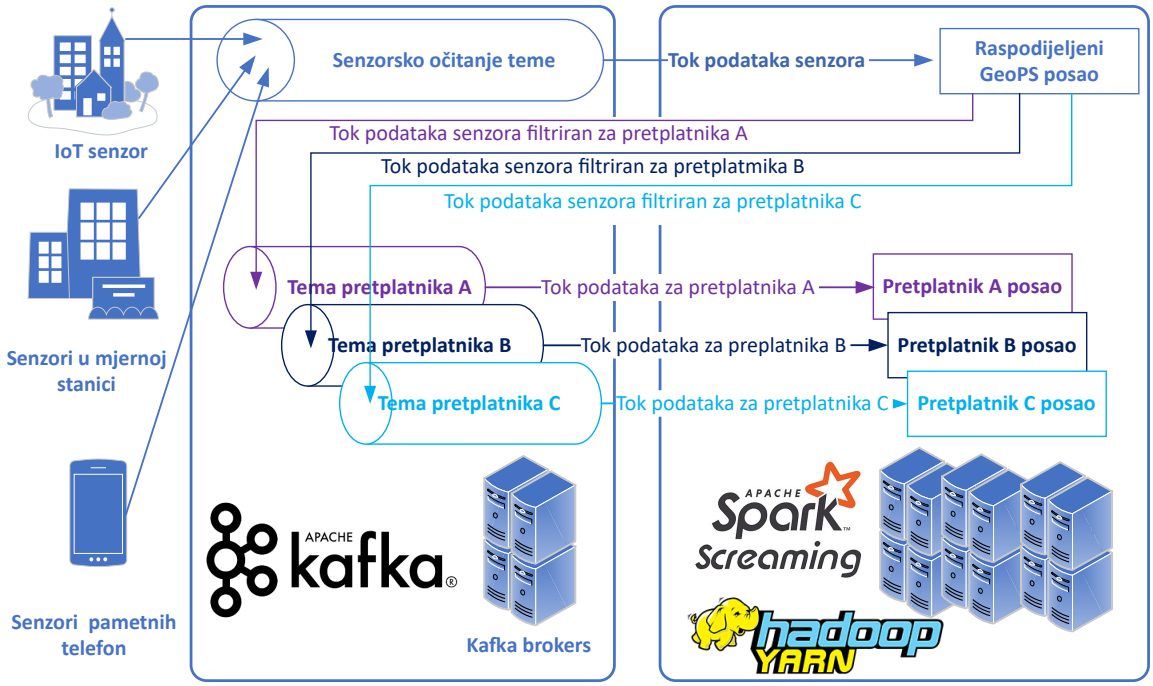
p_i isporučena pretplatniku pretplate s_i iako ju ne zadovoljava: $\exists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i. Međutim, iz 27. i 28. linije Algoritma 6.4 se vidi da za svaku objavu p_i koja je isporučenu pretplatniku pretplate s_i vrijedi $match_{s_i}(p_i) = \top$. Stoga $\nexists p_i \in \boldsymbol{P} : match_{s_i}(p_i) = \bot$ koja je isporučena pretplatniku pretplate s_i što je u kontradikciji s pretpostavljenim. Za dokazivanje svojstva jedinstvenosti pretpostavimo da $\exists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput. Da bi algoritam RIPS isporučio pretplatniku od s_i objavu p_i više od jednom, pretplata s_i se mora ponavljati u skupu pretplata $\boldsymbol{S}(p_i)$ koje zadovoljava p_i u 28. i 29. liniji Algoritma 6.4. Međutim, skup pretplata $\boldsymbol{S}(p_i)$ po svojoj definiciji ne sadrži duplikate. Stoga $\nexists p_i \in \boldsymbol{P}$ koja je pretplatniku pretplate s_i isporučena više od jedanput što je u kontradikciji s pretpostavljenim. ■

Poglavlje 7

Implementacija raspodijeljenog geoprostornog sustava objavi-pretplati

7.1 Arhitektura raspodijeljenog geoprostornog sustava objavi-pretplati

U ovom poglavlju opisana je arhitektura raspodijeljenog geoprostornog sustava objavi-pretplati (engl. *Geospatial Publish/Pubscribe system*). U osnovi se radi o kontinuiranom poslu koji se izvodi na platformi Apache Spark (engl. *Apache Spark Streaming Job*) posredstvom upravljačkog okvira Apache Hadoop YARN [21], koji upravlja raspodjelom poslova u grozdu, kao što je prikazano na slici 7.1. Ovo je raspodijeljeni sustav koji koristi jedan od algoritama usporedbe u raspodijeljenom geoprostornom sustavu objavi-pretplati, koji su predloženi u poglavlju 6. Kao što se vidi na slici 7.1., obrađuje se dolazni tok podataka koji sadrži geoprostorne objave koje se preuzimaju iz odgovarajuće teme platforme Apache Kafka. Za svakog pretplatnika filtriraju se objave iz dolaznog toka podataka i stvara se pretplatnički tok podataka, koji sadrži samo objave za koje je pretplatnik zainteresiran uzimajući u obzir njihovu lokaciju i sadržaj. Takav pretplatnički tok podataka objavljuje se u zasebnoj temi na platformi Apache Kafka koju onda lako može konzumirati neka pretplatnička komponenta poput nekog novog kontinuiranog posla koji se izvodi na platformi Apache Spark, kao što je prikazano na slici 7.1. Ako se ne bi koristio raspodijeljeni geoprostorni sustav objavi-pretplati tada bi svaka pretplatnička komponenta zasebno trebala filtrirati dolazni tok podataka što je iznimno zahtjevan zadatak u slučaju tokova podataka s visokim intenzitetom objavljivanja. Alternativno, umjesto zbirnog prikupljanja tokova podataka iz različitih izvora, kao što je prikazano na slici 7.1, svaka pretplatnička komponenta bi se mogla pretplatiti na izvor toka podataka od interesa, ali bi u tom slučaju bilo iznimno kompliciranо dodavati nove izvore i pretplatničke komponente u takav sustav. U nastavku ovog poglavlja predstavljaju se platforme Apache Spark i Apache Kafka.



Slika 7.1: Arhitektura raspodijeljenog geoprostornog sustava objavi-pretplati

7.1.1 Apache Spark

Apache Spark [22], [23] je platforma otvorenog koda za obradu velikih podataka s ugrađenim modulima za tokove podataka, jezik SQL, strojno učenje i grafove. Na slici 7.2 je prikazana arhitektura platforme Apache Spark. Najniži sloj predstavlja podatkovni sloj za pohranu podataka. Sloj iznad podatkovnog sloja predstavlja upravitelje resursima koji su zaduženi za komunikaciju između pokretača posla (engl. *driver*) i izvršitelja (engl. *executors*). Najbitniji sloj je Spark Core koji raspodijeljuje posao na zadatke, koordinira te vrši raspoređivanje posla na izvršitelje. U najvišem sloju arhitekture platforme Apache Spark nalaze se komponente koje pružaju programska sučelja (engl. *Application Programming Interface, API)* visoke razine apstrakcije za pojednostavljen razvoj aplikacija i brzu obradu podataka. Glavni način za postizanje učinkovite obrade kod platforme Apache Spark je pohranjivanje rezultata u memoriji. To je njena velika prednost u odnosu na tradicionalnu obradu programskim modelom *map-reduce* koja koristi pisanje i čitanje s diska. Platforma Apache Spark može raditi na grozdu u samostalnom načinu rada, koristeći vlastiti upravljački okvir ili pomoću drugog upravljačkog okvira kao što je Apache Hadoop Yarn. Platforma Apache Spark ne koristi se za pohranu podataka, već za njihovu obradu, ali se ona ipak vrlo lako povezuje s tehnologijama za pohranu velikih podataka, kao što su Apache HDFS, Apache Cassandre, Apache Kafke i mnoge druge. Kako bi se olakšao razvoj aplikacija za obradu velikih podataka, platforma Apache Spark pruža programska sučelja za različite programske jezike poput Jave, Scale i Pythona. Platforma Apache Spark također nudi programska sučelja za različite razine apstrakcije podataka, a to su *Resilient Distributed*

Dataset (RDD), *DataFrame* i *DataSet*. Svi oni predstavljaju raspodijeljenu kolekciju podataka koja je nepromjenjiva te otporna na pogreške (engl. *fault-tolerant*).

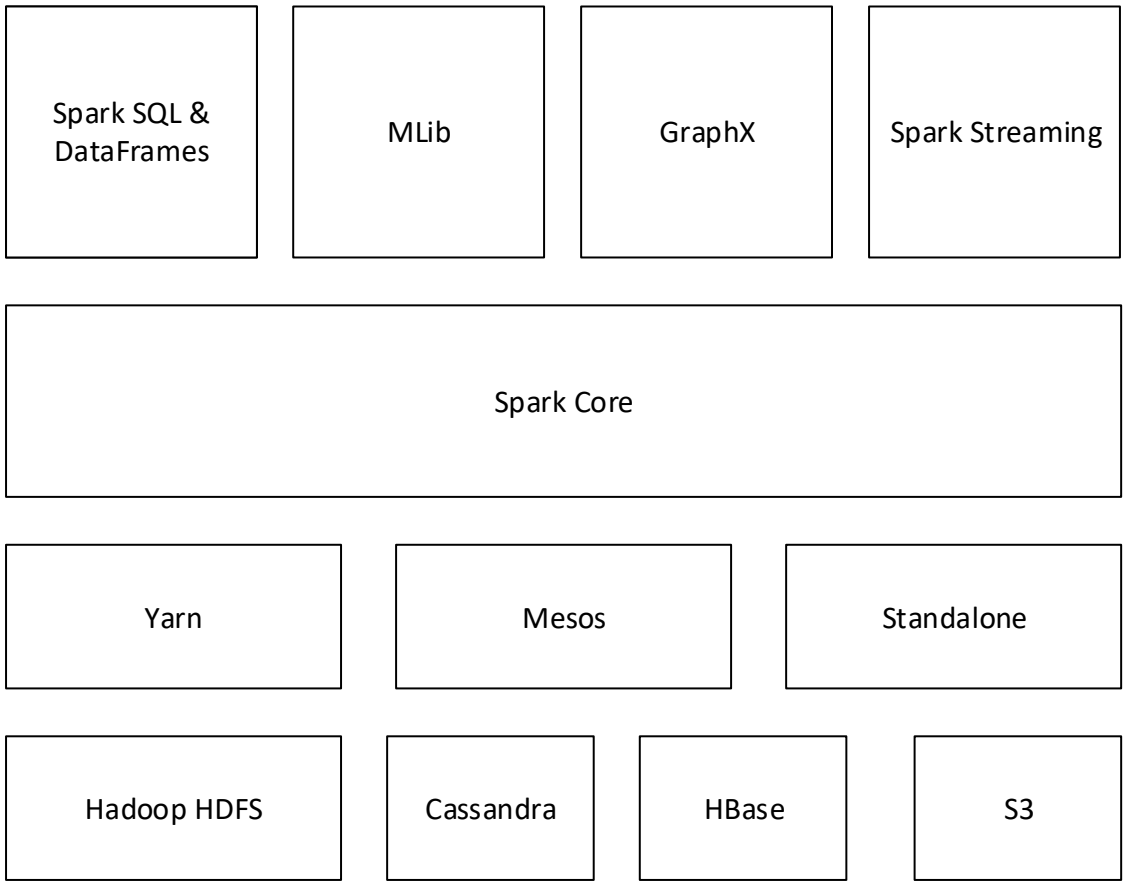
Resilient Distributed Datasets (RDD) osnovna je apstrakcija podataka u platformi Apache Spark. Jedna je od najvažnijih karakteristika ove kolekcije je da se operacije izvršavaju paralelno na Sparkovim izvršiteljima. Kao što je prethodno navedeno, svaki RDD je otporan na pogreške te se može oporaviti ukoliko dođe do gubitka nekog njegovog dijela (particije) zbog kvara. Otpornost na pogreške postiže se zapisivanjem svih operacija koje su provedene pri stvaranju svakog RDD-ja.

DataFrame i DataSet su oblici raspodijeljene nepromjenjive kolekcije podataka koja imaju imenovane stupce s definiranim tipom podataka. Slični su tablici u relacijskoj bazi podataka, kao i strukturi podataka *dataframe* koja se koristi u programskim jezicima R i Python. Pošto se temelje na kolekciji RDD, također su particionirani među Sparkovim izvršiteljima. Nad objektima tipa DataFrame i DataSet se mogu izvoditi razne operacije, kao što su grupiranje, filtriranje, agregiranje i mnoge druge. Prilikom izvođenja upita nad ovim kolekcijama, platforma Apache Spark provodi optimizaciju upita analiziranjem i organiziranjem svih operacija koje treba izvršiti. Za razliku od kolekcije DataFrame, kolekcija DataSet omogućava sigurnost tipova podataka pri prevođenju programskog koda (engl. *compile time type safety*).

U izvedbi raspodijeljenog geoprostornog sustava objavi-pretplati koristi se apstrakcija RDD jer pruža najnižu razinu operacija sa podacima i najbolju kontrolu podataka među tim sučeljima. Svaka Spark aplikacija pokreće jedan upravljački proces koji pokreće skup procesa izvršitelja raspoređenih po radnim čvorovima u grozdu. Izvršitelj ima nekoliko utora za izvršavanje zadataka i istodobno pokreće mnoge od njih tijekom svog životnog vijeka. Proces upravljačkog programa najprije pretvara aplikaciju u jednu ili više Spark poslova, a zatim se svaki posao transformira u logički plan izvršavanja koji je usmjereni aciklički graf (engl. *Directed acyclic graph, DGA)*. Nakon što je DAG kreiran, pokretački proces dijeli ga u faze na granicama određenima operacijom `shuffle` koja zahtijeva ponovno particioniranje podataka u grozdu i na taj način rezultira otežanom komunikacijom između radnih čvorova. Navedene faze se zatim dijele na manje zadatke koji se šalju na obradu izvršiteljima procesa. Faze se moraju izvoditi topološkim poretkom budući da ovise jedna o drugima, dok se zadaci unutar faze mogu izvoditi istodobno.

Spark nudi dva različita sustava za obradu podataka tj. Spark Streaming i Structured Streaming. Prvi sustav se temelji na apstrakciji RDD, dok se drugi temelji na apstrakcijama DataFrame i DataSet. U raspodijeljenom geoprostornom sustavu objavi-pretplati koji se predlaže u ovom doktorskom radu koristi se sustav Spark Streaming koji se temelji na apstrakciji visoke razine koja se naziva diskretizirani tok (engl. *DStream*), a predstavlja kontinuirani tok podataka. Tok podataka u sustavu Spark Streaming može biti iz različitih izvora (npr. Apache Kafka) i obrađuje se pomoću operacija nad apstrakcijom RDD. Sustav Spark Streaming dijeli DStream

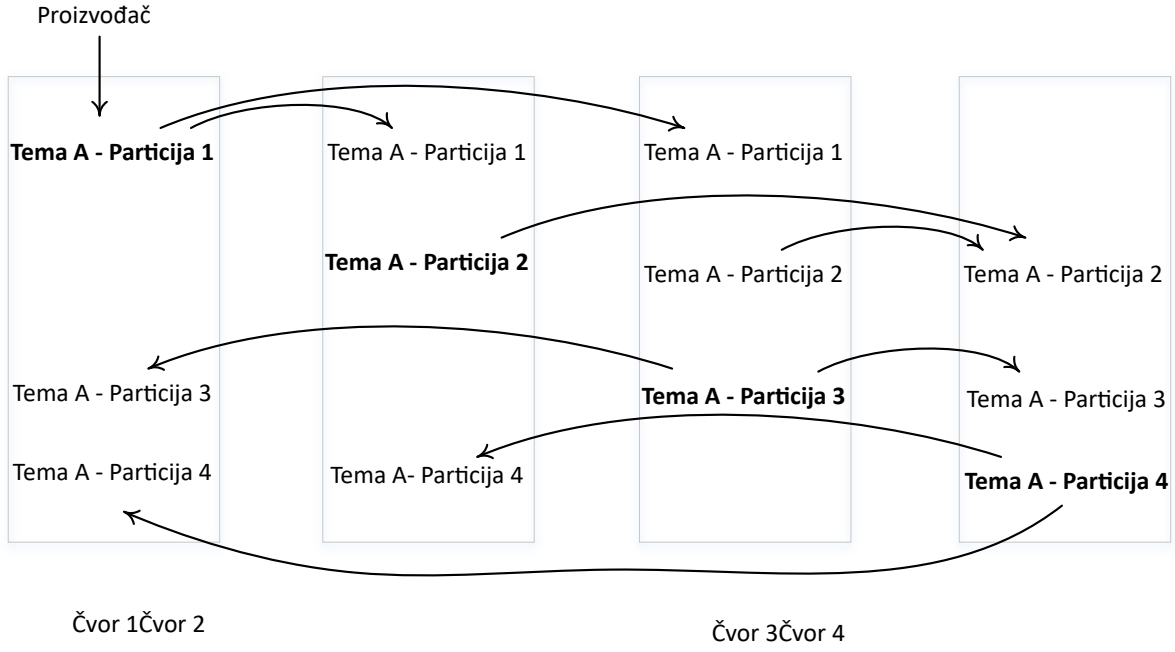
u niz mikro-skupina podataka (engl. *micro-batch*) koje se zatim obrađuju kao zasebni poslovi sustava Spark Streaming. Može se obraditi nekoliko mikro-skupina (kontinuiranih poslova sustava Spark Streaming) istodobno postavljanjem unutarnjeg konfiguracijskog parametara upravljačkog programa `spark.streaming.concurrentJobs` na željenu vrijednosti koja je veća od zadane vrijednosti 1.



Slika 7.2: Arhitektura Apache Spark

7.1.2 Apache Kafka

Apache Kafka [24, 25] je platforma otvorenog koda za obradu velikih podataka koja se temelji na modelu objavi-pretplati. Svaka poruka (engl. *record*) koju poslužitelj primi je par ključ/vrijednost i ima vremensku oznaku (engl. *timestamp*). Protivno CBPS (engl. *Content Base Publish Subscribe*) sustavu u kojem pretplata opisuje interes kao skup različitih ograničenja u sadržaju objava, platforma Apache Kafka podržava jednostavnije pretplate temeljene na temama koje definiraju teme (ili kanale) od interesa. Kafka objavljivači objavljuju svoje zapise (tj. parove ključ /vrijednost) na temu, dok se potrošači preplaćuju na jednu ili više tema. Svaka Kafka tema je samo raspodijeljeni dnevnik zapisivanja, koji je podijeljen na jednu ili više particija koje su pohranjene na posrednicima (engl. *brokers*). Kafka dopušta samo jednog potrošača (iz grupe potrošača) po particiji teme, ali više grupa potrošača može konzumirati zapise iz iste particije. Pri isporuci poruka potrošačima postiže visoku propusnost i nisko kašnjenje zbog različitih tehnika poboljšanja performanci koje primjenjuje, kao što su slijedni pristup disku i priručno spremanje stranica operacijskog sustava umjesto priručnog spremanja u procesima[26]. Na slici 7.3 je prikazana arhitektura platforme Apache Kafka.



Slika 7.3: Arhitektura Apache Kafka

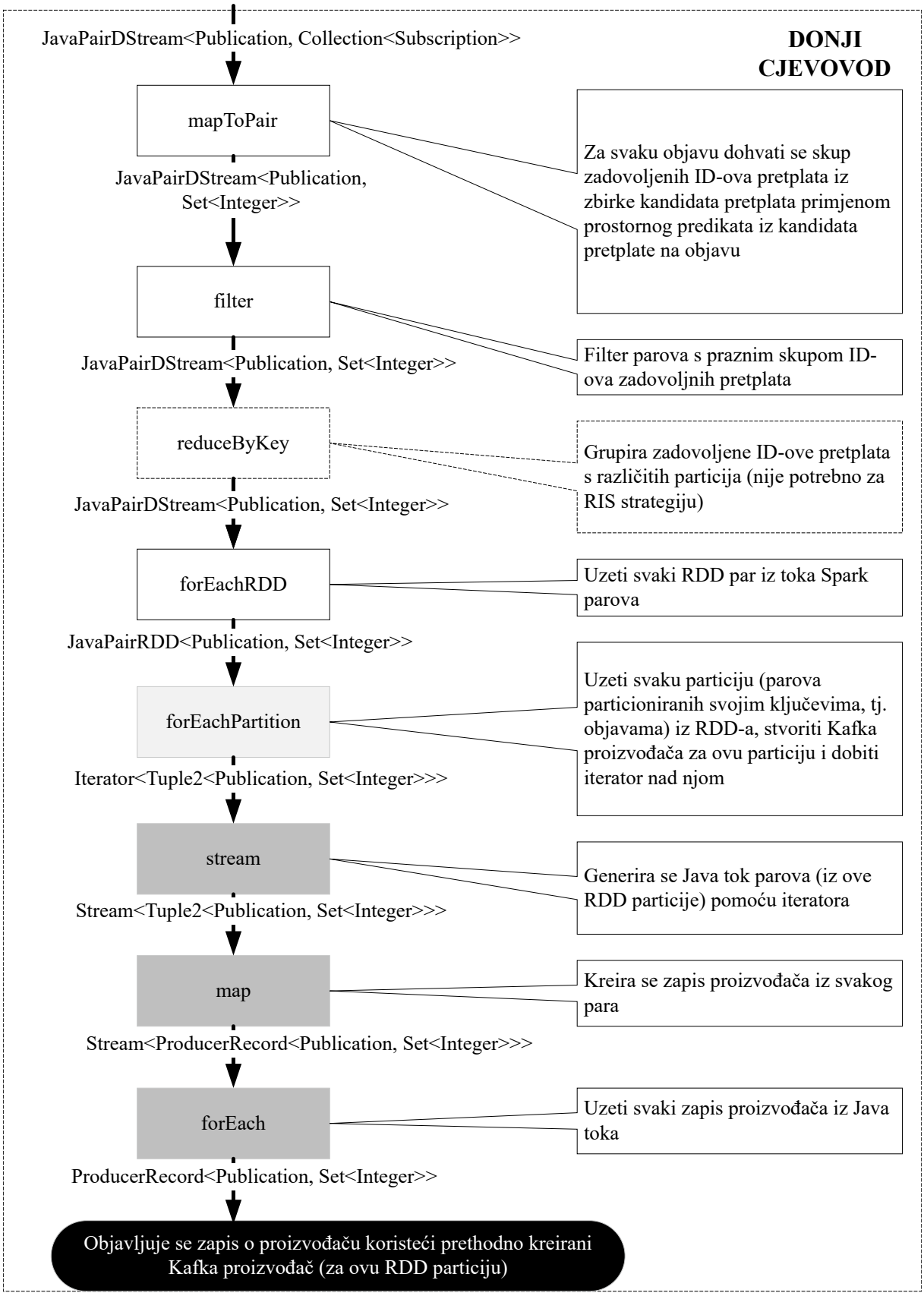
7.2 Algoritmi usporedbe u raspodijeljenom geoprostornom sustavu objavi-pretplati

U ovom poglavlju se predstavlja Sparkova implementacija svakog od četiri algoritma usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati, koji su predloženi

u poglavlju 6. Oni su predstavljeni s cjevovodom operacija koje se izvode nad objavama iz dolaznog toka geoprostornih podataka. Pri tome su moguće sljedeće tri vrste operacija: Sparkove operacije na objektom tipa `DStream`, Sparkove operacije nad objektom tipa `RDD` i Javine operacije nad objektima tipa `Stream`. Za svaku operaciju prikazana je vrsta dolaznih i odlaznih objekata te je ukratko objašnjeno kako se ona izvodi.

Cjevovod svakog algoritma sastoji se od dva dijela odnosno gornjeg cjevovoda i donjeg cjevovoda. U gornjem cjevovodu pronalaze se pretplate koje su zadovoljene dolaznim objavama te je ovaj cjevovod jedinstven za svaki od algoritama, dok je donji cjevovod identičan za sve algoritme pa je prikazan zasebno na slici 7.4. Prikazani cjevovod odgovara algoritamskom odsječku pseudokoda Algoritma 6.1,6.2,6.3. Međutim, postoji razlika u tome kako se pretplatnici obavještavaju o objavama koje zadovoljavaju njihove pretplate u donjem cjevovodu raspodijeljenog geoprostornog sustava objavi-pretplati koji se predstavlja u ovom poglavlju, u odnosu na spomenuti pseudokod. U pseudokodu se objave isporučuju zainteresiranim pretplatnicima, dok se u donjem cjevovodu na posebnoj Kafkinoj temi objavljuju objava i skup ID-jeva pretplata koje ona zadovoljava. Potrošač ove posebne Kafkine teme bi u praksi isporučivao objave zainteresiranim pretplatnicima, ali se on ne razmatra u ovom poglavlju.

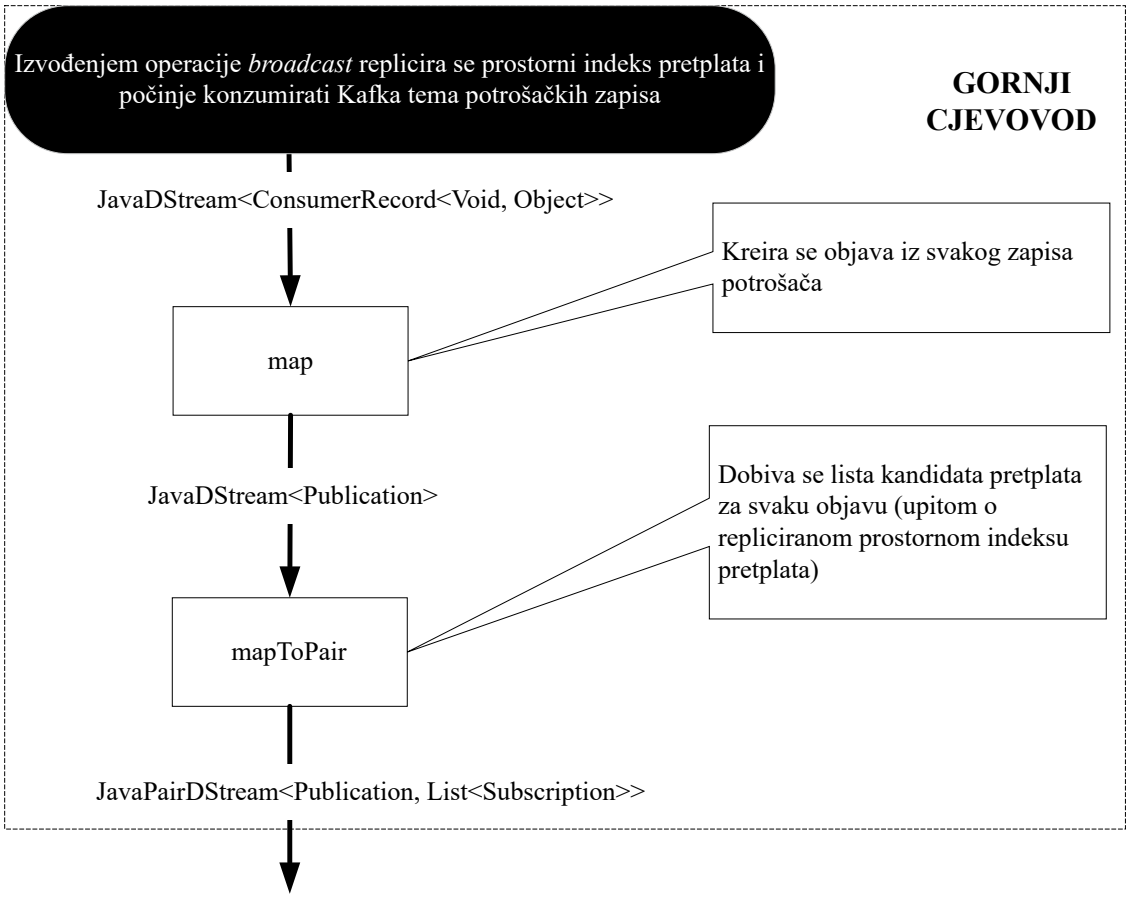
Nakon što se odredio par `<Publication,Collection<Subscription>>` koji sadrži objave i odgovarajuću kolekciju pretplata kandidata u gornjem cjevovodu, kao što je prikazano na slici 7.4, primjenjuje se prostorni predikat iz svake pretplate kandidata na objavu, a zatim se prikupe ID-jevi zadovoljenih pretplata u skup `Set<Integer>`. Nakon toga koristi se operacija `filter` za filtriranje parova koji sadrže objave koje ne zadovoljavaju nikakvu pretplatu, a zatim se prikupljaju parovi `<Publication,Set<Integer>>` s različitih particija (tj. izvršitelja) korištenjem operacije `reduceByKey`. Ovo je zahtjevna operacija koju uključuje operaciju `shuffle`, ali je neophodna jer sprječava višestruku isporuku jedne objave za istu pretplatu i time garantira isporuku samo jednom. Međutim, nije potrebna kada algoritam ne provodi particioniranje pretplata jer se u tom slučaju svaka objava tada obrađuje samo kod jednog izvršitelja. Potom se kod svakog izvršitelja stvara proizvođač objava na Kafkinoj temi, koji se koristi za objavljivanje svih prikupljenih parova `<Publication,Set<Integer>>` na tom izvršitelju. Da bi se to učinilo, prvo se poziva operacija `forEachRDD` za pristup RDD-jevima u toku. Zatim se poziva operacija `forEachPartition`, koja je Sparkova operacija nad objektom tipa `RDD` za stvaranje Kafkinog proizvođača i dobivanje iteratora preko parova `<Publication,Set<Integer>>` i stoga je označena svijetlo sivom bojom na slici 7.4. Konačno se kreira Javin tok parova s operacijom `stream` i primjenjuje operacija `map` na njemu kako bi se dobio odgovarajući zapis `ProducerRecord<Publication,Set<Integer>>`, koji se objavljuje operacijom `forEach`. Posljednje tri operacije nisu Sparkove opeacije nad objektom tipa `DStream` nego Javine operacije nad objektom tipa `Stream` i stoga su označene tamno sivom bojom na slici 7.4.



Slika 7.4: Donji cjevovod svih algoritama

7.2.1 Replicirani indeks i pretplate

Gornji cjevovod algoritma RIS prikazan je na slici 7.5. Prvo se replicira prostorni indeks pretplata na sve izvršitelje što je postignuto u platformi Apache Spark izvođenjem operacije `broadcoast`, a odgovara inicijalizacijskom odsječku u Algoritmu 6.1. Nakon toga počinju se u cjevovodu konzumirati zapisi `ConsumerRecord<Publication>` iz dolaznog toka podataka. Za svaki konzumirani zapis izdvaja se objava pomoću operacije `map` i zatim se koristi za upite nad repliciranim prostornim indeksom operacijom `mapToPair` što odgovara 13. liniji u Algoritmu 6.1. Kao rezultat ove operacije dobiva se lista kandidata pretplata `List<Subscription>` koje predstavljaju ulaz u donji cjevovod.



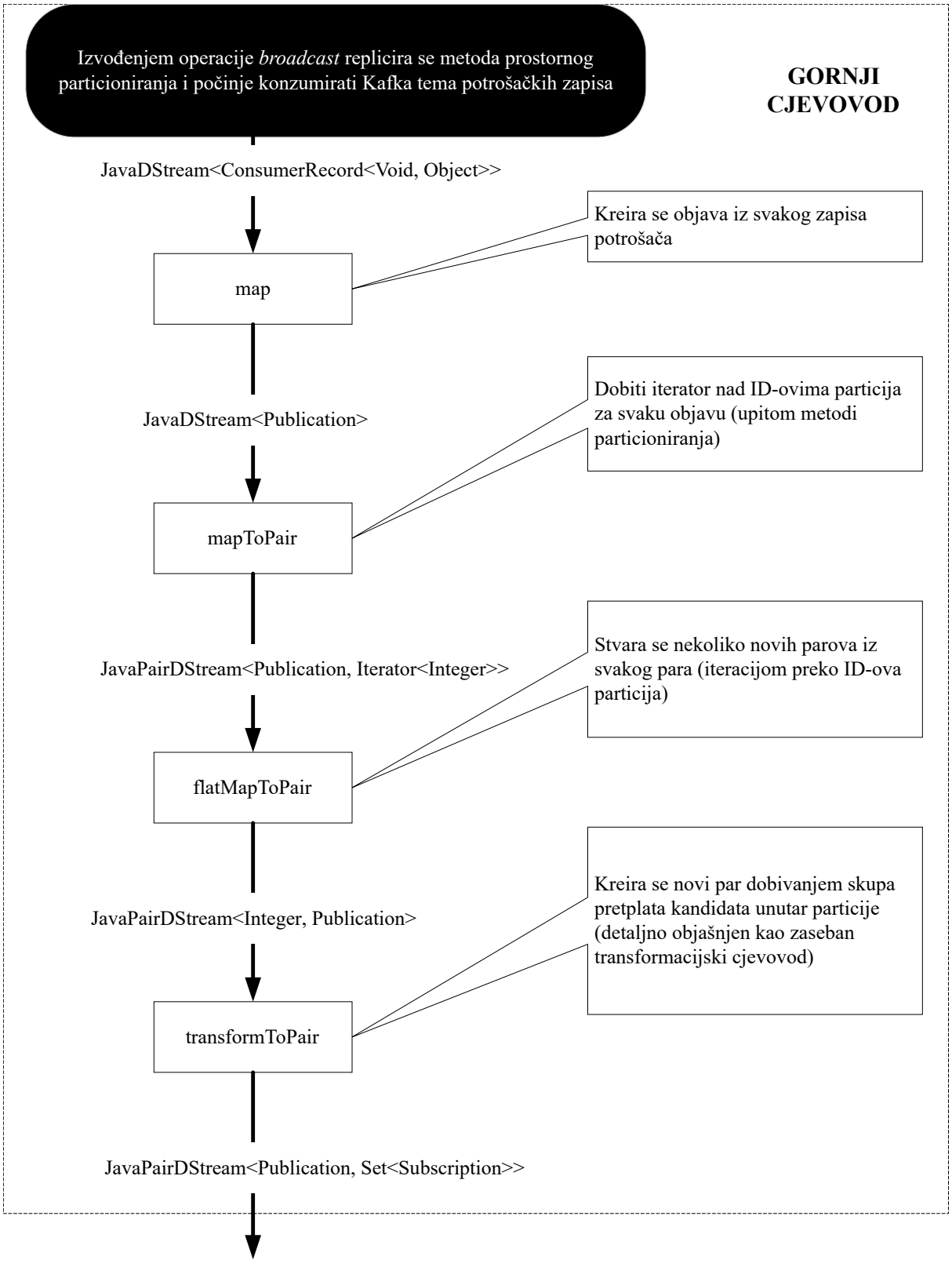
Slika 7.5: Gornji cjevovod algoritma RIS

7.2.2 Prostorno particionirane pretplate

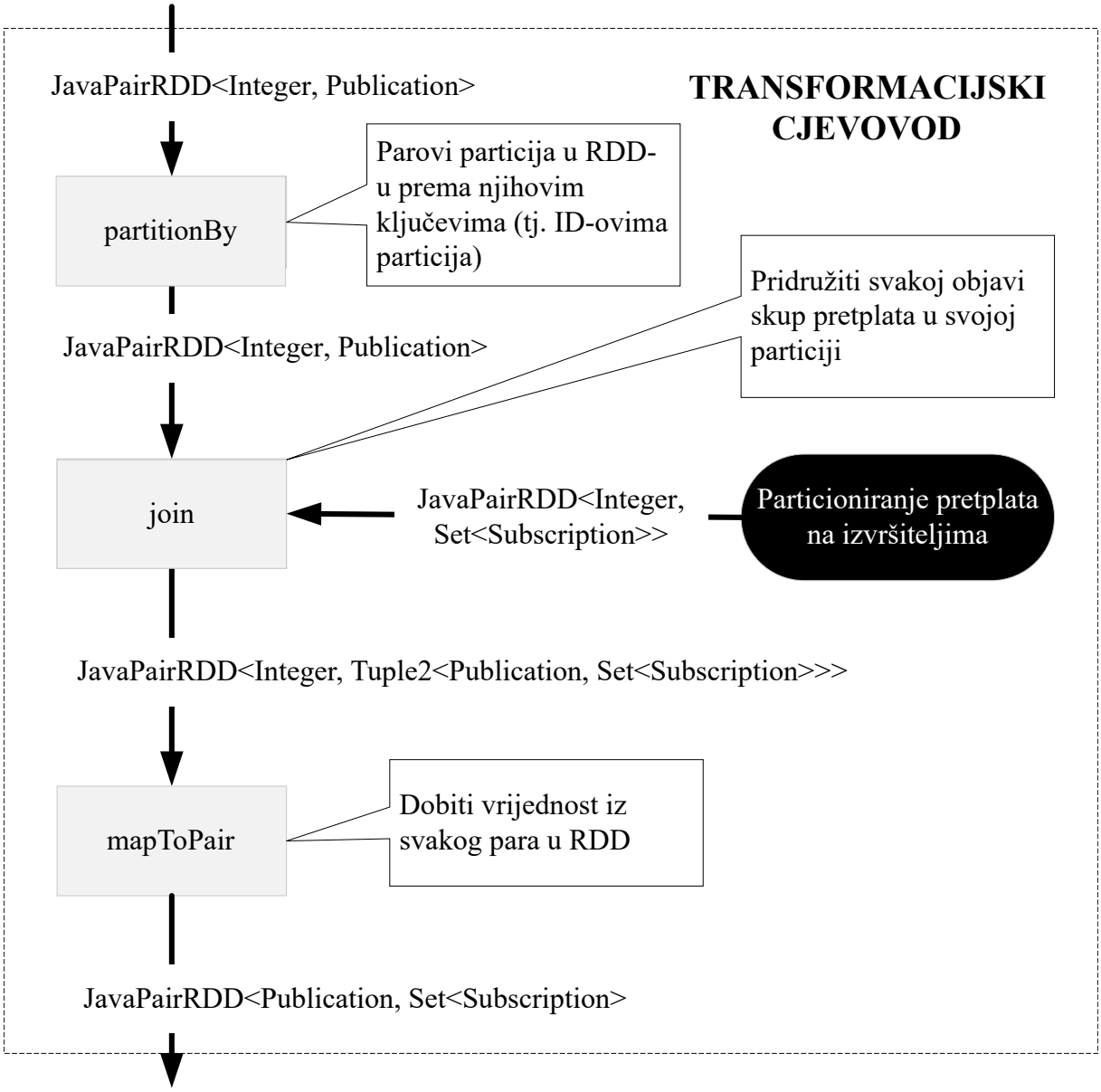
Gornji cjevovod algoritma sPS prikazan je na slici 7.6. Kao što je prikazano završava operacijom `transformToPair` koja uzima svaki RDD iz dolaznog toka podataka i transformira ga u novi RDD. Takva transformacija je u praksi novi cjevovod Sparkovih operacija nad objektima tipa `RDD` koji je prikazan na slici 7.7. Prvo se replicira metoda prostornog particioniranja, koja je velika nekoliko stotina kilobajta, svim izvršiteljima izvedbom operacije `broadcoast` na plat-

formi Apache Spark, kao što je prikazano na slici 7.6 a odgovara inicijalizacijskom odsječku u Algoritmu 6.2. Potom se kako je prikazano na slici 7.7 particioniraju pretplate kako bi se dobio skup `Set<Subscription>` pretplata za svaku particiju što odgovara 6. i 7. liniji Algoritma 6.2. Kao rezultat particioniranja dobiva se popis `List<Tuple2<Integer,Set<Subscription>>>` parova koji sadrži ID particije, kao prvi element, i skup pretplata u particiji, kao drugi element. Potom se kreira RDD parova iz ovih parova koji se raspodijeli po izvršiteljima u grozdu izvođenjem operacije `parallelizePairs` na platformi Apache Spark što odgovara 8. i 9. liniji Algoritma 6.2.

Nakon završetka repliciranja metode prostornog particioniranja i završetka prostornog particioniranja pretplata počinju se u cjevovodu konzumirati zapisi `ConsumerRecord<Publication>` iz dolaznog toka podataka, što odgovara algoritamskom odsječku u Algoritmu 6.2. Za svaki `ConsumerRecord<Publication>` prvo se izdvaja objava pomoću operacije `map`, a zatim se u sljedećem koraku operacijom `mapToPair` postavlja upit metodi particioniranja radi utvrđivanja particija kojima objava pripada što odgovara liniji 12. u Algoritmu 6.2. Potom se za svaki ID particije koju vrati metoda particioniranja koristi operacija `flatMapToPair` za kreiranje para `<Integer,Publication>` čiji je prvi element ID particije. Unutar transformacijskog cjevovoda prikazanog na slici 7.7, koji predstavlja lanac Sparkovih operacija nad objektima tipa `RDD`, particioniraju se parovi na izvršitelje kako bi se spojili s particioniranim pretplatama operacijom `join`. Kao rezultat ove operacije za svaku particiju kojoj objava pripada dobiva se par `<Integer,Tuple2<Publication,Set<Subscription>>>` u kojem je prvi element ID particije, dok je drugi element par koja sadrži objavu i skup pretplata u ovoj particiji što odgovara 16. liniji u Algoritmu 6.2. Konačno operacija `mapToPair` samo izdvaja par `<Publication, Set<Subscription>>` iz svakog para.



Slika 7.6: Gornji cjevovod algoritama sPS i sPIS

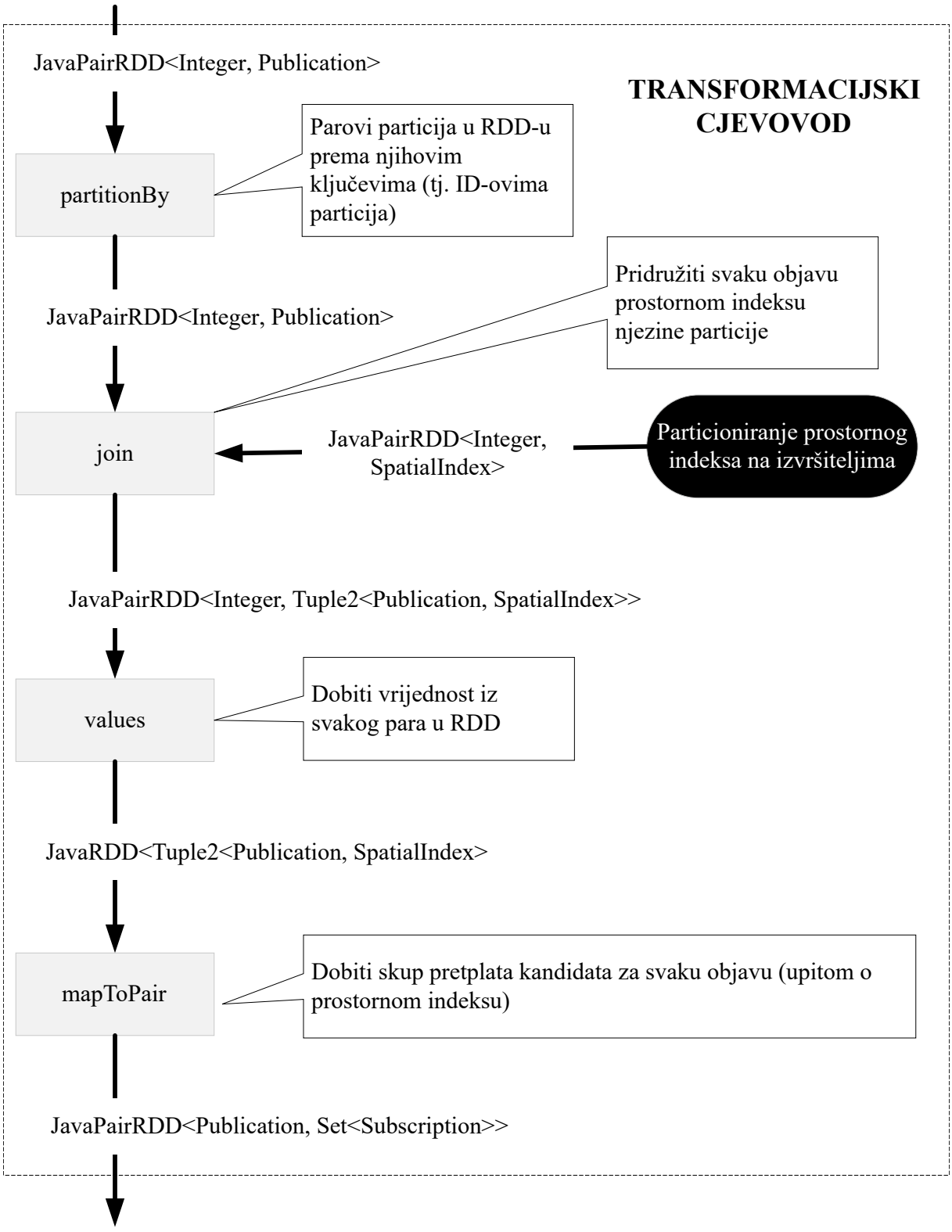


Slika 7.7: Transformacijski cjevovod algoritma sPS

7.2.3 Prostorno particionirani indeks i pretplate

Gornji cjevovod algoritma sPIS identičan je algoritmu sPS i prikazan je na slici 7.6 dok je transformacijski cjevovod drugačiji i prikazan je na slici 7.8. Pretplate se prvo particioniraju određenom metodom particioniranja, a zatim ih se prostorno indeksira kao što je prikazano na slici 7.8 što odgovara inicijalizacijskom odsječku u Algoritmu 6.3. Kao rezultat particioniranja dobiva se popis `List<Tuple2<Integer,SpatialIndex>>` parova koji sadrže ID particije kao prvi element i prostorni indeks pretplata u particiji kao drugi element. Ponovno se kreira par RDD od ovih parova i raspodijeli ih na izvršitelje u grozdu pozivanjem operacije `parallelizePairs` u platformi Apache Spark. Transformacijski cjevovod gotovo je identičan algoritmu sPS budući da ima samo jednu dodatnu Sparkovu operaciju na RDD-om, a to je operacija `mapToPair` koja

se poziva na kraju kako bi se dobio `Set<Subscription>` skup pretplata kandidata iz prostornog indeksa particije kojoj objava pripada što odgovara 17. liniji u Algoritmu 6.3. Operacija `values` kod algoritma sPIS identična je kao i operacija `mapToPair` kod algoritma sPS jer samo izvlači vrijednosti iz svakog para.

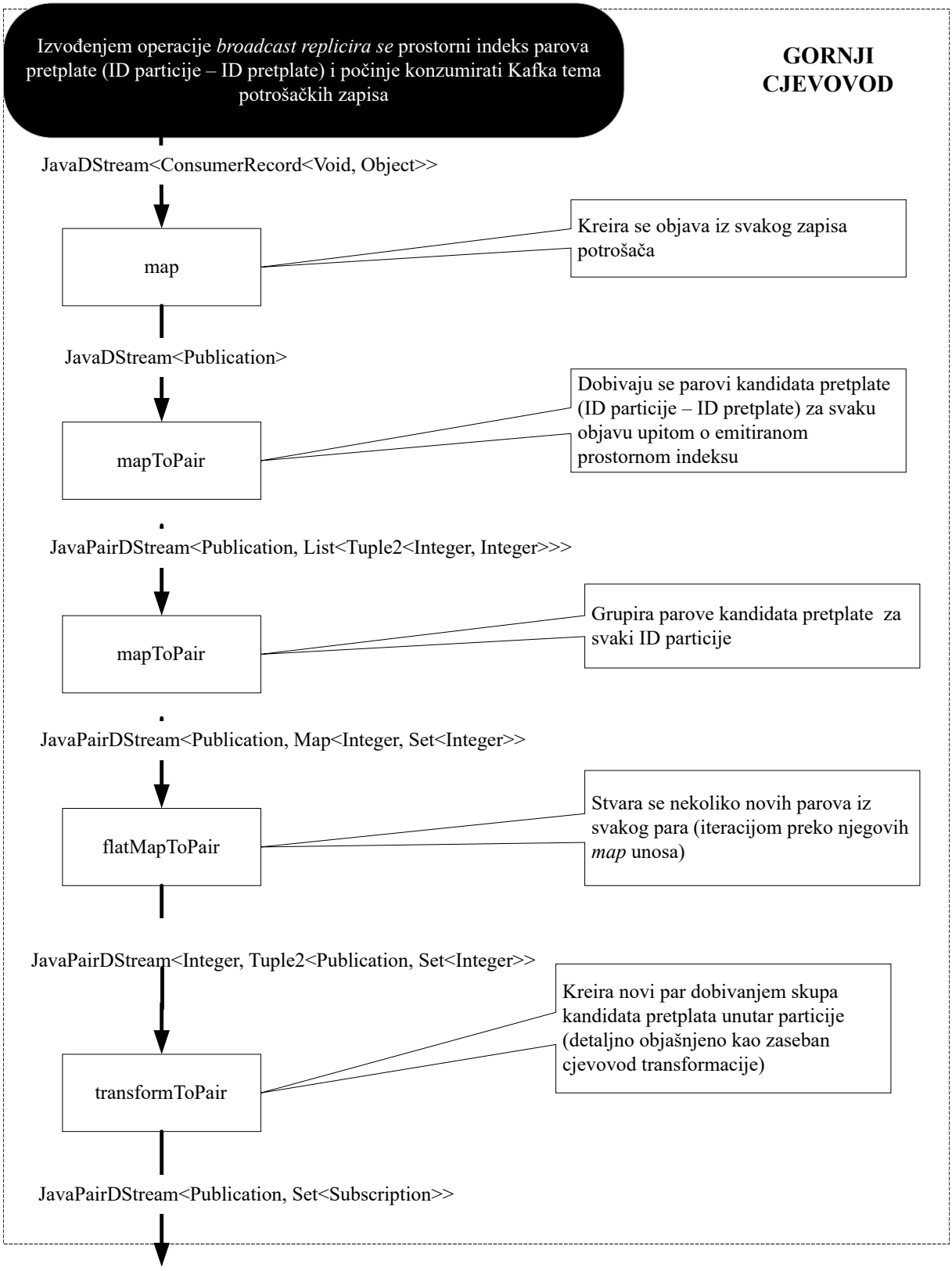


Slika 7.8: Transformacijski cjevovod algoritma sPIS

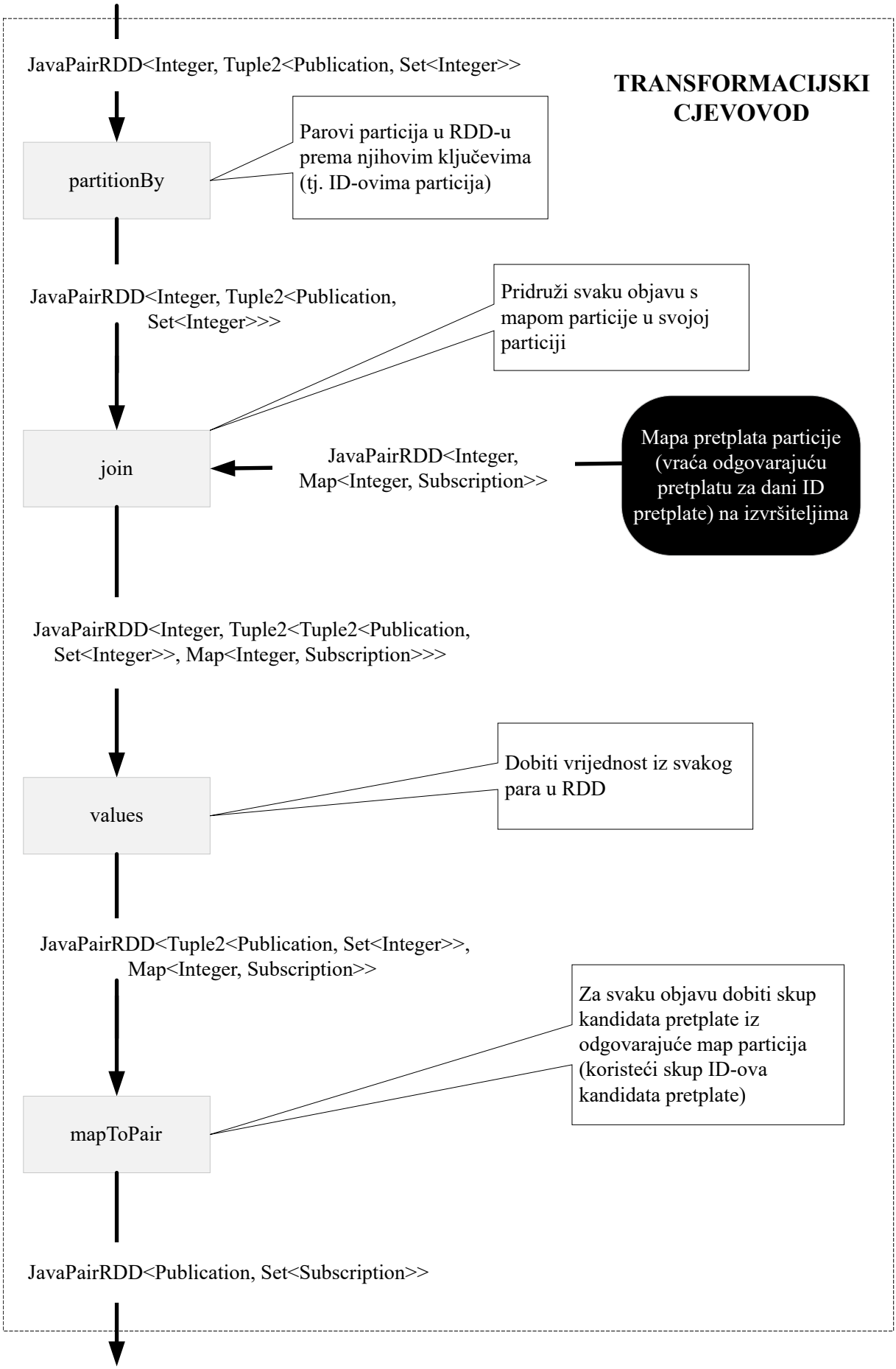
7.2.4 Replicirani indeks i particionirane pretplate

Gornji cjevovod algoritma RIPS prikazan je na slici 7.9 dok je transformacijski cjevovod prikazan na slici 7.10. Analogno algoritmu RIS prvo se replicira prostorni indeks svim izvršiteljima u grozdu izvođenjem operacija `broadcast` u platformi Apache Spark što je prikazano na slici 7.9 a odgovara inicijalizacijskom odsječku Algoritma 6.4. Kao što je prethodno objašnjeno to je prostorni indeks parova pretplata "ID particije - ID pretplate". Takav se par kasnije koristi za dobivanje pretplate iz svoje particije po njenom ID-ju. Pretplate se particioniraju na izvršitelje u mapi pretplata za svaku particiju tj. u mapama particija (engl. *partition maps*) `Map<Integer,Subscription>` da bi se kasnije mogle brzo pronaći po njihovim ID-jevima te je prikazano na slici 7.10. Kao rezultat se dobiva lista parova `List<Tuple2<Integer,Map< Integer, Subscription>>>` koja sadrži ID particije kao prvi element i mapu particije kao drugi element. Kreira se par RDD od ovih parova i raspodijeli se na izvršitelje pozivanjem operacije `parallelizePairs` u platformi Apache Spark.

Nakon repliciranja metode prostornog particioniranja i particioniranja mapa pretplata počinju se konzumirati pristigle objave. Analogno algoritmima sPS i sPIS izdvaja se objava iz svakog `ConsumerRecord<Publication>` i zatim se koristi za upite nad repliciranim prostornim indeksom parova "ID particije - ID pretplate" operacijom `mapToPair` što odgovara algoritamskom odsječku u Algoritmu 6.4. Kao rezultat ove operacije dobivaju se parovi `<Publication , List<Tuple2<Integer,Integer>>>` u kojima je prvi element objava, dok je drugi element lista parova koji sadrže ID particije i ID pretplate kandidata kao njegov prvi i drugi element što odgovara 22. liniji Algoritma 6.4. Zatim se koristi druga operaciju `mapToPair` za kombiniranje ID-jeva pretplate prema ID-ju particije. Kao rezultat ove operacije dobiva se `Map<Integer, Set<Integer>` u kojoj je prvi element ID particije, dok je drugi element skup ID-jeva kandidata pretplata (za taj ID particije). Zatim se koristi operacija `flatMapToPair` za kreiranje para `<Integer, Tuple2<Publication,Set<Integer>>` u kojem je prvi element ID particije dok je drugi element par koji sadrži objavu i skup ID-jeva kandidata pretplata (za taj ID particije). Unutar cjevovoda transformacije koji je prikazan na slici 7.10 kao lanac Sparkovih operacija nad RDD-jem se ovi parovi particioniraju na izvršitelje i združuju s prethodno particioniranim pretplatama u mapama particija pozivajući operaciju `join`. Kao rezultat ove operacije združivanja dobivaju se elementi `<Integer,Tuple2<Tuple2<Publication, Set <Integer>>,Map<Integer,Subscription>>>` u kojima je prvi element ID particije dok je drugi element par koji pakira ugniježđeni par objave i ID-jeve pretplata kandidata, kao prvi element vanjskog para, te mapu pretplata ove particije, kao drugi element vanjskog para. Zatim se izvodi operacija `values` koja samo izdvaja vrijednost iz svakog elementa. Konačno pozivanjem operacije `mapToPair` za svaki ID pretplate kandidata u skupu `Set<Integer>` dobiva se pretplata iz mape `Map<Integer,Subscription>>`. Kao rezultat ove operacije dobiva se par `Tuple2<Publication,Set<Subscription>>`.



Slika 7.9: Gornji cjevovod RIPS algoritma



Slika 7.10: Transformacijski cjevovod RIPS algoritma

7.3 Implementacijski detalji

U implementiranom geoprostornom sustavu objavi-pretplati, geoprostorni objekti unutar pretplata i objava su tipa `Geometry` iz programske knjižnice JTS [7]. Bitno je naglasiti da sadržaj objave i funkcija filtra sadržaja pretplate nisu ograničeni predloženim modelom, već implementiranim sustavom jer u potpunosti ovise o primjeni u praksi. Bitno je naglasiti da ID pretplate u praksi može biti bilo koji jedinstveni objekt koji predstavlja pretplatu (npr. identifikator Kafkine teme pretplatnika). U implementaciji algoritama usporedbe u raspodijeljenom geoprostornom sustavu objavi-pretplati koriste se tri prostorna indeksa iz knjižnice JTS te šest metoda prostornog particioniranja koji su opisani u poglavlju 4. U implementaciji predloženih algoritama RIS, sPS, sPIS i RIPS koriste se klase `QuadTree`, `STRTree` i `HPRTree` iz knjižnice JTS, kao i klase `VoronoiPartitioning`, `EqualPartitioning`, `QuadtreePartitioning`, `KDBTree`, `RtreePartitioning` i `HilbertPartitioning` iz programske knjižnice Apache Sedona (odnosno njezine starije izvedbe pod nazivom GeoSpark). Budući da posljednjih šest klasa samo particioniraju umetnute prostorne objekte također se iz programske knjižnice Apache Sedona (odnosno njezine starije izvedbe pod nazivom GeoSpark) koriste i klase `QuadTreePartitioner` , `KDBTreePartitioner`, `FlatGridPartitioner` i `SpatialPartitioner` da bi se moglo identificirati particije kojima pripada dati geoprostorni objekt. Ovaj skup klasa je gotovo potpuno samostalan s obzirom na ostale klase iz programske knjižnice Apache Sedona (odnosno njezine starije izvedbe pod nazivom GeoSpark) i ne uključuje klase povezane s prostornim RDD-jevima (engl. *Resilient Distributed Datasets*). Stoga se za upravljanje geoprostornim pretplatama izravno koriste čisti Sparkovi mehanizmi niske razine (npr. prilagođeno particioniranje i repliciranje), umjesto da se koriste implementacije visoke razine iz programske knjižnice Apache Sedona (odnosno njezine starije izvedbe pod nazivom GeoSpark). Glavni razlog zašto se ne koriste ostale klase iz programske knjižnice Apache Sedona je njihova neprikladnost za implementaciju raspodijeljenog geoprostornog sustava objavi-pretplati što će biti objašnjeno u nastavku.

Programska knjižnica Apache Sedona (odnosno njezina starija izvedba pod nazivom GeoSpark) podržava posebnu vrstu RDD-ja tj. prostorni RDD s prostornim objektima tipa `Geometry` iz JTS-a. Na taj način ukoliko je potrebno pakirati dodatne podatke s geoprostornim objektom unutar prostornog RDD-ja mora se to napraviti unutar atributa `userData` klase `Geometry`. Prostorni objekti u prostornom RDD-ju mogu biti samo indeksirani, samo particionirani, oboje (indeksirani i particionirani u isto vrijeme) ili ništa od navedenog. Programska knjižnica Apache Sedona podržava četiri vrste prostornih upita. Među te četiri vrste prostornih upita se isprva čini da se prostorni upit združivanja (engl. *spatial join query*) i prostorni upit raspona (engl. *spatial range queries*) mogu iskoristiti za izgradnju raspodijeljenog geoprostornog sustava objavi-pretplati koristeći algoritme sPS ili sPIS, koji su predloženi u poglavlju 7.2. Treba naglasiti

da se programska knjižnica Apache Sedona ne može koristiti za implementaciju algoritama RIS i RIPS, koji su također predloženi u poglavlju 7.2, jer algoritam RIS uopće ne particionira prostorne objekte, dok algoritam RIPS prostorno spaja parove `<Publication,Set<Integer >>` s parovima `Map<Integer,Subscription>`, gdje parovi i mape nisu prostorni objekti tipa `Geometry`.

Prostorni upit raspona (engl. *spatial range queries*) pronalazi objekte ne-particioniranog prostornog RDD-ja koji su prekriveni ili presječeni danim MBR-om (engl. *Minimum Bounding Rectangle*). Kao rezultat dobiva se prostorni RDD pronađenih objekata izvođenjem operacija nad izvorišnim RDD-jem što je razlog zašto se ne može koristiti za implementaciju raspodijeljenog geoprostornog sustava objavi-pretplati koji se temelji na sustavu Spark Streaming budući da bi on izvodio operacije nad RDD-jevima unutar drugih operacija nad RDD-jevima što nije podržano u platformi Apache Spark.

Postorni upit združivanja (engl. *spatial join query*) se izvodi nad particioniranim prostornim RDD-jem kako bi se on združio s drugim particioniranim prostornim RDD-jem. Za svaki geoprostorni objekt u prvom RDD-ju ovaj upit pronalazi geoprostorne objekte iz drugog RDD-ja koji su prekriveni ili presječeni njime. Budući da se ne transformira drugi RDD, već se iz njega samo izdvajaju objekti od interesa, ovaj upit se može koristiti za implementaciju raspodijeljenog geoprostornog sustava objavi-pretplati koji se temelji na sustavu Spark Streaming. U praksi se ovaj upit može koristiti unutar predloženih algoritama sPS i sPIS umjesto ručnog prostornog particioniranja RDD-jeva, ali u tom slučaju uštedilo bi se tek nekoliko redaka izvornog koda, dok bi cjevovod bio uglavnom isti. Međutim to bi dovelo do nekoliko dodatnih problema. Prvo implementacija ovog upita u metodi `SpatialJoinQueryFlat` uništava većinu atributa `userData` rezultata `<Geometry, Geometry>` parova, što je demonstrirano u radu [27]. Kada bi se ova metoda koristila u praksi (za implementaciju raspodijeljenog geoprostornog sustava objavi-pretplati), rezultirajuće pretplate bi izgubile svoje prostorne predikate i ID-jeve što bi bilo neprihvatljivo. Drugo, implementacija istog upita u metodi `SpatialJoinQuery` koja vraća parove `<Geometry,List<Geometry>>` gubi objekte s identičnom geometrijom dok agregira pronađene objekte na rezultirajućem popisu, iako ne uništava atribute `userData`. Kada bi se ona koristila u praksi izgubile bi se neke od odgovarajućih objava što ne bi bilo poželjno. Dodatno, pri tome se agregacija izvodi Sparkovom metodom `groupBy` nad RDD-jem koja je poznata po velikoj vremenskoj složenosti*. Štoviše, agregacija se događa u cjevovodu odmah nakon spajanja objava s pretplatama. Kada bi se u praksi (za implementaciju raspodijeljenog geoprostornog sustava objavi-pretplati) koristila navedena metoda `SpatialJoinQuery`, to bi dovelo do miješanja (operacijom *shuffle*) cijelih pretplata među izvršiteljima što bi povećalo mrežni promet i potrošnju memorije, ali i vremensku složenost obrade u usporedbi s predlože-

*Stoga se u implementiranom raspodijeljenom geoprostornom sustavu objavi-pretplati umjesto nje koristi mnogo učinkovitija operacija `reduceByKey`.

nim pristupom u kojem se agregacija odgađa sve dok se pretplate ne zamijene svojim ID-jevima u cjevovodu. Unatoč ovim problemima s prostornim upitom združivanja (engl. *spatial join query*) u programskoj knjižnici Apache Sedona, u eksperimentalnoj evaluaciji u poglavlju 8.3 se uspoređuju predloženi algoritmi s onima koji se koriste metodu `SpatialJoinQuery`.

Zbog jasnoće prezentacije u poglavlju 7.2 se ne raspravlja o dodavanju i brisanju pretplata niti mijenjanju metoda prostornog particioniranja tijekom rada sustava. Međutim to je u praksi podržano u predloženim algoritmima budući da pokretački proces može ponovno replicirati neophodne podatke ili ponovno paralelizirati korištene RDD-jeve.

U ovom doktorskom radu predlažu se različiti algoritmi usporedbe u raspodijeljenim geoprostornim sustavima objavi-pretplati i uspoređuju se njihove performanse, dok je izvan okvira ovog doktorskog rada pristup koji bi nadzirao dolazni tok podataka tijekom vremena kako bi prilagodbom metoda particioniranja reagirao na neuravnoteženost particija (engl. *partition skew*) pri promjeni raspodjele prostornih podataka. U eksperimentalnoj evaluaciji u ovom poglavlju koristi se početni skup pretplata za izgradnju metode particioniranja koja se zatim replicira na Sparkovim izvršiteljima. Osim toga ne vrednuju se performance algoritama pri dodavanju ili uklanjanju pretplata, kao što je prethodno objašnjeno.

Poglavlje 8

Eksperimentalna evaluacija raspodijeljenog geoprostornog sustava objavi-pretplati

U ovom poglavlju predstavljena je eksperimentalna studija za evaluaciju propusnosti, kašnjenja i potrošnje memorije algoritama RIS, SPS, sPIS, RIsPS i RIhPS, koji su predloženi u poglavlju 7.2, te konkurentskih centraliziranih i raspodijeljenih pristupa.

Uspoređene su različite varijante ovih algoritama ovisno o različitim prostornim indeksima i/ili metodama particioniranja opisanih u poglavlju 4. Ove varijante prikazane su u tablici 8.1 gdje je varijanta prostornog indeksiranja označena uglatim zagradama, a varijanta prostornog particioniranja vitičastim zagradama. Kao što je prikazano u tablici 8.1 postoji 48 različitih varijanti. Međutim, kao što će biti prikazano u prvom eksperimentu, mnoge varijante ne pokazuju zadovoljavajuće rezultate, pa se u daljnjoj evaluaciji koristi samo 20 različitih varijanti predloženih algoritama. Algoritam sPS ima 6 varijanti jer ne indeksira, već samo particionira pretplate. Nadalje, algoritmi RIhPS i RIS imaju 3 varijante budući da ne particioniraju pretplate, već ih samo indeksiraju. Konačno, algoritmi sPIS i RIsPS imaju 18 varijanti budući da ujedno indeksiraju i particioniraju pretplate. U ovom poglavlju se uspoređuju ove varijante kako bi se među njima identificirale najbolje po performancama kao one koje treba koristiti u praksi.

8.1 Provedba eksperimenata

Skupovi podataka korišteni u eksperimentima javno su dostupni u repozitorijima [28],[29]. Eksperimenti su zahtijevali skup podataka s geoprostornim objektima, pa je korišten skup podataka iz stvarnog svijeta iz [29] koji sadrži informacije o prometnim nesrećama u Ujedinjenom Kraljevstvu koje je prikupila policija. Ovaj skup podataka sadrži točnu geografsku lokaciju na kojoj se dogodila nesreća, broj vozila koja su sudjelovala u nesreći, točno vrijeme nesreće, broj žrtava

		Prostorni indeks			
		STR tree	*Quadtree*	*HPR tree*	*none*
Metoda prostornog particioniranja	*Voronoi*	sPIS_[ST]_{VO} RIPS_[ST]_{VO}	sPIS_[QT]_{VO} RIPS_[QT]_{VO}	sPIS_[HT]_{VO} RIPS_[HT]_{VO}	SPS_{VO}
	Equal grid	sPIS_[ST]_{EG} RIPS_[ST]_{EG}	sPIS_[QT]_{EG} RIPS_[QT]_{EG}	sPIS_[HT]_{EG} RIPS_[HT]_{EG}	SPS_{EG}
	Quadtree	sPIS_[ST]_{QT} RIPS_[ST]_{QT}	sPIS_[QT]_{QT} RIPS_[QT]_{QT}	sPIS_[HT]_{QT} RIPS_[HT]_{QT}	SPS_{QT}
	KDB tree	sPIS_[ST]_{KT} RIPS_[ST]_{KT}	sPIS_[QT]_{KT} RIPS_[QT]_{KT}	sPIS_[HT]_{KT} RIPS_[HT]_{KT}	SPS_{KT}
	STR tree	sPIS_[ST]_{ST} RIPS_[ST]_{ST}	sPIS_[QT]_{ST} RIPS_[QT]_{ST}	sPIS_[HT]_{ST} RIPS_[HT]_{ST}	SPS_{ST}
	Hilbert	sPIS_[ST]_{HI} RIPS_[ST]_{HI}	sPIS_[QT]_{HI} RIPS_[QT]_{HI}	sPIS_[HT]_{HI} RIPS_[HT]_{HI}	SPS_{HI}
	none	RIS_[ST] RIhPS_[ST]	RIS_[QT] RIhPS_[QT]	RIS_[HT] RIhPS_[HT]	

Tablica 8.1: Različite varijante strategija replikacija i particioniranja pretplata

i druge kontekstualne pojedinosti (vrsta ceste, ograničenje brzine, vrsta čvorišta, itd.). Međutim, u eksperimentima su korištene samo geografske lokacije iz ovog skupa podataka. Bitno je naglasiti da je geografska lokacija najjednostavniji geografski objekt koji se ne može koristiti za realističnu eksperimentalnu evaluaciju algoritama. Da bi se ispravno vrednovalo ove algoritme stoga je potrebno imati skup podataka sa složenijim geoprostornim objektima kao što su poligoni. Zbog toga je ovaj skup podataka kombiniran s poligonima poštanskih sektora, okruga i područja Ujedinjenog Kraljevstva [28]. Za generiranje poligona u objavama i pretplatama nasumično su korištene lokacije iz [29] kako bi se saznalo [28] kojem poštanskom sektoru, okrugu ili području pripada. Ovaj pristup je nadahnut radom [30] i smatra se adekvatnim za potrebe eksperimentalnog istraživanja budući da na ovaj način generirani poligoni zadržavaju prostornu raspodjelu geografskih lokacija [28]. Dodatno, sadržaj generiranih objava je cijeli broj koji predstavlja njihov ID, dok filtar sadržaja generiranih pretplata prihvaća sve objave. Na taj način eksperimentalna evaluacija uspoređuje samo performance prostorne usporedbe (objava i pretplata), a ne i usporedbe po sadržaju.

Svi predloženi algoritmi usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati implementirani su u Javi 8 (OpenJDK), a eksperimenti su izvedeni na ra-

Tablica 8.2: Zadani parametri eksperimenata

Parametar	**Vrijednost**
Broj objava i pretplata	10.000
Postotak objava točaka, okruga, sektora and i područja	40%, 30%, 20% i 10%
Broj particija pretplata	100
Prostorni predikati pretplata	mješovito (svi osim *disjoint*)
Postotak pretplata okruga, sektora i područja	50%, 30% i 20%
Vrsta Spark izvršitelja	balanced
Broj Spark izvršitelja	32
Broj istodobnih Spark poslova	4
Spark micro-batch interval [*s*]	1
Broj Kafka particija	32
Stopa dolaska [objave/*s*]	2048

čunalnom grozdu sa 16 identičnih radnih čvorova sa 64 GB radne memorije i procesorima Intel Core i7-9700K CPU @ 3.60GHz koji imaju 8 fizičkih jezgri bez tehnologije Hyper-Threading. Čvor sa Sparkovim pokretačem posla (engl. *driver*) je imao 32 GB radne memorije i procesor Intel Core i7-4790 CPU @ 3.60GHz koji ima 4 fizičke jezgre s tehnologijom Hyper-Threadingom. Osim ovih čvorova, grozd je imao i 7 dodatnih čvorova od kojih se 3 čvora koriste kao pomoćni čvorovi, dok se kao posrednici (engl. *brokers*) za platformu Apacke Kafka koriste 4 identična čvora sa 16 GB radne memorije i procesorima Intel Core i7-2600 CPU @ 3.40GHz. Na ovaj način radni čvorovi su posvećeni samo obradi (s ulogama *HDFS DataNode*, *Spark Gateway* i *YARN NodeManager*) i nisu pokrenute nikakve dodatne usluge na njima. Na grozdu je instaliran Cloudera CDH 6.2.1 Express i omogućene su usluge: HDFS, YARN, Spark, Zookeeper i Kafka. Zadani parametri simulacije koji su korišteni u eksperimentima navedeni su u tablici 8.2.

Zadani scenarij evaluacije korišten u svim eksperimentima je sljedeći: Prvo se generiraju pretplate i objave, a zatim se objave objavljuju na Kafkinu temu. Nakon toga se kreće s pokretanjem algoritma (tj. sparkove aplikacije) koji se želi testirati. Ovisno o algoritmu Sparkova aplikacija prvo indeksira, replicira i/ili particionira učitane pretplate, a zatim počinje obrađivati objave s Kafkine teme. Tijekom svakog eksperimenta mjeri se prosječna propusnost (engl. *throughput*) i kašnjenje (engl. *latency*) po obrađenoj objavi ili maksimalna potrošnja memorije po Sparkovom izvršitelju (engl. *executor*).

Platforma Apache Spark podržava dodavanje slušača tipa `StreamingListener` u aplikaciju koja koristi sustav Spark Streaming koji se aktivira kada se preda na obradu, započne ili završi

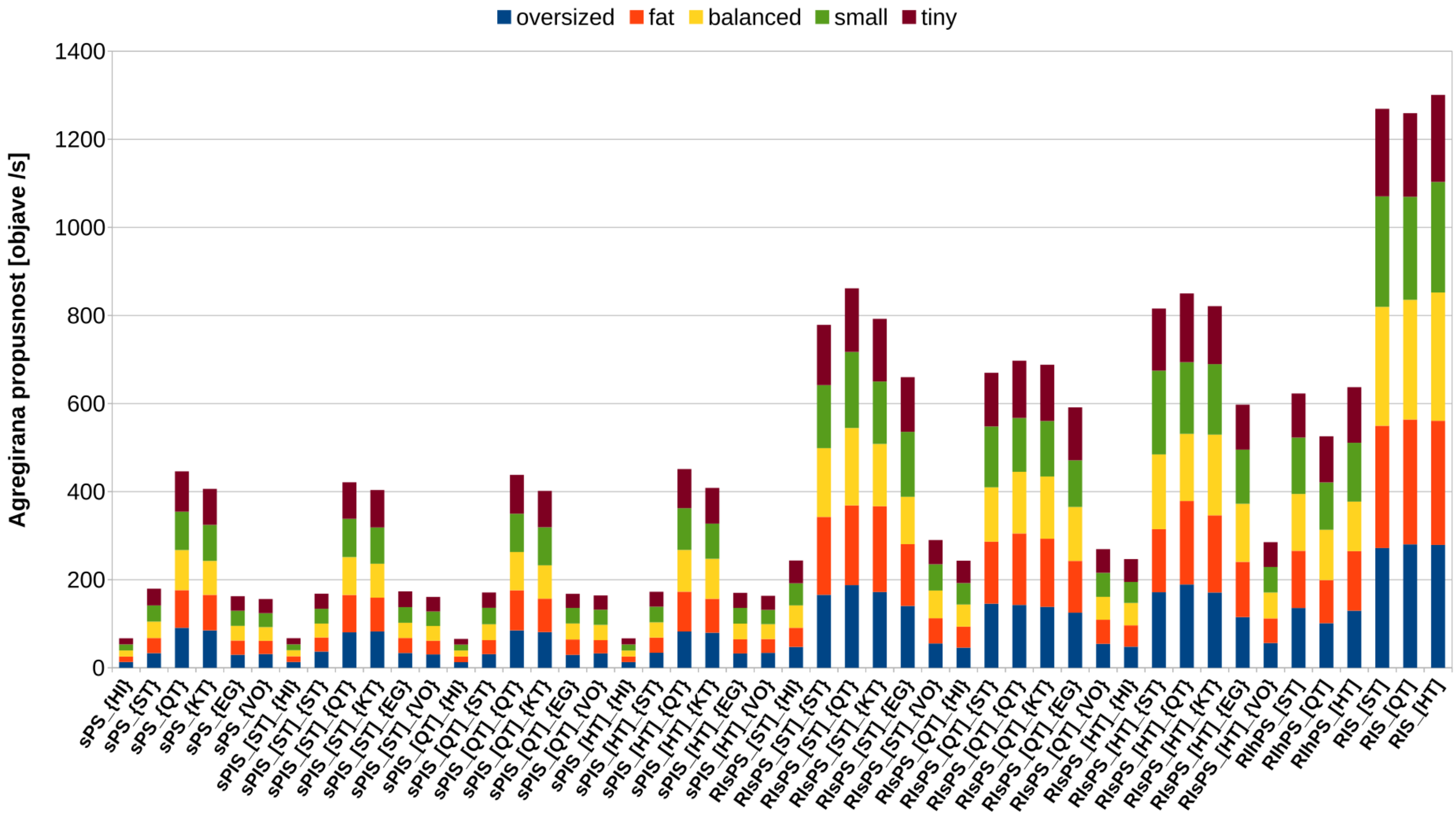
Tablica 8.3: Vrste Sparkovih izvršitelja

Vrste izvršitelja	*tiny*	*small*	*balanced*	*large*	*oversized*
Broj izvršitelja	96	48	32	16	16
Memorija po izvršitelju [GB]	8	16	24	48	48
Jezgre po izvršitelju	1	2	3	6	8

obrada svake mikro-skupine podataka (engl. *micro-batch*). Ovaj slušač tipa `StreamingListener` koristi se da se zadrži statistika tijekom pokretanja aplikacije i da se završno dobije propusnost i kašnjenje kada se izvođenje aplikacije prekine, nakon što se završi obrada dolaznih objava. Pri tome se mjere samo propusnost i kašnjenje obrade koji ne uključuju kašnjenja povezana sa serijalizacijom i deserijalizacijom poruka unutar Kafke zbog znatno veće složenosti obrade u odnosu na navedena kašnjenja. U usporedbi s propusnošću i kašnjenjem, mjerenje potrošnje memorije Sparkovih izvršitelja puno je teže postići budući da verzija instaliranog softvera Cloudera CDH Express koja je bila korištena ne podržava značajku *Container Usage Metrics* koja to nudi. Da bi se to postiglo morao se promijeniti izvorni kod YARN-ove klase `ContainersMonitorImpl`, koja se nalazi unutar paketa `org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor`. Ova klasa je namijenjena praćenju potrošnje memorije YARN-ovih spremnika (u kojima se izvršavaju Sparkovi izvršitelji) na radnim čvorovima i njihovom uništavanju kada njihove potrošnja radne memorije poraste iznad unaprijed definiranog praga. Zbog ove promjene svaki YARN-ov spremnik počeo je povremeno zapisivati korištenje memorije u datoteku. Također, napravljeno je nekoliko skripti koje su prikupljale zapise o korištenju memorije iz datoteka na radnim čvorovima, a zatim ih analizirale kako bi izdvojile maksimalnu vrijednost memorijskog zauzeća.

8.2 Usporedba različitih varijanti predloženih algoritama

Kao što je prethodno objašnjeno i prikazano u tablici 8.1, postoji 48 različitih varijanti predloženih algoritama ovisno o korištenom prostornom indeksu i/ili metodi particioniranja. U ovom poglavlju eksperimentalno se uspoređuje propusnost ovih 48 varijanti kako bi se identificirale one varijante koji bi se trebale koristiti u praksi.

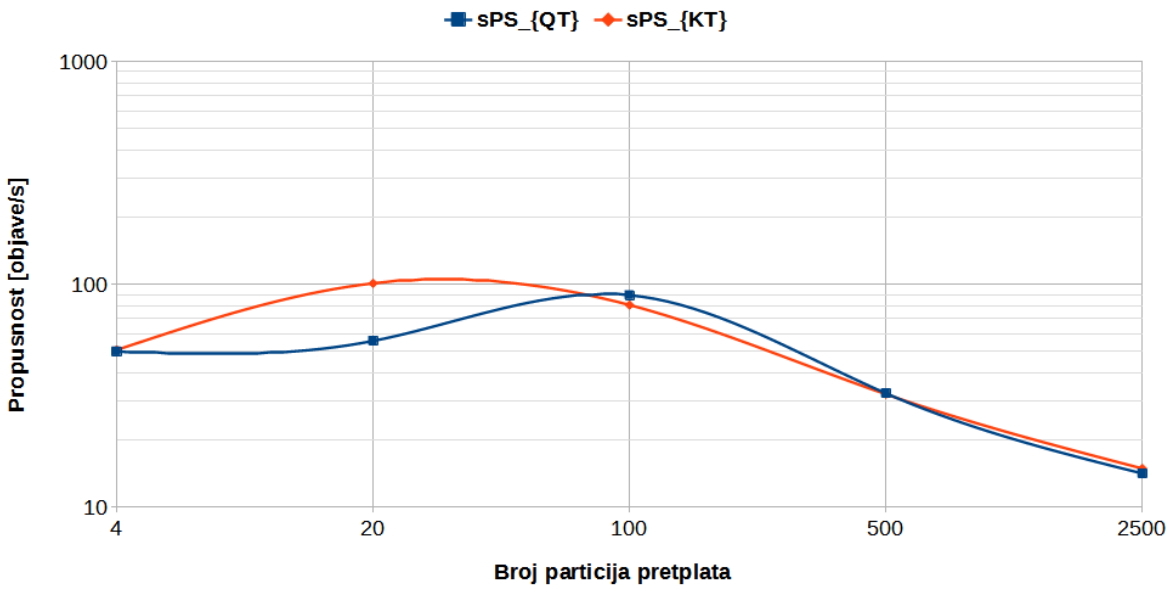


Slika 8.1: Agregirana propusnost vrsta Spark izvršitelja za različite varijante

Na slici 8.1 prikazan je rezultat agregirane propusnosti različitih tipova izvršitelja za različite varijante algoritama. Ova slika jasno pokazuje da neke varijante imaju lošije performance obrade. Najgore varijante su one s Hilbertovom metodom particioniranja i stoga će one biti izostavljene iz nastavka evaluacije. Osim njih, također će biti izostavljene varijante sa sljedećim metodama particioniranja: Equal Grid, Voronoi i STR Tree. Ostalih 20 varijanti imaju bolje rezultate i stoga će ove odabrane varijante biti uključene u daljnju evaluaciju. U nastavku ovog poglavlja se uspoređuje propusnost navedenih varijanti dok se eksperimentalno variraju vrijednosti sljedećih parametara:

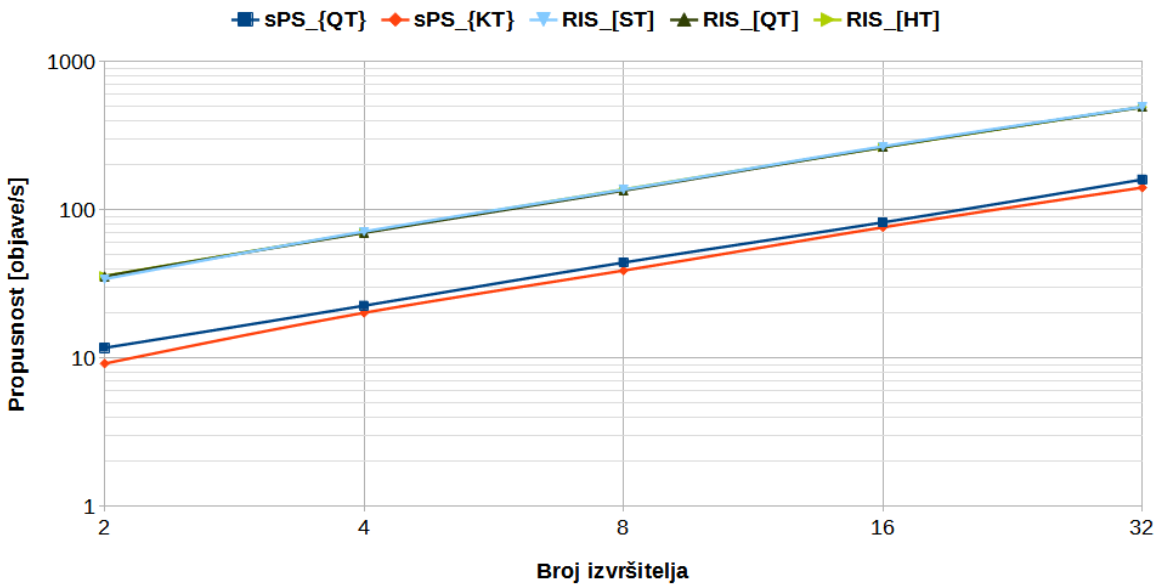
•broj izvršitelja
•broj particija pretplata
•broja pretplata
•brojem paralelnih Spark poslova

8.2.1 Eksperiment A - Usporedba odabranih varijanti algoritama sPS i RIS

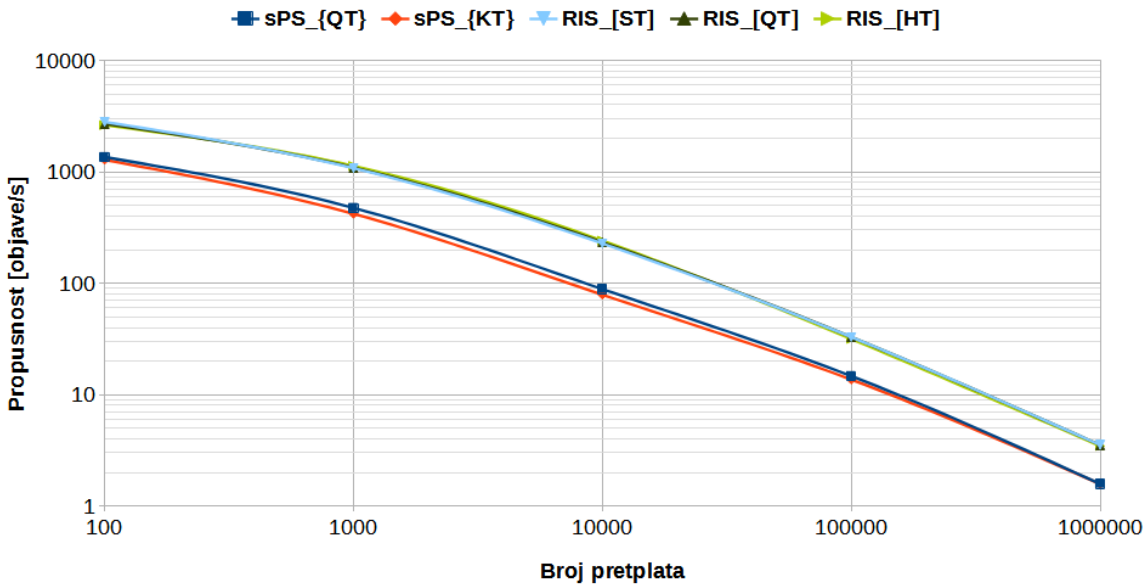


Slika 8.2: Povećanje broja particija pretplata

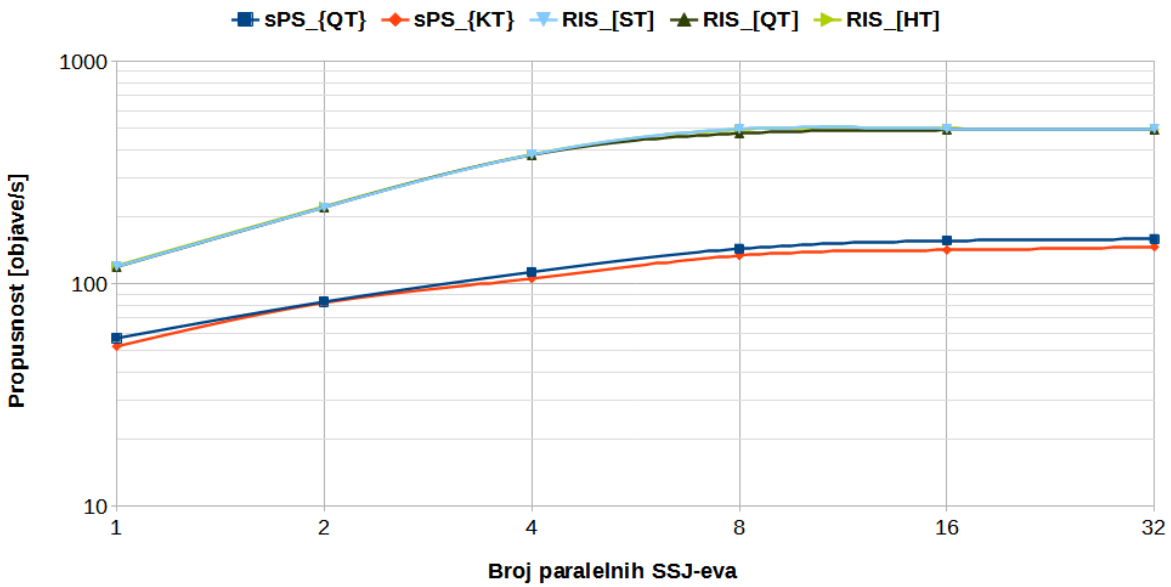
Na slikama od 8.2 do 8.5 prikazani su rezultati usporedbe algoritama odabranih varijanti algoritama sPS i RIS. Očito je da sPS_{QT} varijanta radi nešto bolji od sPS_{KT} varijante algoritma sPS, dok varijante algoritma RIS rade gotovo identično.



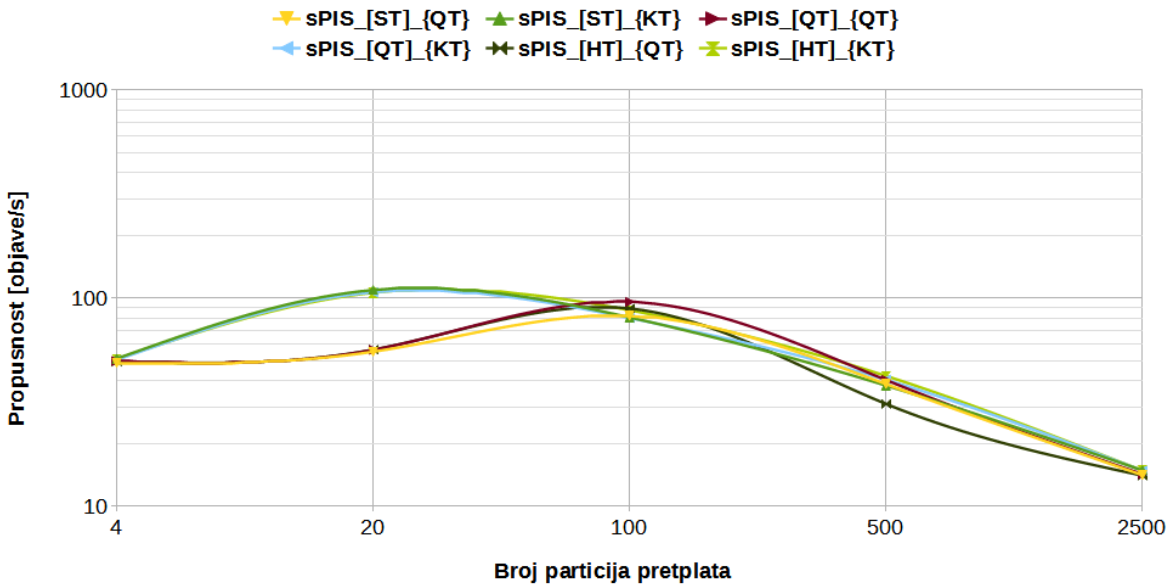
Slika 8.3: Povećanje broja izvršitelja



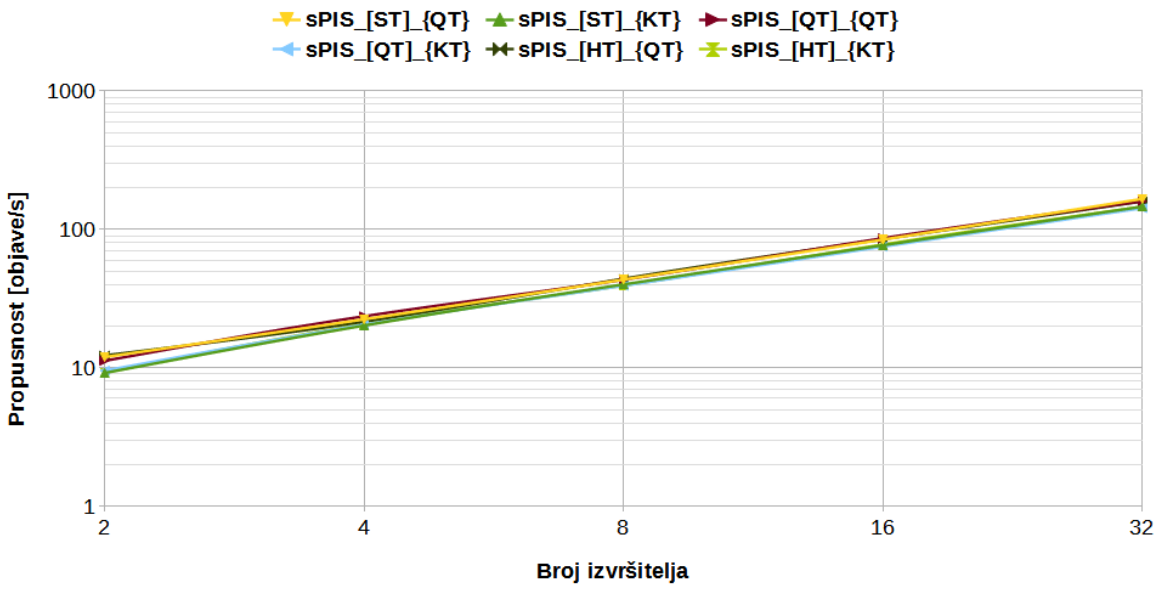
Slika 8.4: Povećanje broja pretplata



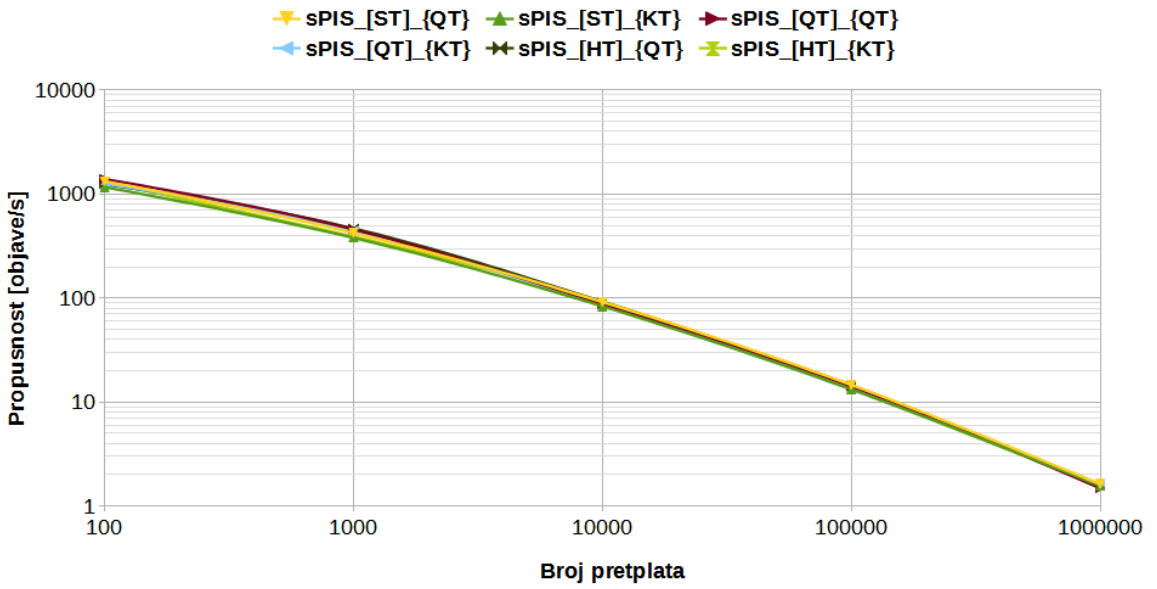
Slika 8.5: Povećanje broja paralelnih SSJ-eva



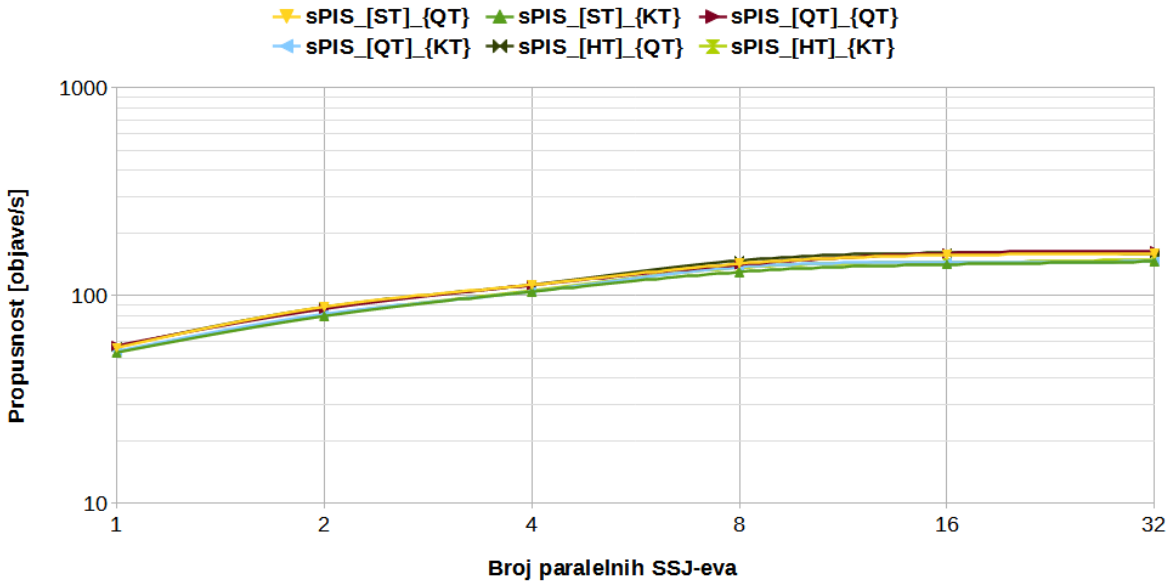
Slika 8.6: Povećanje broja particija pretplata



Slika 8.7: Povećanje broja izvršitelja



Slika 8.8: Povećanje broja pretplata



Slika 8.9: Povećanje broja paralelnih SSJ-eva

8.2.2 Eksperiment B - Usporedba odabranih varijanti algoritma sPIS

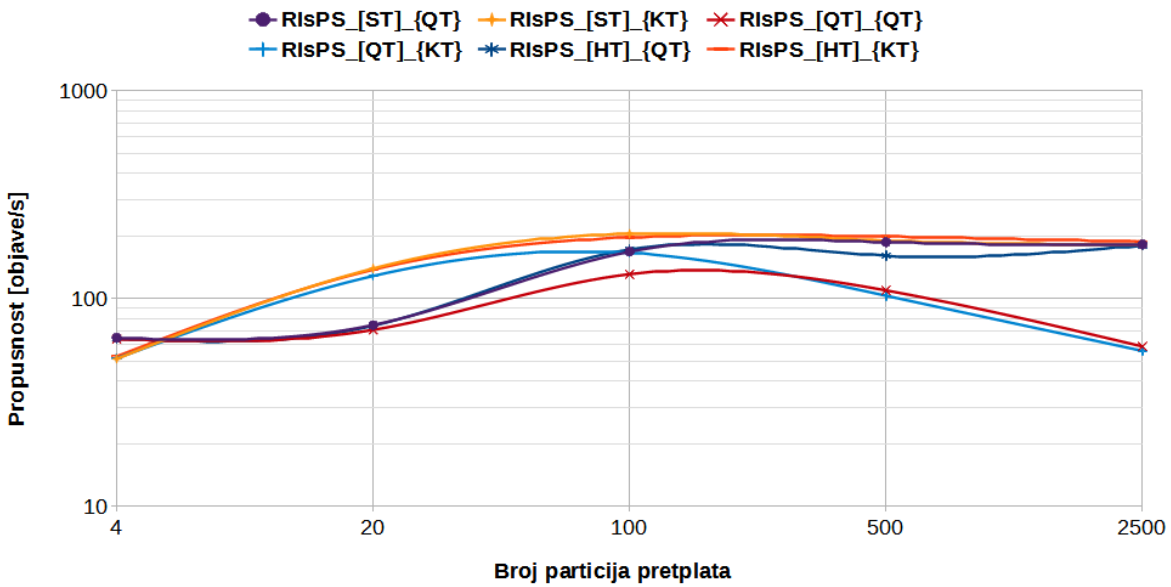
Na slikama od 8.6 do 8.9 prikazani su rezultati usporedbe odabranih varijanti algoritma sPIS. Može se zaključiti da su varijante algoritma sPIS s metodom particioniranja Quadtree ({QT}) uvijek nešto bolja od varijanti s metodom particioniranja KDB Tree ({KT}) što je u skladu s prethodnim zaključkom za varijante algoritma sPS.

8.2.3 Eksperiment C - Usporedba odabranih varijanti algoritma RIsPS

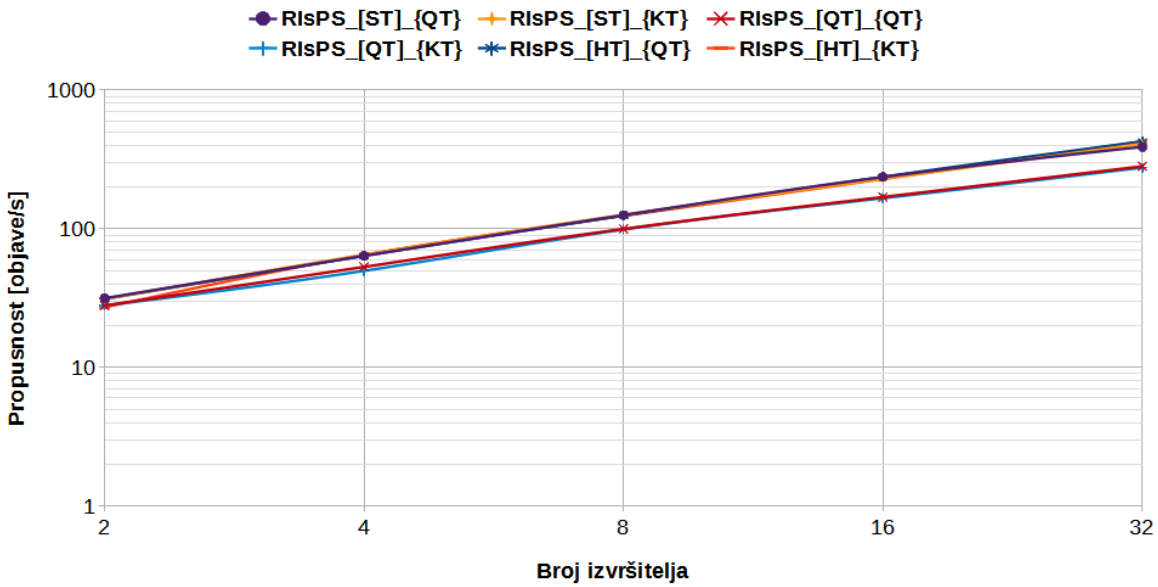
Na slikama od 8.10 do 8.13 prikazani su rezultati usporedbe odabranih varijanti algoritma RIsPS. Jasno je da su varijante algoritma RIsPS s indeksom Quadtree ([QT]) puno lošije od varijanti sa STR Tree ([ST]) i HPR Tree ([HP]) indeksom, osobito kada je broj particija velik. Nadalje je analiziran razlog zbog čega Quadtree ima lošije perfomanse od STR Tree ([ST]) i HPR Tree ([HP]) za veliki broj particija. Došlo se do saznanja da indeks Quadtree vraća znatno više pretplata kandidata, ali iznenađujuće je da se većina njih brzo filtrira kao one koje nisu zadovoljene tijekom primjene prostornog predikata. Posljedično nema značajne razlike među tim stablima kada se rezultat upita nad indeksom čuva na istom radnom čvoru kao u slučaju algoritma RIS. Međutim, u slučaju algoritma RIsPS spomenuti rezultat se šalje drugim izvršiteljima (tj. radnim čvorovima) zbog operacije `partitionBy` koja se događa prije operacije `join` na slici 7.10 i tako uzrokuje značajan pad performansi.

8.2.4 Eksperiment D - Usporedba odabranih varijanti algoritma RihPS

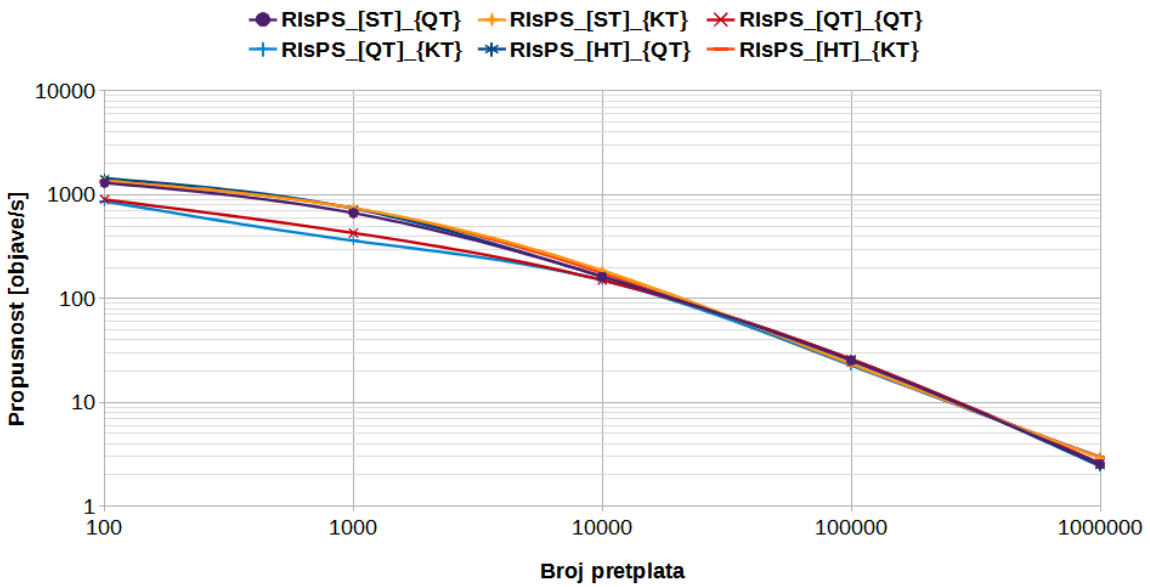
.



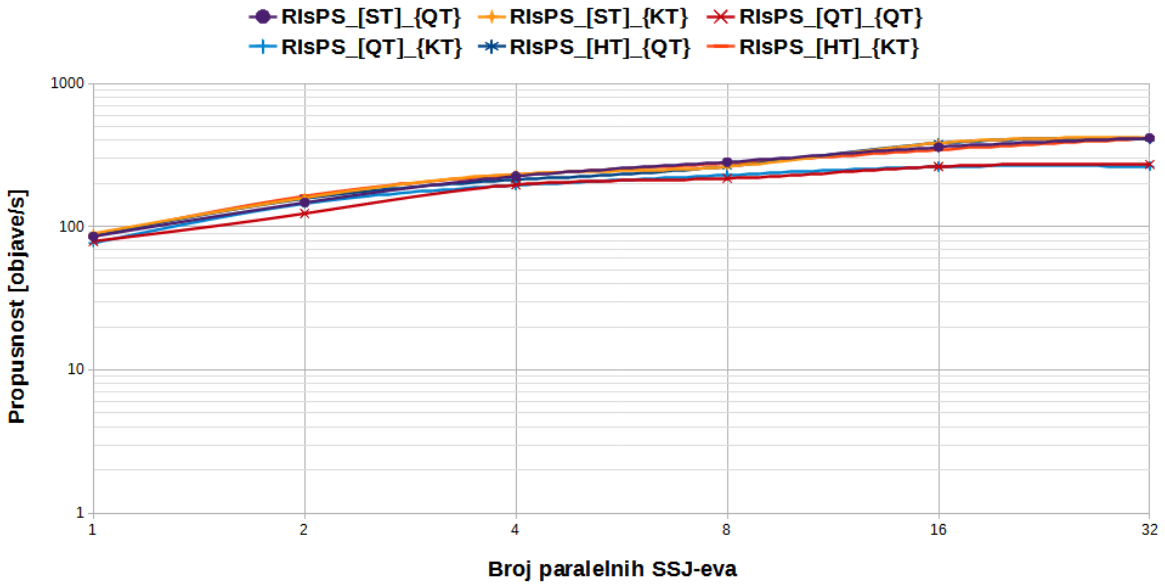
Slika 8.10: Povećanje broja particija pretplata



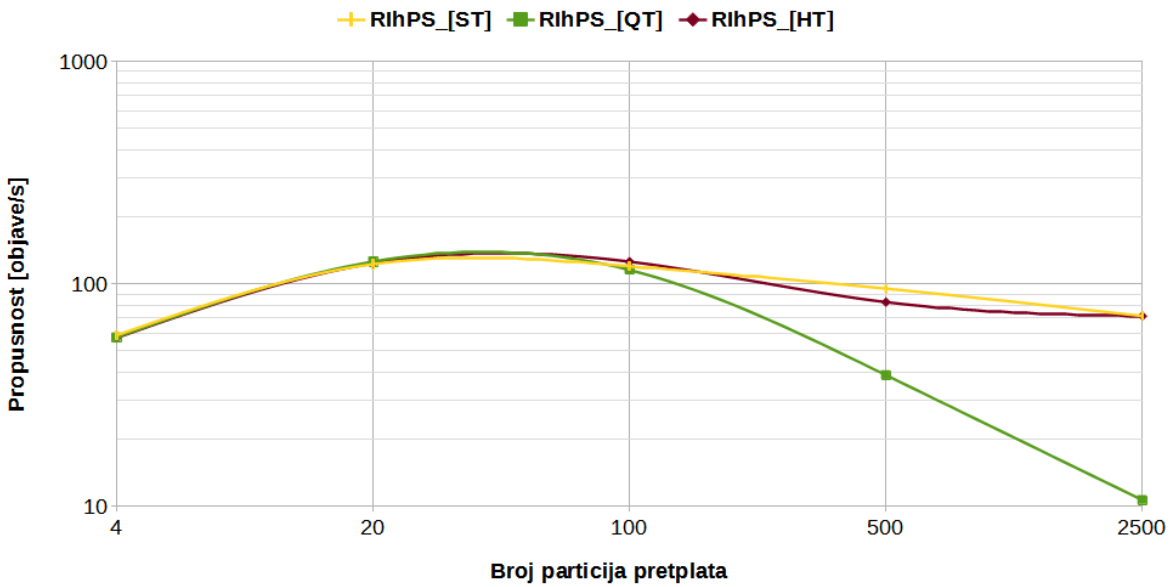
Slika 8.11: Povećanje broja izvršitelja



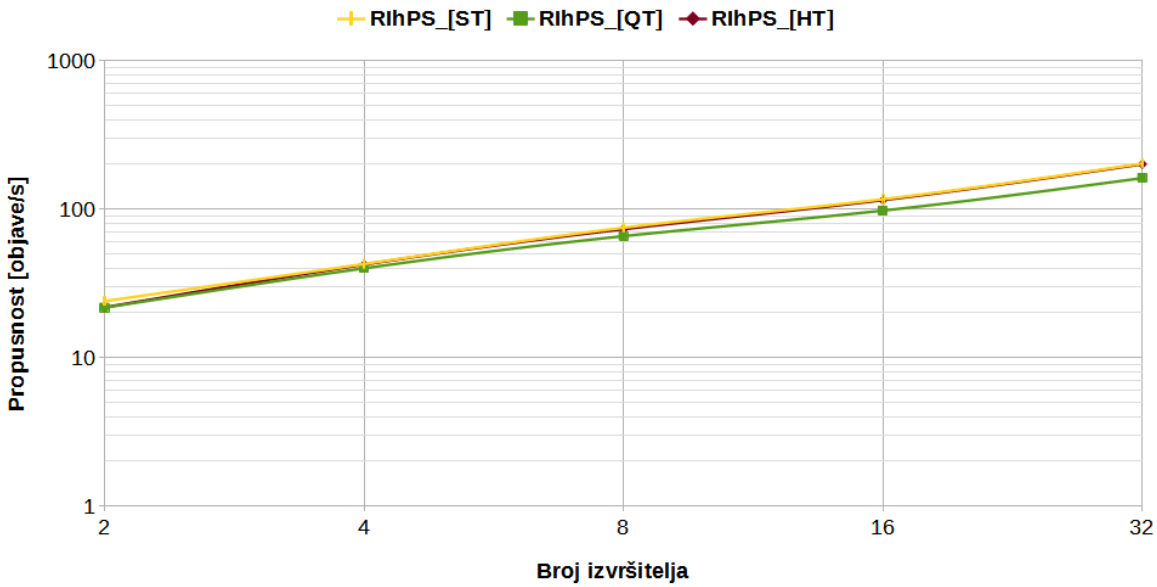
Slika 8.12: Povećanje broja pretplata



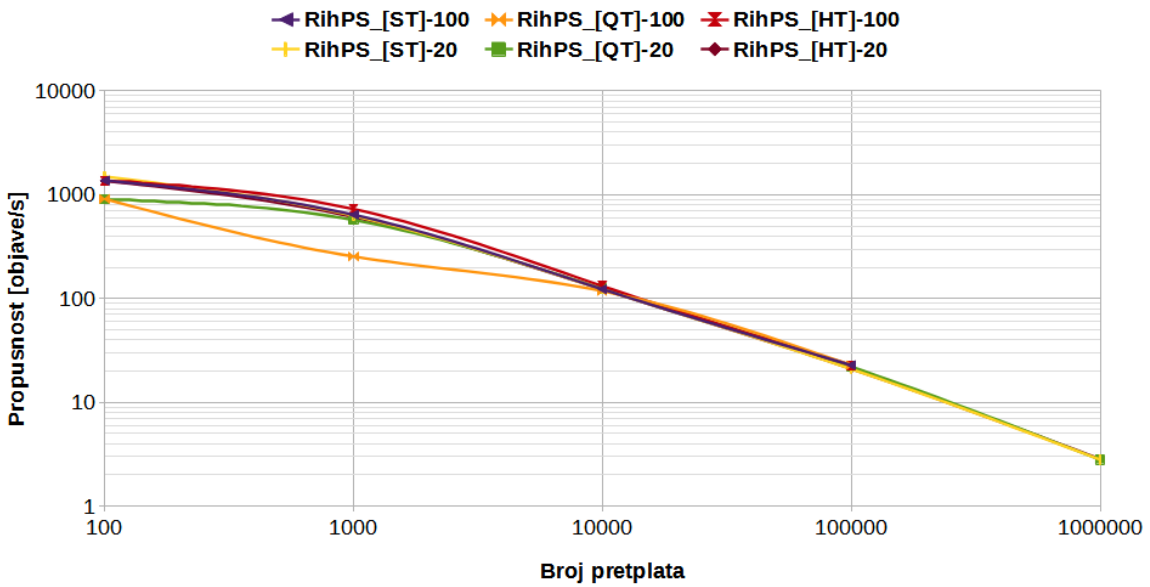
Slika 8.13: Povećanje broja paralelnih SSJ-eva



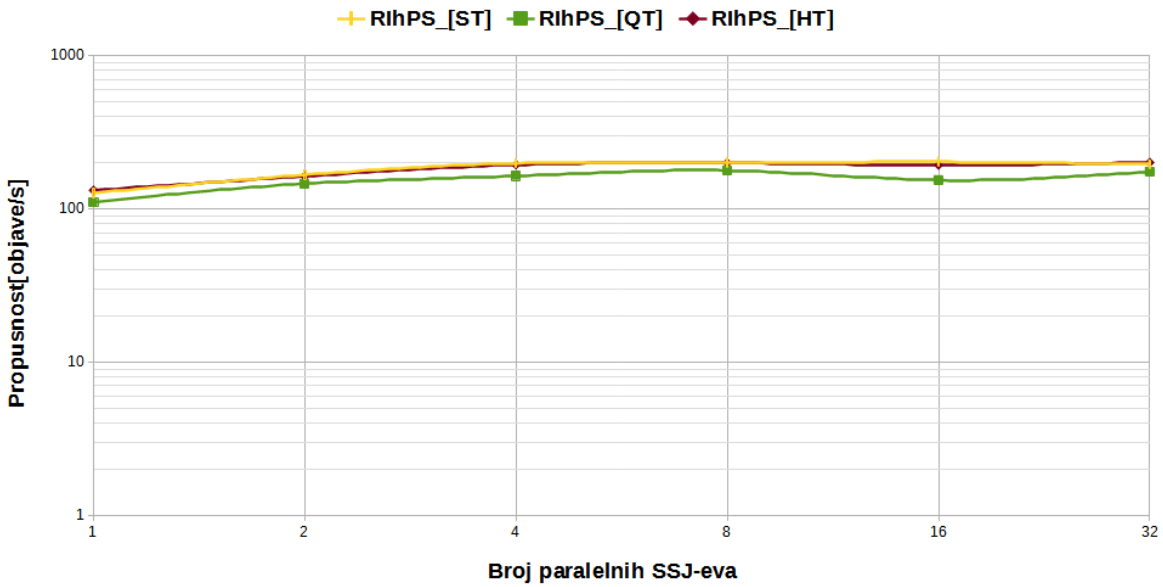
Slika 8.14: Povećanje broja particija pretplata



Slika 8.15: Povećanje broja izvršitelja



Slika 8.16: Povećanje broja pretplata



Slika 8.17: Povećanje broja paralelnih SSJ-eva

Na slikama od 8.14 do 8.17 prikazani su rezultati usporedbe odabranih varijanti algoritma RihPS. Varijanta RIhPS_[QT] lošija je od druge dvije varijante pogotovo kada je broj particija velik. Razlog tome je isti kao i za varijante algoritma RIsPS. Na slici 8.16 može se vidjeti da su varijante algoritma RIhPS uspoređene za 20 i 100 particija pretplata budući da potonja vrijednost uzrokuje probleme s nedostatkom memorije na čvoru upravljačkog programa Spark za milijun objava što će biti objašnjeno u odjeljku 8.3. Vidljivo je da nema značajnih razlika u propusnosti za ove dvije vrijednosti particija pretplata što je u skladu sa slikom 8.19.

8.3 Usporedba predloženih algoritama

Rezultati usporedbe odabranih varijanti predloženih algoritama u prethodnom potpoglavlju pokazuju da su varijante RIS_[HT], sPS_{QT}, sPIS_[QT]_{QT}, RIsPS_[HT]_{KT} i RIhPS_[HT] među najboljima po performansama i stoga se u eksperimentalnoj evaluaciji u ovom i sljedećem potpoglavlju one koriste kao predstavnici predloženih algoritama.

U ovom poglavlju uspoređuju se predloženi algoritmi (tj. varijante koje su predstavnici predloženih algoritama) usporedbe objava i pretplata u raspodijeljenom geoprostornom sustavu objavi-pretplati dok se eksperimentalno variraju vrijednosti sljedećih parametara:

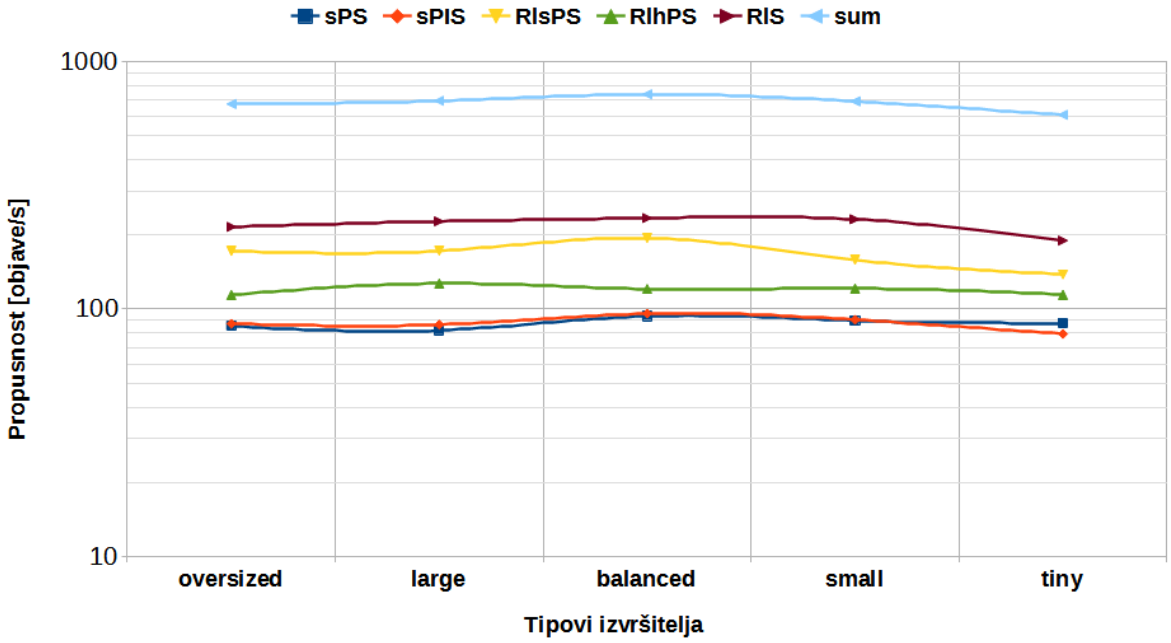
- vrsta Sparkovih izvršitelja
- broj particija pretplata
- broj paralelnih Sparkovih poslova
- broj Kafka particija
- broj objava
- broj Sparkovih izvršitelja
- broj pretplata
- postotak to čkastih objava
- vrsta prostornih predikata pretplata

Rezultati ove eksperimentalne evaluacije prikazani su na slikama od 8.18 do 8.29.

8.3.1 Eksperiment E - Propusnost u odnosu na tipove izvršitelja

U prvom eksperimentu evaluira se propusnost algoritama u odnosu na različite tipove Spark izvršitelja. Budući da svaki radni čvor u grozdu ima 64 GB radne memorije i 8 procesorskih jezgri, moguće je te resurse različito podijeliti među izvršiteljima. U ovom eksperimentu analizira se 5 različitih tipova izvršitelja prikazanih u tablici 8.3. Na slici 8.18 prikazuje se propusnost

različitih algoritama, kao i njihov zbroj za različite tipove izvršitelja. Kao što je vidljivo izvršitelji tipa *balanced* su se pokazali kao najbolji pa su korišteni kao podrazumijevani tip izvršitelja u ostalim eksperimentima. Međutim, razlike između izvršitelja tipa *oversized*, *large* i *balanced* nisu bile toliko značajne.

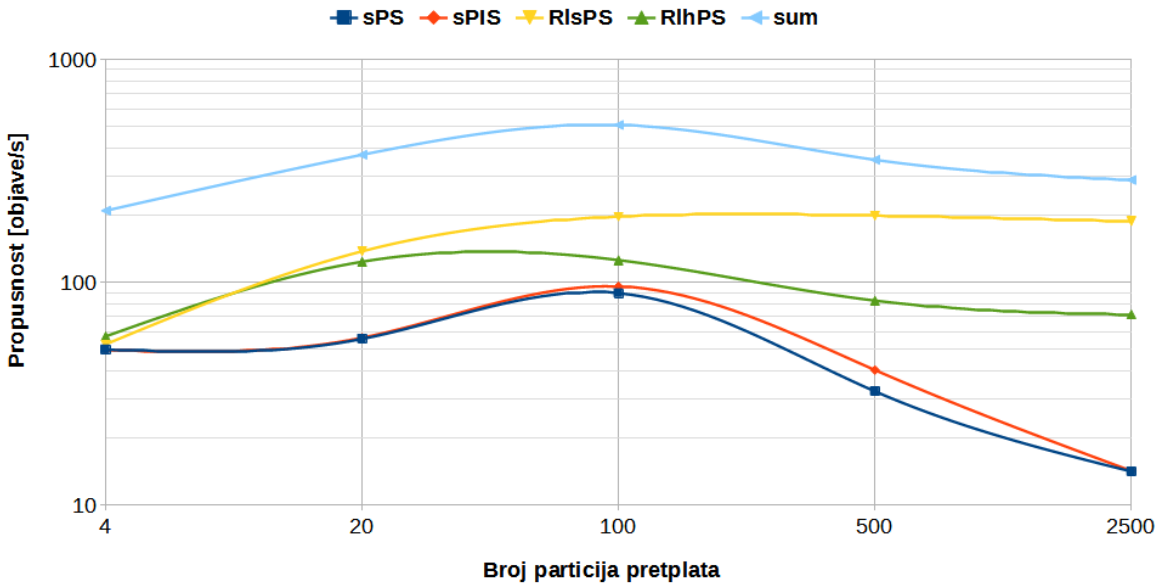


Slika 8.18: Propusnost u odnosu na tipove izvršitelja

8.3.2 Eksperiment F - Propusnost prema broju particija pretplata

U drugom eksperimentu identificira se broj particija pretplata za koje se može postići najveća propusnost. Mijenja se broj particija eksponencijalno od 4 do 2500 uvećanjem broja particija iz prethodnog koraka faktorom 5. Ovaj put je isključen algoritam RIS jer on ne particionira pretplate. Rezultati su prikazani na slici 8.19. Vidljivo je da većina algoritama najbolje radi za 100 particija i stoga je ova vrijednost korištena kao podrazumijevana u drugim eksperimentima. Iznimka je algoritam RIsPS, koji djeluje slično ili nešto bolje u slučaju velikog broja particija pretplata. To ukazuje da algoritam RIsPS ima izvrstan potencijal za horizontalno skaliranje budući da bi se broj particija pretplata trebao povećavati s brojem radnih čvorova. Za sve strategije, broj pretplata unutar particija se smanjuje kada se broj particija povećava i očekivalo bi se da će to imati pozitivan utjecaj na propusnost. Međutim, umjesto toga vidljiv je pad propusnosti za algoritme sPS, sPIS i RIhPS. U slučaju algoritama sPS i sPIS to se može objasniti povećanom složenošću identifikacije particija kojima objava pripada i povećanim brojem identificiranih particija što oboje ima negativan utjecaj na propusnost. To nije slučaj s algoritmima RIsPS i RIhPS jer oni ne identificiraju particije kojima objava pripada, već traže odgovarajuće

pretplate unutar njihovih particija. Međutim, dok algoritam RIsPS mora izvršiti takvo pretraživanje unutar particija koje su prostorno bliske, algoritam RIhPS mora pretraživati nepovezane particije. Stoga, kada se znatno poveća broj particija pretplata, broj particija koje algoritam RIhPS mora provjeriti je također znatno povećan, ali to isto nije slučaj s algoritmom RIsPS. Zaključno može se vidjeti da je najbolji algoritam RIsPS pa zatim algoritam RIhPS, dok su algoritmi sPS i sPIS puno lošiji i imaju vrlo slične performanse.

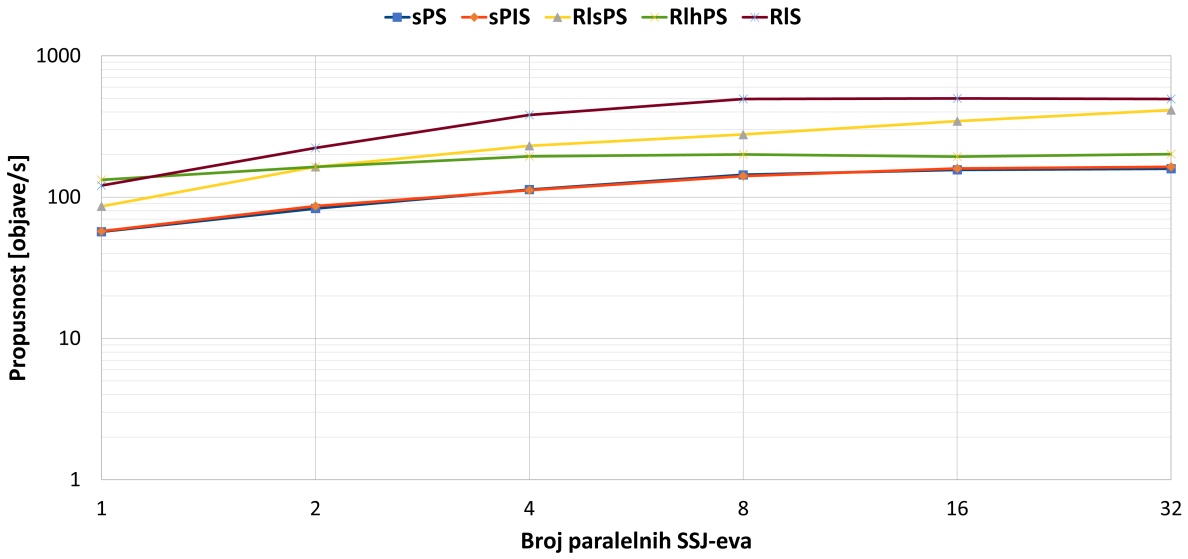


Slika 8.19: Propusnost prema broju particija pretplata

8.3.3 Eksperiment G - Propusnost i kašnjenje prema broju paralelnih poslova u sustavu *Spark Streaming*

U trećem eksperimentu identificira se broj paralelnih poslova u sustavu *Spark Streaming* (engl. *Spark Streaming Jobs, SSJ*) za koje se postiže najbolja ravnoteža između propusnosti i kašnjenja. Kada je u bilo kojem trenutku dopušteno izvršavanje samo jednog SSJ-a, to rezultira slijednim, a ne paralelnim izvršavanjem SSJ-eva. To znači da se mikro-skupina podataka (engl. *micro-batch*) obrađuje slijedno i da je iskoristivost (engl. *utilization*) Sparkovih izvršitelja vrlo niska. Kao izravna posljedica tako niske iskoristivosti dobiva se i vrlo niska propusnost. Međutim, budući da su izvršitelji uglavnom slobodni oni mogu odmah početi s obradom zadanih zadataka, što rezultira vrlo malim kašnjenjem. Naprotiv, ukoliko se poveća broj istodobnih SSJ-eva mikro-skupina podataka će se početi obrađivati paralelno, što će povećati i propusnost i kašnjenje. Propusnost i kašnjenje se povećava jer zadaci dani izvršiteljima moraju čekati dok ne završi obrada prethodno zadanih zadataka. Takav porast propusnosti i kašnjenja uz sve veći broj istodobnih SSJ-eva je prikazan na slikama 8.20 i 8.21. Propusnost prvo brzo raste kada

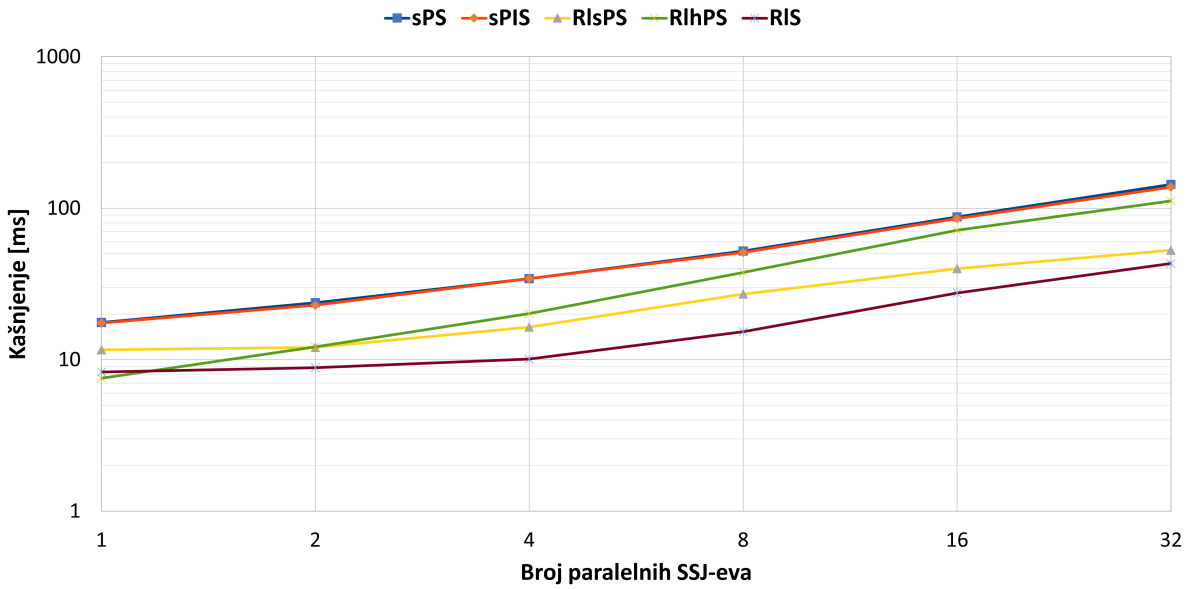
je iskoristivost niska, ali prestaje rasti kada se postigne potpuna iskoristivost. Naprotiv, kašnjenje najprije raste sporo, a kasnije počinje brzo rasti jer se povećava broj zadataka koji čeka na obradu. U ostalim eksperimentima se kao podrazumijevana vrijednost paralelnih SSJ-eva koristi 4 jer se za ovu vrijednost čini da dobro uravnotežuje propusnost i kašnjenje. Kao što je vidljivo na slikama, najbolji algoritam je RIS jer postiže najveću propusnost i najmanje kašnjenje, dok su algoritmi sPS i sPIS najlošiji. Algoritmi RIsPS i RIhPS su u sredini tako da je algoritam RIhPS bolji kada je broj istodobnih SSJ-eva nizak, dok je algoritam RIsPS bolji inače. Osim toga, propusnost aloritma RIsPS povećava se mnogo većom intenzitetom nego kod drugih algoritama. Kao posljedica toga, razlika u propusnosti između algoritama RIS i RIsPS postaje vrlo mala u slučaju velikog broja istodobnih SSJ-eva.



Slika 8.20: Propusnost prema broju paralelnih SSJ-eva

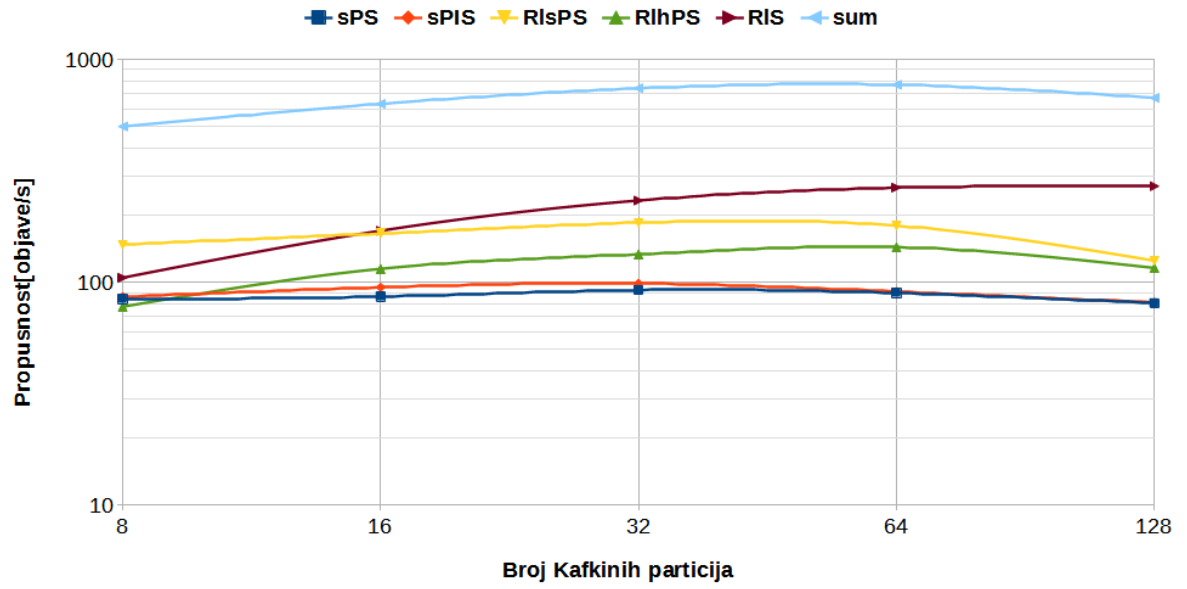
8.3.4 Eksperiment H - Propusnost prema broju Kafkinih particija

U četvrtom eksperimentu evaluira se propusnost uz povećanje broja Kafkinih particija, kao što je prikazano na slici 8.22. Vidljivo je da većina algoritama radi najbolje za 32 Kafkine particije i stoga se ova vrijednost koristi kao podrazumijevana u ostalim eksperimentima. Evidentno je da algoritam RIS postiže veću propusnost za veći broj Kafkinih particija, dok kod ostalih algoritama to nije slučaj. To je i očekivano budući da algoritam RIS obrađuje svaku objavu na jednom izvršitelju i stoga ne vrši operaciju miješanja `shuffle`. Iz navedenog razloga postiže bolji rezultat kada svaka procesorska jezgra obrađuje vlastitu Kafkinu particiju. Budući da postoje 3 procesorske jezgre po svakom izvršitelju tipa *balanced*, kao što je prikazano u tablici 8.3, stoga postoji ukupno $3x32 = 96$ procesorskih jezgri na raspolaganju u čitavom grozdu.



Slika 8.21: Kašnjenje prema broju paralelnih SSJ-eva

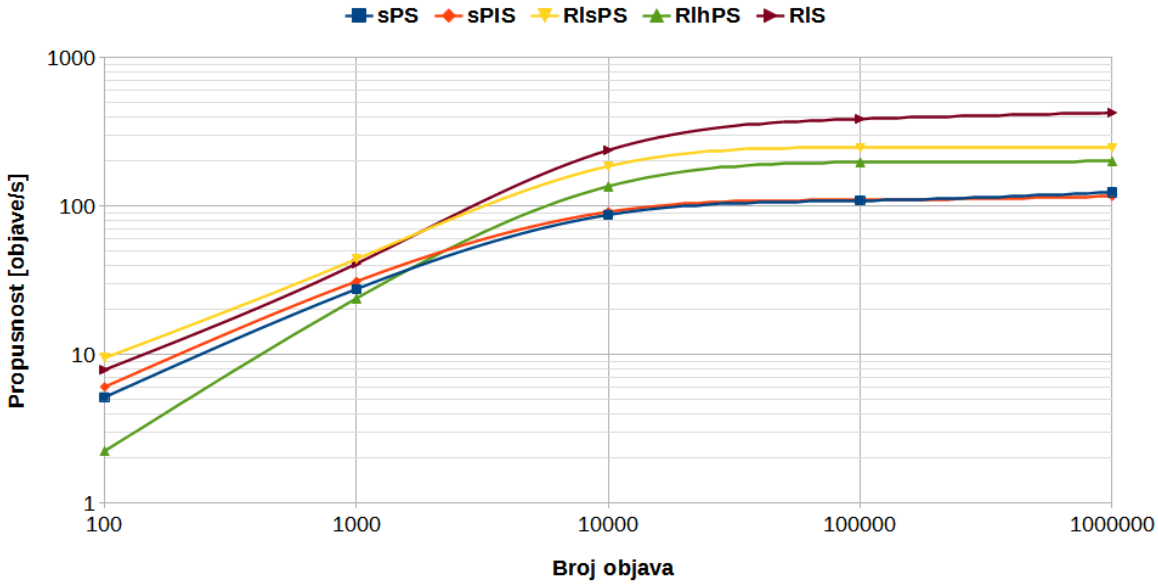
Algoritam RIS ima najbolje rezultate za 128 Kafkinih particija budući da je tada broj dostupnih jezgri u grozdu najbliži broju Kafkinih particija. Međutim, za druge algoritme bolje je po izvršitelju namijeniti samo jednu procesorsku jezgru za dohvaćanje objava s Kafkine teme, dok su preostale dvije jezgre tada slobodne za obavljanje mnogo složenijih zadataka, kao što je operacija `shuffle`.



Slika 8.22: Propusnost prema broju Kafkinih particija

8.3.5 Eksperiment I - Propusnost prema broju objava

U petom eksperimentu evaluira se propusnost uz povećanje broja objava, kao što je prikazano na slici 8.23. Kao što je vidljivo, povećanjem broja objava povećava se i propusnost, ali se intenzitet povećanja smanjuje budući da utjecaj dodatne obrade (engl. *overhead*) postaje sve manji. Međutim, kako eksperimenti traju dulje za veći broj objava, kao podrazumijevani broj objava u ostalim eksperimentima je odabrana vrijednost od 10.000 objava jer ona daje propusnost koja je usporediva s većim vrijednostima, ali značajno ubrzava eksperimente. Opet je najbolji algoritam RIS, a slijede algoritmi RIsPS i RIhPS, dok su algoritmi sPS i sPIS najlošiji i imaju vrlo slične performance.

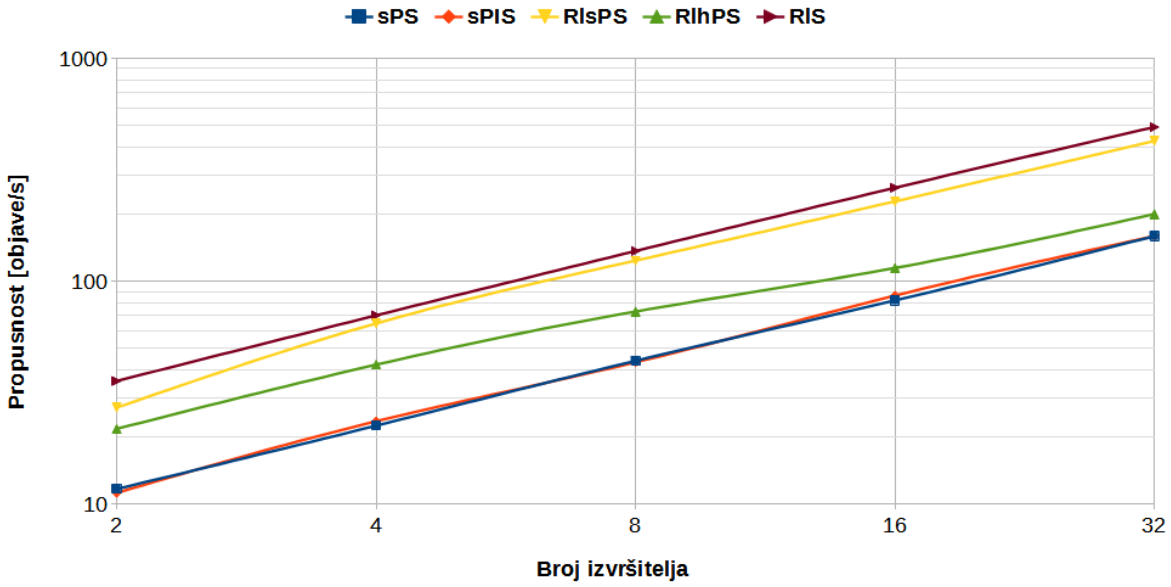


Slika 8.23: Propusnost prema broju objava

8.3.6 Eksperiment J - Propusnost prema broju izvršitelja

U šestom eksperimentu evaluira se skalabilnost algoritama uz povećanje broja izvršitelja tipa *balanced*. Ova evaluacija pokazuje kako se algoritmi horizontalno skaliraju u veće računalne grozdove. Budući da iskoristivost izvršitelja opada kada se njihov broj povećava u slučaju zadanog broja od 4 istodobna SSJ-a, u ovom eksperimentu je povećan broj istodobnih SSJ-eva na 32 kako bi iskoristivost izvršitelja bila gotovo maksimalna tijekom eksperimenta. Osim toga povećan je i broj objava u ovom eksperimentu na 100.000 kako bismo testirali dugotrajno razdoblje koje smanjuje utjecaj dodatne obrade (engl. *overhead*), kao što je prethodno objašnjeno. Na slici 8.24 vidljivo je da propusnost gotovo svih algoritama raste linearno za sve veći broj izvršitelja i stoga je vidljivo da su oni skalabilni. Jedina iznimka je algoritam RIhPS koji se sublinearno povećava između 4 i 16 izvršitelja, a linearno inače. Opet, najbolji algoritam je RIS, ali je samo malo bolji od algoritma RIsPS zbog 32 istodobna SSJ-a. Algoritam RIhPS puno je

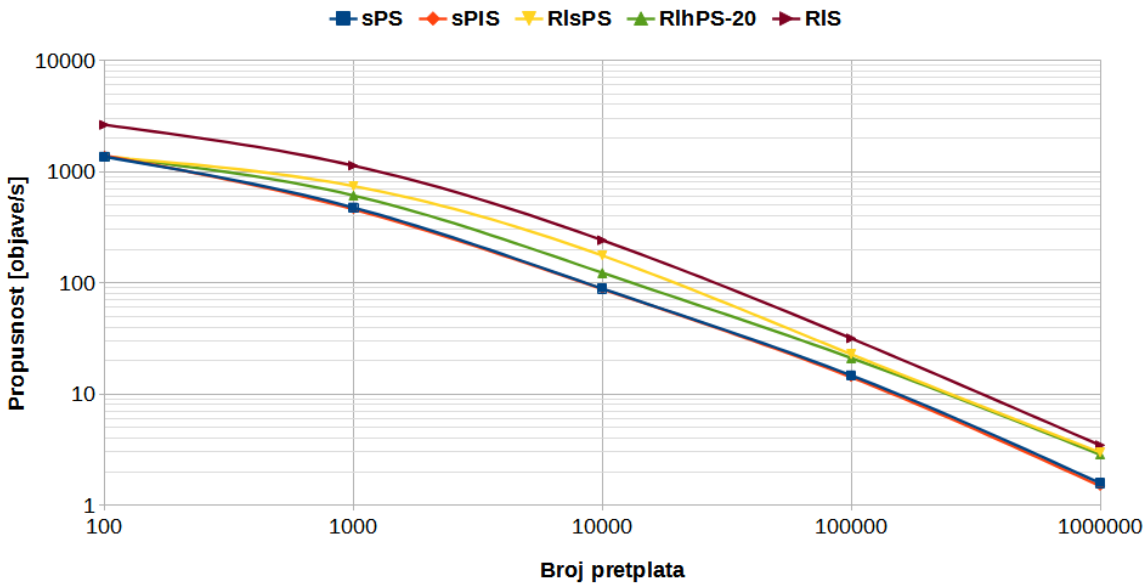
lošiji od algoritm RIsPS, a za veliki broj izvršitelja vrlo je sličan algoritmima sPS i sPIS, koje imaju najlošije rezultate.



Slika 8.24: Propusnost prema broju izvršitelja

8.3.7 Eksperiment K - Propusnost prema broju pretplata

U sedmom eksperimentu evaluira se kako na propusnost algoritama utječe sve veći broj pretplata. Na slici 8.25 vidljivo je da algoritmi pokazuju slično ponašanje: za povećanje broja pretplata, propusnost prvo počinje polako padati, a zatim nastavlja linearnim padom. Međutim stopa pada je nešto manja od stope rasta pretplata što pokazuje da su varijante skalabilne uz povećanje broja pretplata. Algoritam RIhPS sa zadanim brojem od 100 particija pretplata nije se mogao evaluirati za milijun pretplata. Stoga je postavljen broj particija za ovaj algoritam na 20 da bi se mogao završiti ovaj eksperiment. Problem je u potrebnoj memoriji za operaciju `join`, što će se dalje analizirati u sljedeća dva eksperimenta, gdje se evaluira potrošnja memorije. Ponovno je najbolji algoritam RIS, a slijede algoritmi RIsPS i RIhPS, dok su algoritmi sPS i sPIS najlošiji i imaju vrlo slične performanse.



Slika 8.25: Propusnost prema broju pretplata

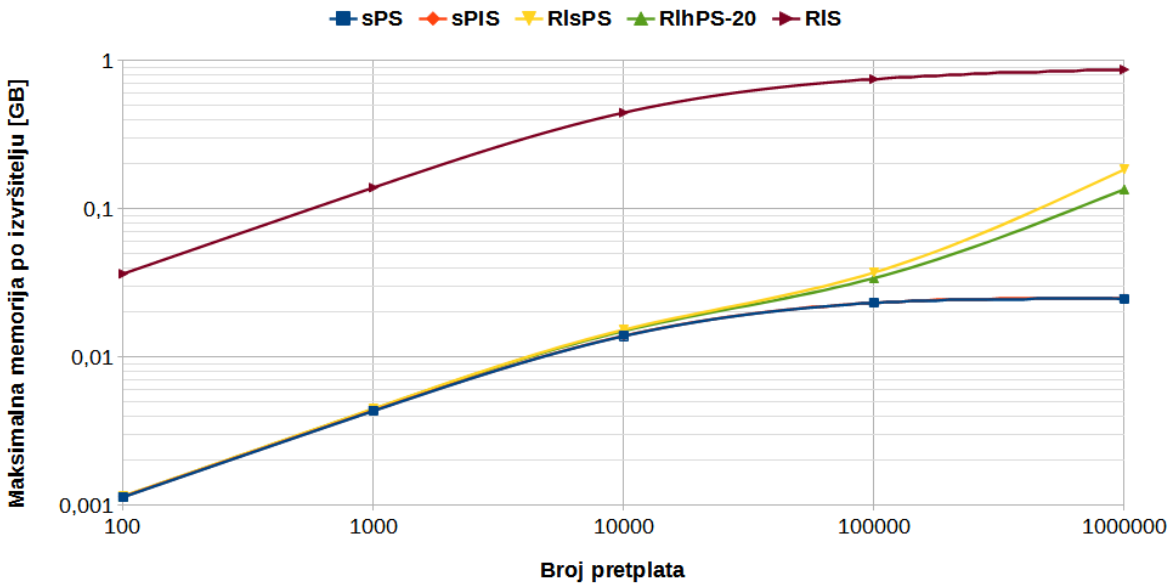
8.3.8 Eksperiment L - Potrošnja memorije uz promjenu broja pretplata

U osmom eksperimentu evaluira se maksimalna potrošnja memorije (engl. *Maximal Memory Consumption, MMC*) algoritama uz povećanje broja pretplata. Analizira se MMC po Sparkovom izvršitelju za:

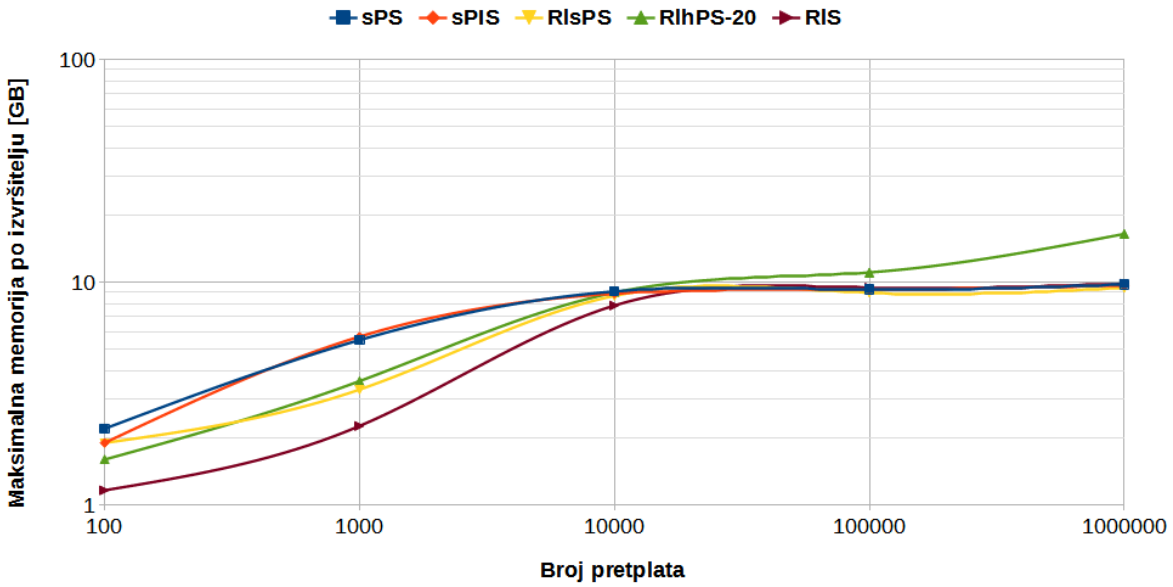
- *statičku memoriju* u kojoj su pohranjene pretplate i povezane strukture podataka (tj. indeks pretplata, metode particioniranja i/ili pretplate)

- *dinamičku memoriju* koja se koristi za izvođenje operacija u cjevovodu tijekom obrade objava

Broj poligona u korištenom skupu podataka [28] bio je 120, 2.736 i 9.232 za poštanske sektore, okruge i područja Ujedinjenog Kraljevstva. Kako se ovaj skup podataka koristio za generiranje pretplata, nije se mogao dobiti veliki broj jedinstvenih geoprostornih pretplata. Prosječna veličina pretplate u eksperimenta bila je oko 0,25 MB. Da su pretplate jedinstvene očekivalo bi se da njihova veličina bude oko 25 MB, 250 MB, 2,5 GB, 25 GB i 250 GB za 100, 1.000, 10.000, 100.000 i 1.000.000 pretplata. Međutim, kao što je vidljivo na slici 8.26 veličina pretplata za algoritam RIS u praksi je manje od 1 GB za 1.000.000 pretplata. Eksperimenti se ne bi mogli izvoditi za algoritam RIS da su pretplate bile jedinstvene, jer Sparkovi izvršitelji tipa *balanced* imaju samo 24 GB RAM memorije na raspolaganju, kao što je prikazano u tablici 8.3. Na slici 8.26 vidljivo je da algoritam RIS zahtijeva do 32 puta više statičke memorije od ostalih strategija budući da se pretplate repliciraju na 32 izvršitelja.



Slika 8.26: Potrošnja memorije (statička) uz povećanje broja pretplata



Slika 8.27: Potrošnja memorije (dinamička) uz povećanje broja pretplata

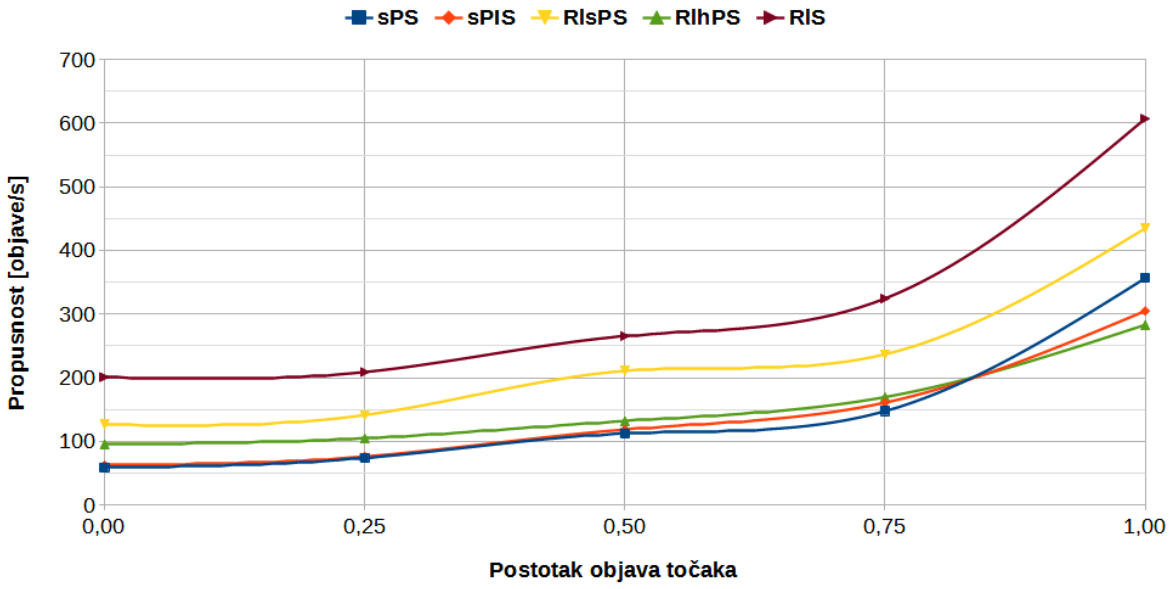
Algoritmi RIsPS i RIhPS zahtijevaju više statičke memorije za veliki broj pretplata budući da replicirani prostorni indeks parova pretplata "ID particije - ID pretplate" postaje veći i oni stoga troše više memorije nego algoritmi sPS i sPIS. Kao što je već spomenuto, algoritam RIhPS sa zadanim brojem od 100 particija pretplata nije se mogao evaluirati za milijun pretplata zbog problema s nedostatkom memorije tijekom operacije `join` na strani Sparkovog pokretača posla (engl. *driver*), koji je imao samo 32 GB radne memorije. Kao što je prethodno objašnjeno parovi trebaju biti particionirani prije izvođenja operacije `join` tako da se oni s istim ID-jem particije šalju na isti čvor. Uočeno je da je broj parova (po objavi) jednak broju particija u slučaju algoritma RIhPS te nekoliko puta manji u slučaju algoritma RIsPS. To je izravna posljedica

korištenih metoda particioniranja. Budući da se operacija `join` na ovim parovima djelomično događa na čvoru Sparkovog pokretača posla, koji je odgovoran za spajanje djelomičnih rezultata, veliki broj ovih parova uzrokuje probleme s memorijom za taj čvor. Kako bi se izbjegao ovaj problem ponovno je smanjen broj particija pretplata za algoritam RIhPS sa 100 na 20.

Slika 8.27 prikazuje MMC za dinamičku memoriju na izvršiteljima uz povećanje broja pretplata. Vidljivo je da se skalira sublinearno za sve algoritme što pokazuje njihovu skalabilnost. Dodatno se vidi da se za sve algoritme osim algoritam RihPS asimptotski približava vrijednosti 10 GB koja stoga predstavlja gornju granicu. Ponašanje algoritma RIhPS opet je povezano s prethodno objašnjenim problemom s operacijom `join`.

8.3.9 Eksperiment M - Propusnost uz povećanje postotka točkastih objava

U devetom eksperimentu ocjenjena je propusnost različitih algoritama uz povećanje postotka objava koje su geoprostorne točke. Rezultati su prikazani na slici 8.28 s napomenom da je ljestvica ordinate linearna u ovom eksperimentu. Može se zaključiti da se propusnost za sve algoritme značajno povećava povećanjem postotka točkastih objava. Vidimo da algoritam RIhPS postaje najlošiji kada su sve objave točke.

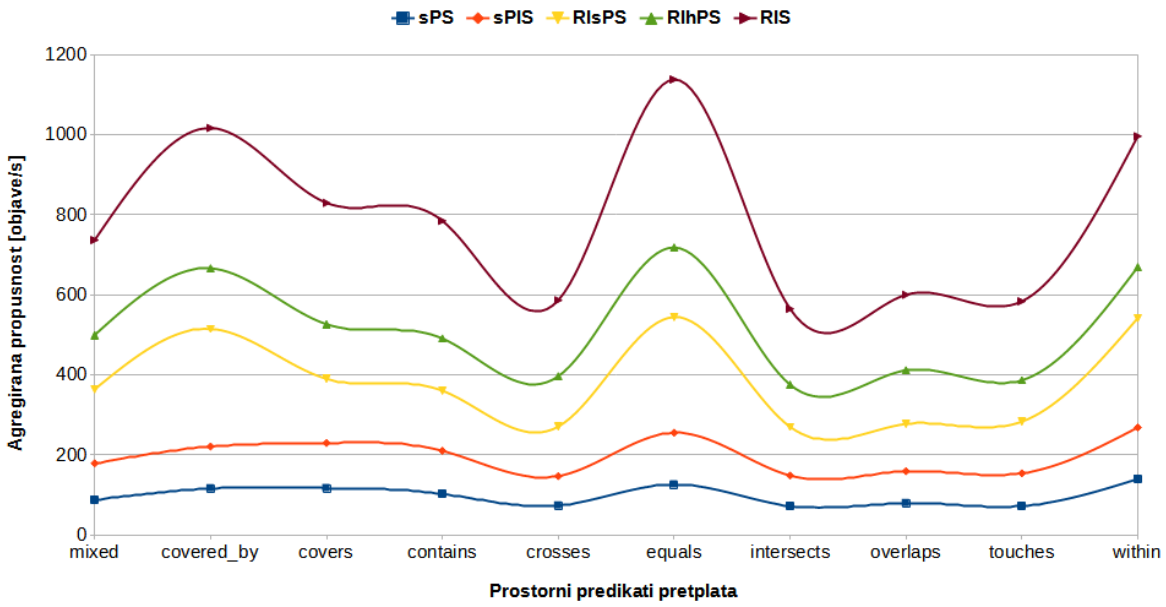


Slika 8.28: Propusnost promjenom postotka točkastih objava

8.3.10 Eksperiment N - Usporedba složenosti različitih prostornih predikata

U desetom eksperimentu uspoređena je složenost različitih prostornih predikata unutar pretplata. Na slici 8.29 se vidi propusnost različitih algoritama za različite prostorne predikate.

Ljestvica ordinate je ponovno linearna u ovom eksperimentu. Miješani (engl. *mixed*) predikat je kombinacija jednakog omjera svih ostalih predikata, a koristi se kao podrazumijevani predikat u ostalim eksperimentima. Kao što je vidljivo, prostorni predikati su podijeljeni prema njihovoj složenosti u tri skupine. Manje složeni predikati su `coveredby`, `equals` and `within`. U sredini su predikati `covers` i `contain`, dok su predikati `crosses`, `intersects`, `overlaps` and `touches` najsloženiji.

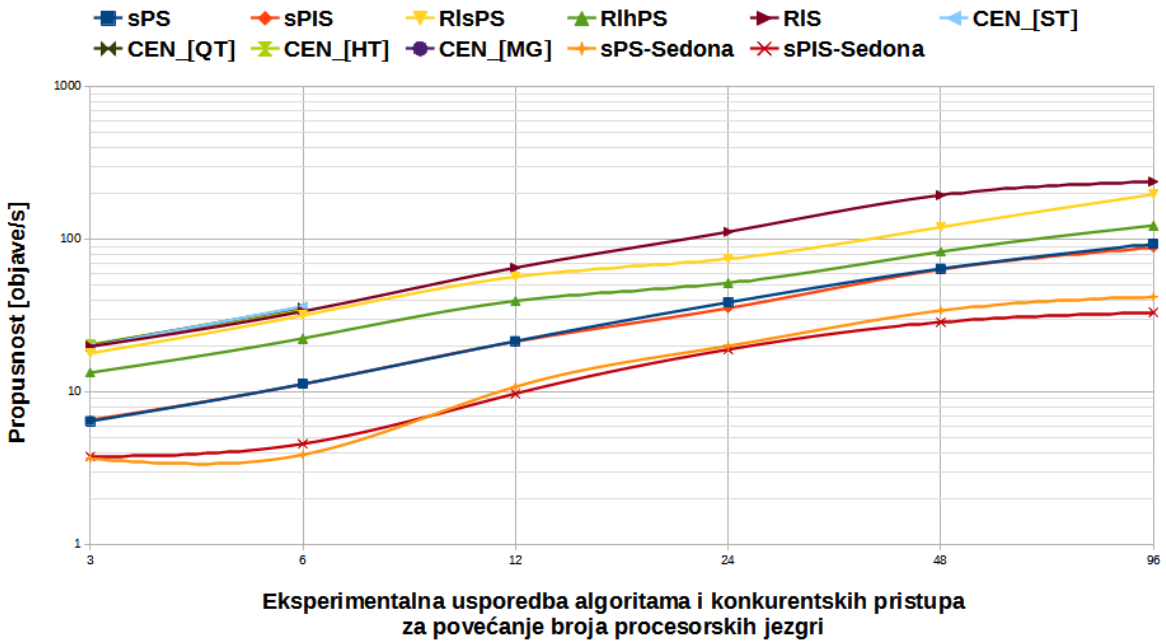


Slika 8.29: Propusnost različitih strategija za različite prostorne predikate pretplata

8.4 Usporedba predloženih algoritama i konkurentskih pristupa

U ovom poglavlju eksperimentalno se uspoređuju predloženi raspodijeljeni algoritmi s centraliziranim varijantama koje se ne izvode na grozdu i predloženim raspodijeljenim algoritmima sPS i sPIS implementiranim pomoću upita za prostorno spajanje iz knjižnice Apache Sedona o kojima se raspravlja u odjeljku 8.1. Ove centralizirane varijante (CEN) izvode se kao Javini procesi na jednom računalu. Oni indeksiraju pretplate u memoriji i istodobno obrađuju dolazne objave iz odgovarajuće Kafkine teme. Bitno je naglasiti da nije moguće imati Sparkove izvršitelje u centraliziranom slučaju pa se stoga na slici 8.24 na apcisi prikazuje broj dostupnih procesorskih jezgri kako bi centralizirane i raspodijeljene varijante bile međusobno usporedive. U ovom eksperimentu analiziraju se i evaluiraju četiri centralizirane varijante: CEN [ST], CEN [QT], CEN [HR] i CEN [MG]. Prve tri od ovih varijanti kao prostorni indeks koriste stabla STR Tree, Quadtree ili HPR Tree, dok zadnja varijanta koristi vlastitu implementaciju višerazinske prostorne rešetke (engl. *Multilevel Grid Spatial Index*) iz [16]. Dakle, CEN [ST] i CEN [QT]

varijante su implementacije koje koriste stabla STR tree i Quadtree iz [31], dok je CEN [MG] implementacija pristupa iz [30]. Korišten je jedan od radnih čvorova za pokretanje centraliziranih varijanti i budući da čvorovi imaju samo 8 jezgri na raspolaganju nije se mogla procijeniti njihova propusnost za veći broj procesorskih jezgri što se može vidjeti na slici 8.30.



Slika 8.30: Eksperimentalna usporedba algoritama i konkurentskih pristupa za povećanje broja procesorskih jezgri

Očito je da algoritam RIS ima gotovo zanemarive troškove dodatne obrade (*overhead*) zbog raspodijeljenosti u usporedbi s centraliziranim varijantama. To je očekivano budući da svi indeksiraju sve pretplate u jednom prostornom indeksu i obrađuju svaku objavu u jednoj dretvi. Ovo ukazuje da Apache Spark uvodi vrlo male troškove dodatne obrade u usporedbi s centraliziranim slučajem i opravdava odluku da se predloženi algoritmi implementiraju korištenjem ove platforme. Osim toga vidljivo je da gotovo nema razlike između centraliziranih varijanti što pokazuju slične performanse četiriju evaluiranih prostornih indeksa. Raspodijeljeni algoritmi se ne skaliraju linearno s brojem dostupnih jezgri budući da je u ovom eksperimentu korišten podrazumijevani broj od 4 istodobna paralelna posla u sustavu *Spark Streaming* i podrazumijevani broj od 10.000 objava što je rezultiralo niskom iskoristivosti izvršitelja, kao što je objašnjeno u prethodnom potpoglavlju. Očito je da algoritmi sPS i sPIS koji su implementirani pomoću upita za prostorno spajanje iz knjižnice Apache Sedona imaju samo polovicu propusnosti algoritama sPS i sPIS koji su implementirani korištenjem čistog Sparkovog operatora spajanja. To se očekuje iz razloga objašnjenih u poglavlju 8.1 i u potpunosti opravdava odluku da se algoritme implementira koristeći čisti Sparkov operator spajanja.

8.5 Rasprava rezultata eksperimentalnog vrednovanja predloženih algoritama s primjenske razine

Ovo poglavlje završava raspravom o evaluaciji i primjeni predloženih algoritama. U slučaju algoritama sPS i sPIS pad propusnosti može se objasniti time što se objava i pretplate kandidati za nju mogu nalaziti u nekoliko različitih particija pa je stoga potrebno nekoliko puta izvršiti identičnu, ali zahtjevnu primjenu prostornog predikata te provjeriti odgovara li pretplata kandidat objavi. To nije slučaj s algoritmima RIsPS i RIhPS jer oni ne identificiraju particije kojima objava pripada, već traže odgovarajuće pretplate unutar njihovih particija. Međutim, dok algoritam RIsPS mora izvršiti takvo pretraživanje unutar particija koje su prostorno bliske, algoritam RIhPS mora pretraživati nepovezane particije. Propusnost algoritma RIsPS povećava se mnogo većom intenzitetom nego kod drugih algoritama prilikom povećanja broja paralelnih poslova u sustavu Spark Streaming (engl. Spark Streaming Jobs, SSJ). Razlika u propusnosti između algoritama RIS i RIsPS postaje vrlo mala u slučaju velikog broja istodobnih SSJ-eva. Također, najbolje rezultate propusnosti uz povećanje broja Kafkinih particija ostvaruje algoritam RIS, jer algoritam RIS obrađuje svaku objavu na jednom izvršitelju i stoga ne vrši operaciju miješanja `shuffle`. Algoritam RIS ima najbolji rezultat za 128 Kafkinih particija budući da je tada broj dostupnih jezgri u grozdu najbliži broju Kafkinih particija. Međutim, za druge algoritme bolje je po izvršitelju namijeniti samo jednu procesorsku jezgru za dohvaćanje objava s Kafkine teme, dok su preostale dvije jezgre tada slobodne za obavljanje mnogo složenijih zadataka, kao što je operacija `shuffle`. Propusnost uz povećanje broja izvršitelja tipa `balanced` pokazuje kako se algoritmi horizontalno skaliraju u veće računalne grozdove. Propusnost gotovo svih algoritama raste linearno za sve veći broj izvršitelja i stoga je vidljivo da su oni skalabilni. Ponovno je najbolji algoritam je RIS, ali je samo malo bolji od algoritma RIsPS dok je algoritam RIhPS puno lošiji od algoritm RIsPS, a za veliki broj izvršitelja vrlo je sličan algoritmima sPS i sPIS, koje imaju najlošije rezultate. Prilikom povećanja broja pretplata propusnost prvo počinje polako padati, a zatim nastavlja linearnim padom ali je stopa pada nešto manja od stope rasta pretplata što pokazuje da su varijante skalabilne uz povećanje broja pretplata. Ponovno je najbolji algoritam RIS, a slijede algoritmi RIsPS i RIhPS, dok su algoritmi sPS i sPIS najlošiji i imaju vrlo slične performanse.

Može se zaključiti da algoritam RIS postiže najveću propusnost i najmanje kašnjenje, ali na račun znatne potrošnje memorije. Drugi najbolji algoritam je RIsPS koji postiže više od polovice propusnosti i manje od dvostrukog kašnjenja algoritma RIS te gotovo jednaku propusnost i kašnjenje za veliki broj paralelnih poslova u sustavu Spark Streaming i veliki broj pretplata. Pri tome algoritm RIsPS zahtijeva manje od četvrtine memorije koju zahtijeva algoritam RIS. Algoritam RIhPS je treći najbolji algoritam, ali ga treba koristiti s oprezom jer se za veliki broj izvršitelja i veliki postotak objava točaka ponaša slično kao algoritmi sPS i sPIS. Algoritmi sPS

i sPIS rade gotovo identično što ukazuje da particioniranje pretplata ima mnogo veći utjecaj na propusnost od indeksiranja pretplata. U praksi se preporučuje korištenje algoritama RIS, RIsPS ili sPS zadanim redoslijedom, ali konačnu odluku treba donijeti ovisno o količini dostupne memorije radnih čvorova. Ne preporuča se korištenje algoritama RIhSP i sPIS budući da se RIhSP ponaša loše u nekim situacijama, dok je sPIS složeniji od algoritma sPS, a ponaša se gotovo identično.

Poglavlje 9

Zaključak

U ovom doktorskom radu analizira se geoprostorni sustav objavi/pretplati u kojem objavljene poruke (tj. objave) i pretplate uključuju geoprostorni objekt koji se koristi za izražavanje informacija o lokaciji objave i lokacije od interesa za pretplatu. Glavni problem postojećih geoprostornih sustava objavi/pretplati je njihova centralizirana arhitektura koja se ne može koristiti za obradu u stvarnom vremenu vrlo čestih dolaznih tokova geoprostornih podataka. Kako bi se riješio navedeni problem, napravljen je raspodijeljeni geoprostorni sustav objavi/pretplati u grozdu koji je izgrađen pomoću platforme Apache Spark te su implementirana četiri različita rapodijeljena algoritma za usporedbu objava i pretplata u brojnim varijantama koje koriste tri različita prostorna indeksa i šest različitih metoda prostornog particioniranja pretplata.

Kako bi se pokazala učinkovitost i skalabilnost predloženih algoritama, predstavljeni su rezultati opsežne eksperimentalne studije u kojoj su uspoređene njihove propusnosti, kašnjenje i potrošnja memorije. Usporedba s najsuvremenijim centraliziranim pristupima pokazuje da su dodatni troškovi obrade predloženih raspodijeljenih algoritama gotovo zanemarivi. Eksperimentalna evaluacija pokazuje da je algoritam RIS vremenski najučinkovitiji, ali zahtijeva replikaciju pretplata na svim radnim čvorovima i stoga troši najviše memorije. Algoritam sPS ima najlošije rezultate, ali mu je cjevovod najjednostavniji. Algoritam sPIS je sličan algoritmu sPS, ali je složeniji od algoritma sPS te postiže tek nešto malo vremenski učinkovitiju obradu. Algoritam RIPS ima najkompleksniji cjevovod, ali najbolje balansira između vremenske učinkovitosti i memorijskog zauzeća. Njegova varijanta RIhPS (koja koristi metodu raspršenog particioniranja) troši previše memorije na strani Sparkovog pokretača posla (engl. *driver*) u slučaju velikog broja pretplata te obično ima lošije performance od varijante RIsPS (koja koristi metodu prostornog particioniranja).

U budućnosti se planira ponoviti eksperimentalnu evaluaciju na dodatnim skupovima podataka te također pokušati usporediti performance različitih platformi otvorenog koda za grozd tijekom usporedbe dolaznih tokova geoprostornih podataka sa skupom pohranjenih geoprostornih pretplata. Dodatno, planira se proširiti predloženi raspodijeljeni geoprostorni sustav objavi-

pretplati rješenjem koje će pratiti dolazni tok podataka tijekom vremena te dinamički prilagođavati metodu particioniranja kako bi se izbjegla pojava neuravnoteženosti particija (engl. *partition skew*).

Literatura

[1]Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., Sheth, A. P., "Machine learning for internet of things data analysis: A survey", Digital Communications and Networks, Vol. 4, No. 3, 2018, str. 161–175.

[2]Ouksel, A. M., Jurca, O., Podnar, I., Aberer, K., "Efficient probabilistic subsumption checking for content-based publish/subscribe systems", in ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, 2006, str. 121–140.

[3]Baldoni, R., Querzoni, L., Tarkoma, S., Virgillito, A., Distributed Event Routing in Publish/Subscribe Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, str. 219–244, dostupno na: https://doi.org/10.1007/978-3-540-89707-1_10

[4]Huang, C.-Y., "Geopubsubhub: A geospatial publish/subscribe architecture for the world-wide sensor web", 2014.

[5]Tarkoma, S., Publish/subscribe systems: design and principles. John Wiley & Sons, 2012.

[6]Consortium, O. G. *et al.*, "Opengis simple features specification for sql", OGC Implementation Standard, 1999.

[7]"Jts topology suite javadoc", http://locationtech.github.io/jts/javadoc/index.html, [Online; accessed 20-February-2019]. 2019.

[8]Strobl, C., "Dimensionally extended nine-intersection model (de-9im)", 2008.

[9]Baranovi ć, M., Škrlec, D., "Zdravko galić: Geoprostorne baze podataka", Kartografija i geoinformacije, Vol. 5, No. 6, 2006, str. 149–151.

[10]Shen, J., Zhou, T., Chen, M., "A 27-intersection model for representing detailed topological relations between spatial objects in two-dimensional space", ISPRS International Journal of Geo-Information, Vol. 6, No. 2, 2017, str. 37.

[11]Finkel, R. A., Bentley, J. L., "Quad trees a data structure for retrieval on composite keys", Acta informatica, Vol. 4, No. 1, 1974, str. 1–9.

[12]Guttman, A., "R-trees: A dynamic index structure for spatial searching", in Proceedings of the 1984 ACM SIGMOD international conference on Management of data, 1984, str. 47–57.

[13]Leutenegger, S. T., Lopez, M. A., Edgington, J., "Str: A simple and efficient algorithm for r-tree packing", in Proceedings 13th international conference on data engineering. IEEE, 1997, str. 497–506.

[14]Robinson, J. T., "The kdb-tree: a search structure for large multidimensional dynamic indexes", in Proceedings of the 1981 ACM SIGMOD international conference on Management of data, 1981, str. 10–18.

[15]"UK postcode boundary polygons", https://www.opendoorlogistics.com/downloads/, [Online; accessed 10-March-2019].

[16]Kassab, A., Liang, S., Gao, Y., "Real-time notification and improved situational awareness in fire emergencies using geospatial-based publish/subscribe", International Journal of Applied Earth Observation and Geoinformation, Vol. 12, No. 6, 2010, str. 431 - 438.

[17]Huang, C.-Y., Liang, S., "Ahs model: Efficient topological operators for a sensor web publish/subscribe system", ISPRS International Journal of Geo-Information, Vol. 6, No. 2, 2017, str. 54.

[18]Assilzadeha, H., Zhongb, Z., Kassab, A., Levyc, Y. G. J., "Development of an even-driven and scalable oil spill monitoring and management system", in The 2010 Canadian Geomatics Conference and Symposium of Commission I, ISPRS, Jun. 2010, dostupno na: http://www.isprs.org/proceedings/XXXVIII/part1/02/02_04_Paper_16.pdf

[19]Zhong, Z., Jing, N., Gao, Y., Jha, M., Kassab, A., Assilzadeh, H., "An active real-time system for oil spill detection and information distribution", in 2015 2nd International Conference on Information Science and Control Engineering. IEEE, 2015, str. 110–115.

[20]Nam, B., Sussman, A., "Spatial indexing of distributed multidimensional datasets", in CCGrid, Vol. 2. IEEE, 2005, str. 743–750.

[21]Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S. *et al.*, "Apache hadoop yarn: Yet another resource negotiator", in Proceedings of the 4th annual Symposium on Cloud Computing, 2013, str. 1–16.

[22]Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., "Spark: Cluster computing with working sets", in 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10), 2010.

[23]Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M. J., Shenker, S., Stoica, I., "Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing", in 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), 2012, str. 15–28.

[24]Kreps, J., Narkhede, N., Rao, J. *et al.*, "Kafka: A distributed messaging system for log processing", in Proceedings of the NetDB, Vol. 11, 2011, str. 1–7.

[25]Estrada, R., Apache Kafka 1.0 Cookbook: Over 100 practical recipes on using distributed enterprise messaging to handle real-time data. Packt Publishing Ltd, 2017.

[26]Goodhope, K., Koshy, J., Kreps, J., Narkhede, N., Park, R., Rao, J., Ye, V. Y., "Building linkedin's real-time activity data pipeline.", IEEE Data Eng. Bull., Vol. 35, No. 2, 2012, str. 33–45.

[27]Heiler, G., "Geospark join problem", https://github.com/geoHeil/geoSparkProblemJoinLoosingUserData, [Online; accessed 17-October-2021]. 2017.

[28]Pope, A., "GB postcode area, sector, district, [dataset]. University of Edinburgh", https://doi.org/10.7488/ds/1947, [Online; accessed 20-February-2021]. 2017.

[29]"UK Car Accidents 2005-2015", https://www.kaggle.com/silicon99/dft-accident-data/, [Online; accessed 25-February-2021]. 2021.

[30]Liang, S., Gao, Y. *et al.*, "Real-time notification and improved situational awareness in fire emergencies using geospatial-based publish/subscribe", International Journal of Applied Earth Observation and Geoinformation, Vol. 12, No. 6, 2010, str. 431–438.

[31]Pripuži ć, K., Katušić, D., Marjanović, M., Antonić, A., Livaja, I., "A performance evaluation of spatial indices for geospatial publish/subscribe systems", in 2019 15th International Conference on Telecommunications (ConTEL). IEEE, 2019, str. 1–8.

[32]Edward Chang, Z. M., Pnueli, A., Logic and Algebra of Specifications. Springer-Verlag, 1991, ch. The Safety-Progress Classification.

[33]Tanin, E., Harwood, A., Samet, H., "Using a distributed quadtree index in peer-to-peer networks", VLDB J., Vol. 16, No. 2, 2007, str. 165–178.

[34]Ottenwälder, B., "Mobility-awareness in complex event processing systems", Doktorski rad, University of Stuttgart, 2016.

[35]Du Mouza, C., Litwin, W., Rigaux, P., "Large-scale indexing of spatial data in distributed repositories: the sd-rtree", VLDB J., Vol. 18, No. 4, 2009, str. 933–958.

[36]Starks, F., Goebel, V., Kristiansen, S., Plagemann, T., "Mobile distributed complex event processing—ubi sumus? quo vadimus?", in Mobile Big Data. Springer, 2018, str. 147–180.

[37]Ottenwälder, B., Koldehofe, B., Rothermel, K., Hong, K., Lillethun, D., Ramachandran, U., "Mcep: A mobility-aware complex event processing system", ACM T. Internet Techn., Vol. 14, No. 1, 2014, str. 1–24.

[38]Gong, Y., Kuang, H., Cai, X., Hu, H., Song, W., Lü, J., "Parallelized mobility-aware complex event processing", in ICWS. IEEE, 2017, str. 898–901.

[39]Gounaris, A., Kougka, G., Tous, R., Montes, C. T., Torres, J., "Dynamic configuration of partitioning in spark applications", IEEE T. Parall. Distr., Vol. 28, No. 7, 2017, str. 1891–1904.

[40]Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N., Theodoridis, Y., R-Trees: Theory and Applications. Springer, 2005.

[41]Yu, J., "Move to jts 1.18, minimize the dependency on jtsplus", https://github.com/apache/incubator-sedona/pull/488, [Online; accessed 25-October-2021]. 2020.

[42]D'Roza, T., Bilchev, G., "An overview of location-based services", BT Technol. J., Vol. 21, No. 1, 2003, str. 20–27.

[43]Marjanovi ć, M., Skorin-Kapov, L., Pripužić, K., Antonić, A., Žarko, I. P., "Energy-aware and quality-driven sensor management for green mobile crowd sensing", Journal Netw. Comput. Appl., Vol. 59, 2016, str. 95–108.

[44]He, B., Tong, X.-h., Chen, P., "A multi-agent based architecture for processing continuous geospatial queries in transportation networks", J. Converg. Inf. Technol., Vol. 7, No. 20, 2012.

[45]Ardagna, C. A., Cremonini, M., Damiani, E., Di Vimercati, S. D. C., Samarati, P., "Location privacy protection through obfuscation-based techniques", in DBSec. Springer, 2007, str. 47–60.

[46]Antoni ć, A., Marjanović, M., Pripužić, K., Žarko, I. P., "A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things", Future Gener. Comp. Sy., Vol. 56, 2016, str. 607–622.

[47]Motlagh, N. H., Lagerspetz, E., Nurmi, P., Li, X., Varjonen, S., Mineraud, J., Siekkinen, M., Rebeiro-Hargrave, A., Hussein, T., Petaja, T. *et al.*, "Toward massive scale air quality monitoring", IEEE Commun. Mag., Vol. 58, No. 2, 2020, str. 54–59.

[48]Livaja, I., Skvorc, D., Pripuzic, K., "Geospatial publish/subscribe systems for the internet of things", in SoftCOM. IEEE, 2017, str. 1–8.

[49]Takasu, A. *et al.*, "An efficient distributed index for geospatial databases", in DEXA. Springer, 2015, str. 28–42.

[50]Wang, L., Chen, B., Liu, Y., "Distributed storage and index of vector spatial data based on hbase", in Geoinformatics. IEEE, 2013, str. 1–5.

[51]Wan, S., Zhao, Y., Wang, T., Gu, Z., Abbasi, Q. H., Choo, K.-K. R., "Multi-dimensional data indexing and range query processing via voronoi diagram for internet of things", Future Gener. Comp. Sy., Vol. 91, 2019, str. 382–391.

[52]Zhang, F., Zheng, Y., Xu, D., Du, Z., Wang, Y., Liu, R., Ye, X., "Real-time spatial queries for moving objects using storm topology", ISPRS Int. J. Geo-Inf., Vol. 5, No. 10, 2016, str. 178.

[53]Walton, C. B., Dale, A. G., Jenevein, R. M., "A taxonomy and performance model of data skew effects in parallel joins.", in VLDB, Vol. 91, 1991, str. 537–548.

[54]Rowstron, A. I. T., Kermarrec, A.-M., Castro, M., Druschel, P., "Scribe: The design of a large-scale event notification infrastructure", in NGC, 2001, str. 30-43.

[55]Carzaniga, A., Rutherford, M. J., Wolf, A. L., "A routing scheme for content-based networking", in INFOCOM, 2004, str. 918-928.

[56]Mühl, G., "Large-Scale Content-Based publish/subscribe systems", Doktorski rad, Darmstadt University of Technology, 2002.

[57]Carzaniga, A., "Architectures for an event notification service scalable to wide-area networks", Doktorski rad, Politecnico di Milano, 1998.

[58]Baldoni, R., Querzoni, L., Tarkoma, S., Virgillito, A., "Distributed event routing in publish/subscribe systems", in MiNEMA. Springer, 2009, str. 219–244.

[59]Wang, A., Xian, X., Tsung, F., Liu, K., "A spatial-adaptive sampling procedure for online monitoring of big data streams", J. Qual. Technol., Vol. 50, No. 4, 2018, str. 329–343.

[60]Amagata, D., Hara, T., "A general framework for maxrs and maxcrs monitoring in spatial data streams", ACM Trans. Spat. Algorithms Syst., Vol. 3, No. 1, 2017, str. 1–34.

[61]Kraft, R., Birk, F., Reichert, M., Deshpande, A., Schlee, W., Langguth, B., Baumeister, H., Probst, T., Spiliopoulou, M., Pryss, R., "Efficient processing of geospatial mhealth data using a scalable crowdsensing platform", Sensors, Vol. 20, No. 12, 2020, str. 3456.

[62]Pandey, V., Kipf, A., Neumann, T., Kemper, A., "How good are modern spatial analytics systems?", Proc. VLDB Endow., Vol. 11, No. 11, 2018, str. 1661–1673.

[63]Tang, M., Yu, Y., Malluhi, Q. M., Ouzzani, M., Aref, W. G., "Locationspark: A distributed in-memory data management system for big spatial data", Proc. VLDB Endow., Vol. 9, No. 13, 2016, str. 1565–1568.

[64]Salloum, S., Dautov, R., Chen, X., Peng, P. X., Huang, J. Z., "Big data analytics on apache spark", Int. J. Data Sci. Anal., Vol. 1, No. 3, 2016, str. 145–164.

[65]You, S., Zhang, J., Gruenwald, L., "Large-scale spatial join query processing in cloud", in ICDE. IEEE, 2015, str. 34–41.

[66]Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., Saltz, J., "Hadoop-gis: A high performance spatial data warehousing system over mapreduce", in VLDB. NIH Public Access, 2013, str. 1009-1020.

[67]Yu, J., Zhang, Z., Sarwat, M., "Spatial data management in apache spark: The geospark perspective and beyond", GeoInformatica, Vol. 23, No. 1, 2019, str. 37–78.

[68]Yu, J., Wu, J., Sarwat, M., "Geospark: A cluster computing framework for processing large-scale spatial data", in SIGSPATIAL, 2015, str. 1–4.

[69]Eldawy, A., "Spatialhadoop: towards flexible and scalable spatial processing using mapreduce", in Proceedings of the 2014 SIGMOD PhD symposium, 2014, str. 46–50.

[70]Landset, S., Khoshgoftaar, T. M., Richter, A. N., Hasanin, T., "A survey of open source tools for machine learning with big data in the hadoop ecosystem", J. Big Data, Vol. 2, No. 1, 2015, str. 1–36.

[71]Laney, D., "3d data management: Controlling data volume, velocity and variety", Meta Group, Tech. Rep., 2001.

[72]Gandomi, A., Haider, M., "Beyond the hype: Big data concepts, methods, and analytics", Int. J. of Inf. Manage., Vol. 35, No. 2, 2015, str. 137-144.

[73]Lee, J.-G., Kang, M., "Geospatial big data: Challenges and opportunities", Big Data Res., Vol. 2, No. 2, 2015, str. 74-81, visions on Big Data.

[74]Rieke, M., Bigagli, L., Herle, S., Jirka, S., Kotsev, A., Liebig, T., Malewski, C., Paschke, T., Stasch, C., "Geospatial iot—the need for event-driven architectures in contemporary spatial data infrastructures", ISPRS Int. J. Geo-Inf., Vol. 7, No. 10, 2018, str. 385.

[75]Bigagli, L., Rieke, M., "The new ogc publish/subscribe standard-applications in the sensor web and the aviation domain", Open Geospat. Data Softw. Stand., Vol. 2, 2017, str. 1–10.

[76]Okabe, A., Boots, B., Sugihara, K., Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. USA: John Wiley & Sons, Inc., 1992.

[77]Kamel, I., Faloutsos, C., "On packing r-trees", in CIKM, 1993, str. 490–499.

[78]Kassab, A. S., "Geospatial-based publish-subscribe: improving real-time notification and situational awareness in fire emergency", Master's thesis, University of Calgary, 2009.

[79]Huang, C.-Y., "Geopubsubhub: A geospatial publish/subscribe architecture for the world-wide sensor web", Doktorski rad, University of Calgary, 2014.

[80]Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N., Theodoridis, Y., R-trees: Theory and Applications. Springer Science & Business Media, 2010.

[81]Samet, H., "The quadtree and related hierarchical data structures", ACM Comput. Surv., Vol. 16, No. 2, 1984, str. 187–260.

[82]Burcea, I., Jacobsen, H.-A., "L-topss–push-oriented location-based services", in TES. Springer, 2003, str. 131–142.

[83]Chen, X., Chen, Y., Rao, F., "An efficient spatial publish/subscribe system for intelligent location-based services", in DEBS. ACM, 2003, str. 1–6.

[84]Meier, R., Cahill, V., "Steam: Event-based middleware for wireless ad hoc network", in ICDCSW, 2002, str. 639-644.

[85]Fiege, L., Gärtner, F. C., Kasten, O., Zeidler, A., "Supporting mobility in content-based publish/subscribe middleware", Lect. Notes Comput. Sci., Vol. 2672, 2003, str. 103–122.

[86]Sivaharan, T., Blair, G. S., Coulson, G., "Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing", Lect. Notes Comput. Sci., Vol. 3760, 2005, str. 732–749.

[87]Cugola, G., de Cote, J. E. M., "On introducing location awareness in publish-subscribe middleware", in ICDCSW, 2005, str. 377-382.

[88]Krishnamurthy, B., Rosenblum, D. S., “Yeast: A general purpose event-action system”, IEEE T. Software. Eng., Vol. 21, No. 10, 1995, str. 845-857.

[89]Segall, B., Arnold, D., Boot, J., Henderson, M., Phelps, T., “Content based routing with elvin4”, in AUUG, 2000.

[90]Segall, W., Arnold, D., “Elvin has left the building: A publish/subscribe notification service with quenching”, in AUUG, 1997.

[91]Mansouri-Samani, M., Sloman, M., “Gem: a generalized event monitoring language for distributed systems”, Distrib. Syst. Eng., Vol. 4, 1997, str. 9.

[92]Wray, M., Hawkes, R., “Distributed virtual environments and vrml: an event-based architecture”, Comput. Netw. ISDN Syst., Vol. 30, 1998, str. 43–51.

[93]Notification Service Specification - Version 1.1, http://www.omg.org/technology/documents/formal/notification_service.htm, Object Management Group, October 2004.

[94]Ramani, S., Trivedi, K. S., Dasarathy, B., “Reliable messaging using the corba notification service”, in DOA, 2001.

[95]Gruber, R. E., Krishnamurthy, B., Panagos, E., “The architecture of the ready event notification service”, in ICDCS. IEEE, 1999, str. 108–113.

[96]Banavar, G., Chandra, T., andJay Nagarajarao, B. M., Strom, R. E., Sturman, D. C., “An efficient multicast protocol for content-based publish-subscribe systems”, in ICDCS, 1999.

[97]Aguilera, M. K., Strom, R. E., Sturman, D. C., Astley, M., Chandra, T. D., “Matching events in a content-based subscription system”, in PODC, 1999.

[98]Carzaniga, A., Rosenblum, D. S., Wolf, A. L., “Achieving scalability and expressiveness in an internet-scale event notification service”, in PODC, 2000.

[99]Pereira, J., Fabret, F., Llirbat, F., Preotiuc-Pietro, R., Ross, K. A., Shasha, D., “Publish/subscribe on the web at extreme speed”, in VLDB, 2000.

[100]Cugola, G., Nitto, E. D., Fuggetta, A., “The jedi event-based infrastructure and its application to the development of the opss wfms”, IEEE T. Software. Eng., Vol. 27, 2001, str. 827-850.

[101]Cao, F., Singh, J. P., “Efficient event routing in content-based publish/subscribe service network”, in INFOCOM, 2004.

[102]Gupta, A., Sahin, O. D., Agrawal, D., Abbadi, A. E., “Meghdoot: content-based publish/subscribe over p2p networks”, Lect. Notes Comput. Sci., Vol. 3231, 2004, str. 254–273.

[103]Mühl, G., Ulbrich, A., Herrmann, K., Weis, T., “Disseminating information to mobile clients using publish-subscribe”, IEEE Internet. Comput., Vol. 8, 2004, str. 46–53.

[104]Fabret, F., Jacobsen, H. A., Llirbat, F., Pereira, J., Ross, K. A., Shasha, D., “Filtering algorithms and implementation for very fast publish/subscribe systems”, in SIGMOD, 2001.

[105]Hapner, M., Burridge, R., Sharma, R., “Java message service specification - version 1.1”, Sun Microsystems, Tech. Rep., 2002.

[106]Lamport, L., “Proving the correctness of multiprocess programs.”, IEEE Trans. Software Eng., Vol. 3, 1977, str. 125-143.

[107]Sistla, A. P., “On characterization of safety and liveness properties in temporal logic”, in PODC, 1985.

[108]Guttman, A., “R-trees: A dynamic index structure for spatial searching”, in MOD. ACM, 1984, str. 47–57.

[109]Finkel, R. A., Bentley, J. L., “Quad trees: A data structure for retrieval on composite keys”, Acta Inform., Vol. 4, No. 1, Mar. 1974, str. 1–9.

[110]Strobl, C., “Dimensionally extended nine-intersection model (de-9im)”, in Encyclopedia of GIS. Springer, 2017, str. 470–476.

[111]Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., Polyzos, G. C., “A survey of information-centric networking research”, IEEE Commun. Surv. Tutor, Vol. 16, No. 2, 2013, str. 1024–1049.

[112]Trossen, D., Parisis, G., “Designing and realizing an information-centric internet”, IEEE Commun. Mag, Vol. 50, No. 7, 2012, str. 60–67.

[113]C Sofia, R., M Mendes, P., “An overview on push-based communication models for information-centric networking”, Future Internet, Vol. 11, No. 3, 2019, str. 74.

[114]Bayer, R., “Symmetric binary b-trees: Data structure and maintenance algorithms”, Acta informatica, Vol. 1, No. 4, 1972, str. 290–306.

[115]Meier, R., Cahill, V., “Steam: Event-based middleware for wireless ad hoc networks”, in Proceedings 22nd International Conference on Distributed Computing Systems Workshops. IEEE, 2002, str. 639–644.

[116]Fiege, L., Gartner, F. C., Kasten, O., Zeidler, A., “Supporting mobility in content-based publish/subscribe middleware”, in ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, 2003, str. 103–122.

[117]Sivaharan, T., Blair, G., Coulson, G., “Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing”, in OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, 2005, str. 732–749.

[118]Cugola, G., de Cote, J. E. M., “On introducing location awareness in publish-subscribe middleware”, in 25th IEEE International Conference on Distributed Computing Systems Workshops. IEEE, 2005, str. 377–382.

[119]Kassab, A., Kassab, A. S., “Geospatial-based publish/subscribe: Improving real-time notification and situational awareness in fire emergency, msc”, 2009.

[120]Assilzadeh, H., Zhong, Z., Kassab, A., Gao, Y., Levy, J., “Development of an even-driven and scalable oil spill monitoring and management system”, in Canadian Geomatics Conference and ISPRS Technical Commission I Symposium. Citeseer, 2010.

[121]Carzaniga, A., “Architectures for an event notification service scalable to wide-area networks”, Doktorski rad, PhD thesis, Politecnico di Milano, Milano, Italy, 1998.

[122]Mühl, G., “Large-scale content-based publish-subscribe systems”, Doktorski rad, Technische Universität, 2002.

[123]Carzaniga, A., Rutherford, M. J., Wolf, A. L., “A routing scheme for content-based networking”, in IEEE INFOCOM 2004, Vol. 2. IEEE, 2004, str. 918–928.

[124]Rowstron, A., Kermarrec, A.-M., Castro, M., Druschel, P., “Scribe: The design of a large-scale event notification infrastructure”, in International workshop on networked group communication. Springer, 2001, str. 30–43.

[125]Lee, J., Kang, M., “Geospatial big data: challenges and opportunities. big data res 2: 74–81”, 2015.

[126]Gandomi, A., Haider, M., “Beyond the hype: Big data concepts, methods, and analytics”, International journal of information management, Vol. 35, No. 2, 2015, str. 137–144.

Popis slika

Popis tablica

Popis algoritama

Životopis

Ivan Livaja je viši predavač na Veleučilišta u Šibeniku. Nakon završene tehničke škole u Šibeniku, 1998. godine upisuje studij na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. U srpnju 2003. godine diplomira i stječe zvanje diplomirani inženjer elektrotehnike, smjer telekomunikacije i informatika. Magistrirao je u polju radiokomunikacija na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu 2008. godine. Od 2016. student je doktorskog studija iz područja tehničkih znanosti, znanstvenog polja računarstva na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Od prosinca 2003. do 2012. zaposlen je u tvrtki Hrvatski Telekom d.d. Od 2015 radi na Veleučilištu u Šibeniku kao vanjski suradnik u nastavi. Od 2015. zaposlen je na Veleučilištu u Šibeniku. Predavač je Oracle akademije na Veleučilištu u Šibeniku. Stručni interesi uključuju razvoj baza podataka, filtriranje informacija, obrada velikih količina podataka, raspodijeljeni informacijski sustavi, te Internet stvari. Objavio više znanstvenih i stručnih radova na konferencijama s međunarodnom recenzijom. Član je strukovne udruge IEEE.

Popis objavljenih djela

1.I. Livaja, F. Urem, A. Grubiši ć, T. Radić Lakoš, I. Žaja, (2015), E-učenje podržano rješenjima u oblaku, Proceedings of MIPRO 2014 International Convention on CTI Conference, 2015.

2.I. Livaja, A. Granc, I. Žaja, F. Urem, (2014), Primjena cloud rješenja na Veleu čilištu u Šibeniku, Proceedings of WICT/I: Workshop on Information and Communication Technologies, SoftCom, Split, 2014.

3.I. Bumbak, I. Livaja, F. Urem, (2015), SEO na primjeru web sjedišta "Apartmani Sandra", Zbornik radova Veleučilišta u Šibeniku, No.3-4/2015, ISBN 1846-6699, prosinac 2015.

4.K. Čoko, I. Bumbak, I. Livaja, (2015), Primjena NFC tehnologije u turističkom poslovanju, Zbornik radova 2. međunarodne znanstveno-stručne konferencije Izazovi današnjice - turizam i lokalni razvoj, Šibenik, 2015.

5.N. Poljak, M. Ševo, I. Livaja, (2015), Upotreba informati čkih tehnologija u turizmu, Zbornik radova 2. međunarodne znanstveno-stručne konferencije Izazovi današnjice - turizam i lokalni razvoj, Šibenik, 2015.

6.N. Poljak, D. Zlatovi ć, I. Livaja, (2015), Security and privacy in an IT context: keeping businesses and clients safe, Zbornik radova 2. međunarodne znanstveno-stručne konferencije Izazovi današnjice - turizam i lokalni razvoj, Šibenik, 2015.

7.I. Smilj čić, I. Livaja, J. Acalin, (2017), ICT U OBRAZOVANJU, Zbornik radova Veleučilišta u Šibeniku, No.3-4/2017, ISSN 1846-6699, rujan 2017.

8.I. Bumbak, I. Livaja, (2017), Internet of Things u nauti čkom i obalnom turizmu , , Zbornik radova 3. međunarodne znanstveno-stručne konferencije Izazovi današnjice - turizam i lokalni razvoj, Šibenik, 2017.

9.I. Bumbak, I. Livaja, (2017), Besplatni Google Web alati na primjeru studentskog portala „SKVUŠ" , Zbornik radova Veleučilišta u Šibeniku, No.3-4/2017, ISSN 1846-6699 , rujan 2017.

10.I. Livaja, J. Acalin, (2017), XMPP , Zbornik radova Veleu čilišta u Šibeniku, No.3-4/2017, ISSN 1846-6699 , studeni 2017.

11.I. Livaja, T. Kova čević, M. Fijolić, (2007), Internet Protocol over DWHM- Proceedings of MIPRO 2007 International Convention on CTI Conference, pages: 162-166, 2007

12.I. Livaja, I. Žaja, M. Šišul, (2008), Modularna T-Com rješenja za hotele - Proceedings of MIPRO2008 International Convention on CTI Conference, pages: 175-179, 2008

13.F. Urem, K. Fertalj, I. Livaja (2014) , Modeling of software failure cost in ERP systems, Proceedingsof MIPRO 2014 International Convention on CTI Conference, 2014.

14.F. Urem, K. Fertalj, I. Livaja, B. Šupe (2015) , Anonymity ensurance in creation of satisfaction surveys – experience of Polytechnic of Šibenik, Proceedingsof MIPRO 2015

International Convention on CTI Conference, 2015.

15.I. Livaja, D. Škvorc, K. Pripuži ć, (2017), Geospatial Publish/Subscribe Systems for the Internet of Things , Proceedings of WICT/I: Workshop on Information and Communication Technologies, SoftCom, Split, 2017.

16.N. Poljak, M. Ševo, I. Livaja (2016), Security and privacy in an IT context – a low-cost WIDS employed against MITM attacks (concept), Proceedings of MIPRO 2016 International Convention on CTI Conference, 2016.

17.M. Klari ć, I. Kuzle, I. Livaja (2016), Adaptive and modular urban smart infrastructure, Proceedings of MIPRO 2016 International Convention on CTI Conference, 2016.

18.F. Urem, K. Fertalj, I. Livaja, (2011), The impact of upgrades on ERP system reliability, World Academy of Science, Engineering and Technology, Issue 59.Venice, Italy: WASET, pages:1237-1241, 2011.

19.Vranji ć, Petra; Livaja, Ivan; Klarin, Zvonimir, Računalna kreativnost // Proceedings of the 5th International Scientific and Professional Conference "The Challenges of Today" / Filiposki, Oliver ; Metodijeski, Dejan ; Zlatovic, Dragan (ur.).Šibenik: Veleučilište u Šibeniku, 2021. str. 537-547 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), stručni)

20.Urem, F.; Jurekovic, D.; Livaja, I., How to teach databases in a student-centered environment - Oracle Academy program experiences // 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO),Opatija, Hrvatska: IEEE, 2021. str. 128-135 doi:10.23919/mipro52101.2021.9596945 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

21.Zlatovi ć, Dragan; Livaja, Ivan; Galić, Ante: Zaštita znaka udruge prema Zakonu o udrugama i posebnim propisima // (Proceedings Polytechnic of Sibenik 4th International Conference The Challenges of Today 10th – 12th October 2019)/ Šibenik: Veleučilište u Šibeniku, 2019. str. 503-511 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

22.Livaja, Ivan; Klarin, Zvonimir: Utjecaj 5G mreže na Internet stvari // Zbornik radova Veleučilišta u Šibeniku, 1-2/2020 (2020), 1-2/2020; 155-169 (domaća recenzija, članak, stručni)

23.Pripuži ć, Krešimir; Katušić, Damjan; Marjanović, Martina; Antonić, Aleksandar; Livaja, Ivan, A Performance Evaluation of Spatial Indices for Geospatial Publish/Subscribe Systems // Proceedings of the 15th International Conference on Telecommunications (ConTEL 2019),Graz: IEEE, 2019. str. 1-8 doi:10.1109/ConTEL.2019.8848503 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Rad u časopisima

1.Livaja, Ivan; Pripuži ć, Krešimir; Sovilj, Siniša; Vuković, Marin: A distributed geospatial publish/subscribe system on Apache Spark // Future Generation Computer Systems, 132 (2022), 282-298 doi:10.1016/j.future.2022.02.013 (međunarodna recenzija, članak, znanstveni)

Biography

Ivan Livaja is a senior lecturer at the Polytechnic of Šibenik. After completing the technical school in Šibenik, in 1998 he enrolled in the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (UNIZG-FER). In 2003 and 2008 he received the diploma degree in electrical engineering, with a major in telecommunications and informatics, and the M.Sc degree in the field of radiocommunications from the UNIZG-FER, respectively. Since 2016 he has been a PhD student at UNIZG-FER. From 2003 to 2012 he was employed by the Croatian Telecom. From 2013 to 2015, he worked at the Polytechnic of Šibenik as an external associate. Since 2015 he has been employed at the Polytechnic of Šibenik, first as a lecturer and then as a senior lecturer. He is also a lecturer at the Oracle Academy at the Polytechnic of Šibenik. His main research interests are databases, information filtering systems, big data processing, distributed information systems and internet of things. He has published many scientific and professional papers at international conferences. He is a member of a professional association of IEEE.