

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

Ana Aponel, DGTfm-22

**Didžiųjų duomenų apdorojimo kursinis darbas**

Vilnius, 2023

TURINYS

[1. Įvadas 5](#_Toc135414377)

[1.1. Sprendžiamų uždavinių pristatymas 5](#_Toc135414378)

[1.2. Naudojamų priemonių pristatymas 5](#_Toc135414379)

[1.2.1. „MapReduce“ 5](#_Toc135414380)

[1.2.2. „Hadoop“ platforma 5](#_Toc135414381)

[1.2.3. „Oracle VM VirtualBox“ 6](#_Toc135414382)

[1.2.4. „WinSCP“ 6](#_Toc135414383)

[1.2.5. „Ambari“ 6](#_Toc135414384)

[1.2.6. „PuTTY“ 6](#_Toc135414385)

[1.2.7. „Apache Spark“ 6](#_Toc135414386)

[1.2.7. „Google Colab“ 7](#_Toc135414387)

[2. Pirmojo uždavinio sprendimas 8](#_Toc135414388)

[2.1. „MapReduce“ programavimo modelis 8](#_Toc135414389)

[2.2. „Combiner“ tipo užduoties įtraukimas 9](#_Toc135414390)

[2.3. „Hadoop MapReduce“ naudojimas 9](#_Toc135414391)

[3. Antrojo uždavinio sprendimas 12](#_Toc135414392)

[3.1. „DAG“ programavimo modelis 12](#_Toc135414393)

[3.1.1. Pirminis „RDD“ 12](#_Toc135414394)

[3.1.2. Transformacijos 12](#_Toc135414395)

[3.1.3. Veiksmai 13](#_Toc135414396)

[3.1.4. Duomenų priklausomybės 13](#_Toc135414397)

[3.1.5. Acikliškumas 13](#_Toc135414398)

[3.1.6. Topologinė tvarka 13](#_Toc135414399)

[3.2. „Spark“ transformacijos ir veiksmai 14](#_Toc135414400)

[3.2.1. „RDD“ rinkinių transformacijos 14](#_Toc135414401)

[3.2.2. „RDD“ rinkinių veiksmai 14](#_Toc135414402)

[3.2.3. Duomenų siuntimas dėl transformacijų 14](#_Toc135414403)

[3.2.4. Naudotos „Spark“ transformacijos ir veiksmai 14](#_Toc135414404)

[4. Trečiojo uždavinio sprendimas 18](#_Toc135414405)

[4.1. Duomenų aproksimacijos vizualizacija 18](#_Toc135414406)

[4.2. Naudotos „Python“ bibliotekos ir komandos 18](#_Toc135414407)

[4.2.1. „Matplotlib“ 18](#_Toc135414408)

[4.2.2. „Pandas“ 19](#_Toc135414409)

[4.2.5. „pyspark.ml“ 20](#_Toc135414410)

[4.2.6. „pyspark.sql“ 22](#_Toc135414411)

[4.3. Duomenų pertvarkymai 23](#_Toc135414412)

[4.4. Duomenų siuntimas 24](#_Toc135414413)

PAVEIKSLŲ SĄRAŠAS

[1 pav. „Map“ fazė 7](#_Toc135223486)

[2 pav. duomenys gauti po „Map“ fazės 7](#_Toc135223487)

[3 pav. „Reduce“ fazė 8](#_Toc135223488)

[4 pav Duomenys po „Reduce“ fazės 8](#_Toc135223489)

[5 pav. Vaizdas paleidus „Hortonworks HDP Sandbox“ naudojant „VirtualBox“ 9](#_Toc135223490)

[6 pav. „Sandbox“ paleidimas 9](#_Toc135223491)

[7 pav. „Files View“ mygtukas 9](#_Toc135223492)

[8 pav. „PuTTY“ terminalo vaizdas, kuomet „MapReduce“ modelis sėkmingai įvykdytas 10](#_Toc135223493)

[9 pav. Sėkmingai paleistas „MapReduce“ modelis per „Hadoop“ sistemą 10](#_Toc135223494)

[10 pav. Pirminis „RDD“ 11](#_Toc135223495)

[11 pav. „RDD“ transformacijos 11](#_Toc135223496)

[12 pav. „RDD“ veiksmai 12](#_Toc135223497)

[13 pav. Duomenų priklausomybės 12](#_Toc135223498)

[14 pav. „flatMap()“ panaudojimas antrajame laboratoriniame darbe 14](#_Toc135223499)

[15 pav. Pirminis duomenų rinkinys 14](#_Toc135223500)

[16 pav. Duomenys po „flatMap()“ panaudojimos 14](#_Toc135223501)

[17 pav. „map()“ ir „filter()“ panaudojimas antrajame laboratoriniame darbe 14](#_Toc135223502)

[18 pav. Duomenys po „map()“ panaudojimo 15](#_Toc135223503)

[19 pav. Duomenys po „filter()“ panaudojimo kai yra išfiltruojamos „None“ reikšmės 15](#_Toc135223504)

[20 pav. „goupByKey()“, „mapValues()“, „filter()“ panaudojimas antrajame laboratoriniame darbe 15](#_Toc135223505)

[21 pav. Duomenys po „groupByKey“ panaudojimo 16](#_Toc135223506)

[22 pav. Duomenys po „mapValues()“ panaudojimo 16](#_Toc135223507)

[23 pav. „collect()“ panaudojimas antrajame laboratoriniame darbe 16](#_Toc135223508)

[24 pav. Duomenys po „collect()“ panaudojimo 16](#_Toc135223509)

# Įvadas

## Sprendžiamų uždavinių pristatymas

Šiame kursiniame darbe bus sprendžiami trys uždaviniai. Visiems jiems yra duotas tas pats duomenų rinkinys, kuriame pateikti dalinia struktūrizuoti duomenys apie siuntų išvežiotojų sustojimus, kurie gali būti apibūdinami sekančiai parametrais: "marsrutas", "sustojimo data", "sandelio id", "Firma", "Marsruto tipas", "Masinos tipas", "sustojimo tipas", "sustojimo savaites diena", "laikas", "Sustojimo numeris", "siuntu skaicius", "svoris", "svorio grupe", "geografine zona", "pasto kodas", "Aptarnavimo grupe", "tipas", "Laikas iki sustojimo", "Laikas po sustojimo", "Uzkrovimo tipas", "Ar reikalingos paletes", "Laukia", "Sustojimo klientu skaicius", "sustojimo klientu sarasas", "kaina procentas", "kaina vienetais".

* Pirmame uždavinyje reikėjo naudojant „MapReduce“ programavimo modelį ir „Hadoop“ platformą suskaičiuoti mažiausią, didžiausią ir vidutinį (aritmetinį vidurkį) siuntų svorį (laukas „svoris“) skirtingose svorio grupėse („svorio grupė“).
* Antrame uždavinyje reikėjo naudojant „Apache Spark“ rasti maršrutus (siuntas su tuo pačiu lauku „marsrutas“), kurie aplanko daugiau nei vieną geografinę zoną (parametras „geografine zona“). Rasti atvejus, kai tai daroma tą pačią dieną.
* Trečiame uždavinyje prisidėjo papildomi duomenys apie maršrutus skirtingomis savates dienomis (t.y. duomenys sugrupuoti pagal šių parametrų porą: „marsrutas“, „sustojimo data“) su sekančiais parametrais: „marsrutas“, „sustojimo data“, „M“, „BendrasAtstumas“, „BendrasSvoris“, „BendrasLaikas“, „BendraKaina“. Šiam uždaviniui reikėjo naudojant „Apache Spark“ ir „DataFrame“ duomenų struktūrą, ištirti tiesinę priklausomybę parametro „BendraKaina“ nuo parametro „siuntų skaičius“ (agreguojant pagal maršrutą ir datą, pritaikius sumos operaciją), kai nagrinėjami duomenys tik su viena ta pačia reikšme „Masinos tipas“. Analizės metu reikėjo pritaikyti tiesinę regresiją.

## Naudojamų priemonių pristatymas

Šiame skyriuje bus aprašytos naudojamos priemonės: programinė įranga ir didžiųjų duomenų sistemos.

### „MapReduce“

„MapReduce“ programavimo modelis skirtas didelės apimties skaičiavimams realizuoti. Pirmą kartą pasauliui jis buvo pristatytas „Google“ įmonės, tačiau šiuo metu jį naudoja daug organizacijų. Pagrindinė „MapReduce“ idėja yra padalinti didelį duomenų rinkinį į mažesnius gabalus ir atskirai apdoroti kiekvieną duomenų rinkinį skirtinguose klasterių mazguose. Apdorojimas susideda iš dviejų fazių: „Map“ ir „Reduce“. „Map“ fazėje duomenys transformuojami į raktų-reikšmių poras, o „Reduce“ fazėje – duomenys sujungiami pagal raktą. Šis modelis buvo naudojamas pirmajame laboratoriniame darbe. Jis buvo pritaikytas išspręsti duotai užduočiai.

### „Hadoop“ platforma

„Hadoop“ yra galinga atvirojo kodo sistema, leidžianti paskirstyti ir apdoroti didelius duomenų rinkinius. Jis sukurtas dirbti su pagrindinės aparatinės įrangos grupėmis. „Hadoop“ sukurta remiantis dviem pagrindiniais komponentais: „Hadoop Distributed File System“ (HDFS) ir „MapReduce“. „HDFS“ yra paskirstyta failų sistema, kuri gali saugoti ir valdyti didelius failus keliuose grupės mazguose. „Hadoop“ galimybė keisti mastelį horizontaliai reiškia, kad ji gali lengvai prisitaikyti prie augančių duomenų rinkinių ir darbo krūvių. Ši sistema buvo naudojama pirmajame laboratoriniame darbe duomenis saugoti ir tvarkyti.

### „Oracle VM VirtualBox“

„Oracle VM VirtualBox“ yra galinga virtualizacijos programinė įranga, leidžianti vartotojams vienu metu paleisti kelias operacines sistemas viename kompiuteryje. Šis įrankis gali būti naudingas sukuriant „Hadoop“ klasterius. Naudojant „VirtualBox“ vartotojas gali sukurti „Hadoop“ klasterį virtualioje aplinkoje, nereikalaujant fizinės įrangos. Paleidus kelias virtualias mašinas vartotojas gali eksperimentuoti su skirtingomis „Hadoop“ konfigūracijomis, pvz., keisti mazgų skaičių, atminties ar saugyklos kiekį arba naudojamą „Hadoop“ versiją.

### „WinSCP“

„WinSCP“ yra populiarus atvirojo kodo „SFTP“ (saugaus failų perdavimo protokolo) klientas, skirtas „Windows“ operacinei sistemai, leidžiantis vartotojams saugiai perkelti failus tarp vietinių ir nuotolinių sistemų. Nors „WinSCP“ nėra tiesiogiai susijęs su „Hadoop“, jis gali būti naudingas įrankis valdyti „Hadoop“ grupes. Naudodami „WinSCP“ vartotojai gali lengvai perkelti failus į „Hadoop HDFS“ failų sistemą ir iš jos, o tai ypač naudinga įkeliant didelius duomenų rinkinius, skirtus apdoroti arba atsisiųsti duomenų apdorojimo užduočių rezultatus. Be to, patogi „WinSCP“ sąsaja ir galingos failų valdymo funkcijos gali padėti lengviau naršyti ir tvarkyti „Hadoop“ grupes. Šis klientas buvo naudojamas pirmajame laboratoriniame darbe failų įkėlimui į „Hadoop“ klasterį iš lokalios aplinkos.

### „Ambari“

„Apache Ambari“ yra „Hadoop“ grupių atvirojo kodo valdymo platforma, supaprastinanti didelio masto paskirstytų sistemų administravimą ir stebėjimą. Ši platforma suteikia žiniatinklio grafinę vartotojo sąsają (angl. „GUI“), leidžiančią vartotojams konfigūruoti ir valdyti „Hadoop“ grupes iš vienos vietos. „Ambari“ automatizuoja daugelį įprastų administracinių užduočių, tokių kaip klasterio sąranka, programinės įrangos diegimas, konfigūracija ir priežiūra, taip sumažinant laiką ir pastangas, reikalingas „Hadoop“ klasteriui valdyti. Tai taip pat apima stebėjimo ir perspėjimo funkcijas, kurios realiuoju laiku suteikia klasterio būsenos, našumo ir panaudojimo matomumą. „Ambari“ yra labai išplečiamas ir palaiko pasirinktinių įrankių ir programų integravimą, todėl tai yra universali platforma, skirta valdyti ir stebėti bet kokio dydžio „Hadoop“ grupes. Ši platforma buvo naudojama pirmajame laboratoriniame darbe klasteriui valdyti šio proceso metu, stebėti darbo našumą bei eigą, taip pat peržiūrėti gautus rezultatus.

### „PuTTY“

„PuTTY“ yra populiarus atvirojo kodo „SSH“ (angl. „Secure Shell“) ir „Telnet“ klientas, skirtas „Windows“ operacinei sistemai, leidžiantis vartotojams saugiai pasiekti nuotolines sistemas. Nors „PuTTY“ nėra tiesiogiai susijęs su „Hadoop“, jis gali būti naudingas įrankis valdyti jo grupes. Naudodami „PuTTY“ vartotojai gali lengvai prisijungti prie pagrindinio „Hadoop“ klasterio mazgo arba atskirų mazgų ir atlikti tokias užduotis kaip „Hadoop“ komandų vykdymas, žurnalų tikrinimas ir konfigūracijos failų redagavimas. Be to, „PuTTY“ terminalo emuliacijos funkcijos leidžia vartotojams pasiekti „Hadoop“ komandinės eilutės sąsają (angl. „CLI“), todėl lengviau atlikti sudėtingas operacijas ir šalinti triktis. Šis klientas buvo naudojamas pirmajame laboratoriniame darbe nuotoliniu būdu pasiekti ir valdyti „Hadoop“ klasterį, vykdyti „Hadoop“ komandas ir stebėti apdorojimo eigą.

### „Apache Spark“

„Apache Spark“ yra atvirojo kodo paskirstyta skaičiavimo sistema, skirta dideliems duomenų kiekiams lygiagrečiai apdoroti kompiuterių klasteriuose. „Spark“ yra naudojamas dėl greito apdorojimo greičio, lankstumo ir naudojimo paprastumo. Jis naudoja talpyklą atmintyje ir optimizuotą užklausų valdymą, kad užtikrintų didelio našumo paketinio, srautinio ir mašininio mokymosi darbo krūvių apdorojimą. „Spark“ gali integruotis su įvairiais duomenų šaltiniais, įskaitant „Hadoop Distributed File System“ („HDFS“) ir kitus. Jo pagrindinė abstrakcija yra elastingas paskirstytas duomenų rinkinys (angl. „RDD“), leidžiantis gedimams atsparų lygiagretų apdorojimą mašinų klasteriuose. Pagrindinė „Apache Spark“ skaičiavimo modelio koncepcija yra „DAG“ programavimo modelis. „Spark“ programoje „DAG“ yra loginės operacijų sekos, reikalingos tam tikram duomenų transformacijų rinkiniui apskaičiuoti, vaizdas. „Spark“ elastingas paskirstytas duomenų rinkiniai yra nekintantys ir gali būti pakeisti tik sukūrus naują. Kiekviena transformacija sukuria naują „RDD“, kuri priklauso nuo pirminio, ir šios priklausomybės yra vaizduojamos kaip kaštai „DAG“ modelyje. Kai „RDD“ rinkiniui yra iškviečiamas veiksmas, „Spark“ įvertina „DAG“ modelį ir tam tikra tvarka atlieka reikiamas transformacijas, kad gautų norimą rezultatą.

### „Google Colab“

„Google Colab“ yra nemokama debesų kompiuterija pagrįsta platforma, kuri suteikia prieinamą ir patogią aplinką paleisti kodą žiniatinklio aplinkoje. Vienas iš pagrindinių „Google Colab“ pranašumų yra galimybė teikti nemokamą prieigą prie tokių išteklių kaip procesorius, operatyvioji atmintis ir saugykla, todėl tai yra patraukli galimybė duomenų apdorojimo ir analizės užduotims atlikti. „Google Colab“ suteikia patogią platformą darbui su „Apache Spark“. Naudojant „PySpark“ biblioteką, kuri yra „Spark API Python“ apvalkalas, naudotojai gali rašyti ir paleisti „Spark“ kodą „Google Colab“ užrašų knygutėje.

# Pirmojo uždavinio sprendimas

Šiame skyriuje bus aptartas pirmojo laboratorinio darbo sprendimas, pateiktas pasiūlymas kaip būtų galima įtraukti „Combiner“ tipo uždavinį į užduoties sprendimą bei pagrįsti tokio įtraukimo naudą, bei aprašyta kaip buvo naudojamas „Hadoop MapReduce“ programavimo modelis.

## „MapReduce“ programavimo modelis

Šiame skyriuje bus pateiktas „MapReduce“ programavimo modelis kaip iliustracija naudojant pirmojo laboratorinio darbo medžiagą. Kaip anksčiau ir buvo minėta „MapReduce“ struktūra apima dviejų žingsnių procesą: „Map“ ir „Reduce“. „Map“ fazė yra pavaizduota (1 pav.). Šiame žingsnyje yra sukamas ciklas per duomenų eilutes, kuriose yra ieškomi parametrai “svorio grupe” ir “svoris”. Tuomet surastos parametrų reikšmės yra konvertuojamos į „tuple“ duomenų tipo poras, kuriuose pirmasis elementas yra raktas, o antrasis reikšmė. Prieš „Map“ fazę buvo paruošti duomenys juos išskaidant ir suplokštinant.

A picture containing text, screenshot

Description automatically generated

pav. Duomenys prieš paruošimą „Map“ fazei

A screen shot of a computer code

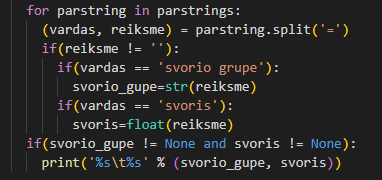
Description automatically generated with low confidence

pav. Duomenų paruošimo kodas prieš „Map“ fazę

A picture containing text, screenshot, font, information

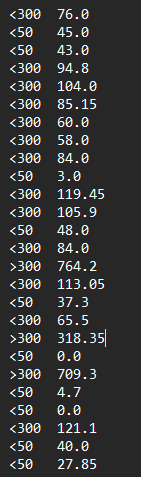
Description automatically generated

pav. Paruošti duomenys prieš „Map“ fazę



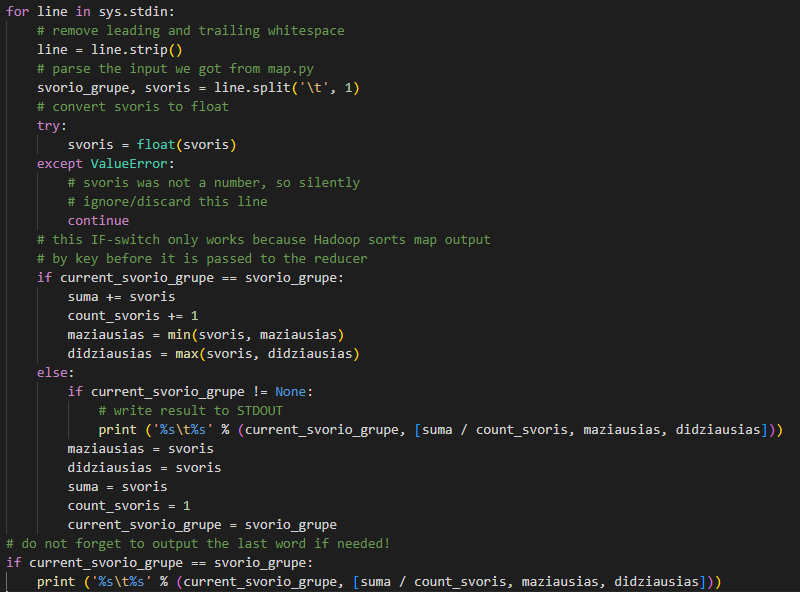
4 pav. „Map“ fazė

Po „Map“ fazės yra gaunamos surastos poros, kurių raktas yra svorio grupės reikšmės, o reikšmės - svorio parametrai (2 pav.).



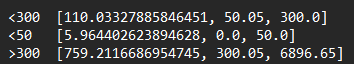
5 pav. Duomenys gauti po „Map“ fazės

„Reduce“ žingsnyje, visos tos pačios raktų reikšmės yra grupuojamos ir suskaičiuojamos, o rezultatai yra grąžinami kaip raktas ir sumažinta reikšmė. Spręsto uždavinio atveju, kaip matome iš (3 pav.) sukant ciklą per gautus duomenis po „Map“ fazės yra tikrinama ar nepasikeitė rakto reikšmė ir yra grupuojamos ir skaičiuojamos porų reikšmių vertės kol nepasikeičia raktas t.y. visos tos pačios raktų reikšmės. Taip yra mažinamos raktų reikšmės kol nebelieka kitų porų su tokiu pačiu raktu.



6 pav. „Reduce“ fazė

Po „Reduce“ fazės yra gaunamos rezultatų poros (4 pav.), kuomet rakto vertė yra unikali, o poros reikšmė yra masyvas. Bendru atveju poros reikšmė gali būti vienas skaičius ar kitoks duomenų tipas. Pirmajam laboratoriniam darbui išspręsti reikėjo trijų statistinių verčių, todėl buvo nuspręsta sukurti masyvą kuriame būtų saugomos tos reikšmės.



7 pav Duomenys po „Reduce“ fazės

Taigi, pirmajam uždaviniui išspręsti „MapReduce“ programavimo modelio dalis „Map“ buvo panaudota reikiamų duomenų atrinkimui iš bendro duomenų rinkinio, o „Reduce“ - statistinių verčių apskaičiavimui.

## „Combiner“ tipo užduoties įtraukimas

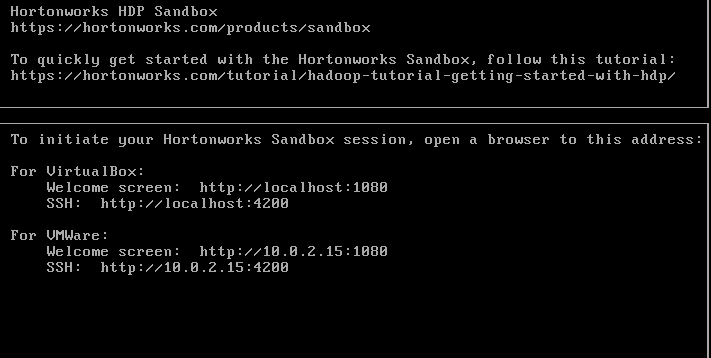
„Hadoop“ platformoje „Combiner“ tipo užduotis yra funkcija, leidžianti sumažinti duomenų kiekį, kuris yra įterpiamas tarp „Map“ ir „Reduce“ fazių. „Combiner“ funkcijos įvestis yra „Map“ fazės išvestis, o išvestis yra „Reduce“ fazės įvestis. Dažniausiai „Combiner“ funkcija yra naudojama tuomet, kai „Map“ išvestis sukuria didelį duomenų kiekį, kurį reikia maišyti ir surūšiuoti prieš siunčiant jį į „Reduce“ fazę, kai yra naudojama „Combiner“ funkcija, šis procesas yra vykdomas joje, o ne „Reduce“ fazės metu. Reikia nepamiršti, kad „Combiner“ funkcija turėtų būti asociatyvi ir komutatyvi t.y. ji turėtų sukurti tą pačią išvesti, nepaisant įvesties duomenų apdorojimo tvarkos.

Pirmajame laboratoriniame darbe galima įtraukti „Combiner“ tipo uždavinį, kuris atliks dalinį kiekvienos svorio grupės svorių agregavimą „Map“ mazguose. Šios funkcijos metu būtų sudaromos naujos poros, kurių raktas liktų toks pat, tačiau reikšmėje būtų naujas „tuple“ duomenų tipų rinkinys, kuriame būtų suskaičiuotas visas svorio grupės svoris ir kiek svorių buvo rasta toje svorių grupėje. Be šios funkcijos „Reduce“ fazė gautų visas tarpines raktų-reikšmių poras iš visų „Map“ mazgų ir turėtų atlikti daug didesnio duomenų rinkinio agregavimą. Pridėjus „Combiner“, galima sumažinti duomenų kiekį, kurį „Reduce“ fazė turi apdoroti, o tai gali lemti greitesnį užduočių vykdymo laiką ir efektyvesnį „Hadoop“ klasterio išteklių naudojimą.

## „Hadoop MapReduce“ naudojimas

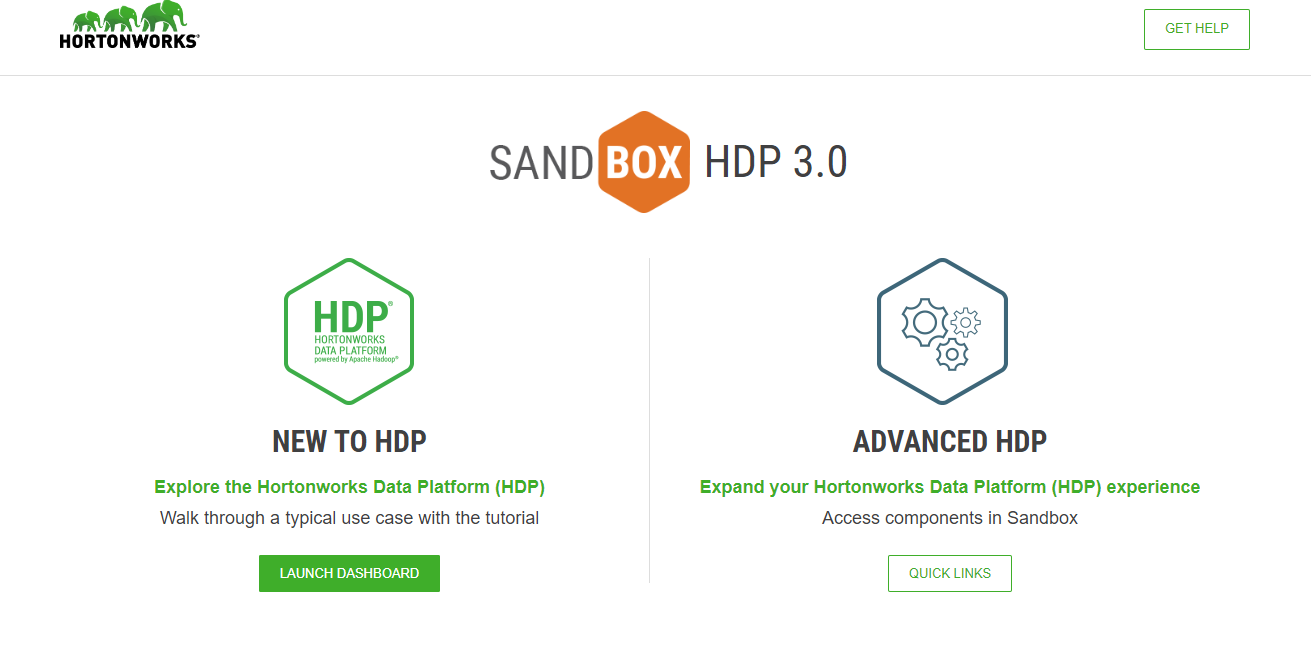
Norint naudotis „Hadoop MapReduce“ sistema sprendžiant pirmojo laboratorinio darbo uždavinį ir naudojant „Windows“ operacinę sistemą ir „VirtualBox“ virtualizacijos technologiją reikėtų atlikti šiuos veiksmus:

1. Perskaičius užduoties aprašymą nustatyti kokios rakto - reikšmių poros yra reikalingos šiam uždaviniui išspręsti.
2. Parašyti „MapReduce“ kodą. „Map“ fazė apdoroja kiekvieną įvesties įrašą ir sudaro tarpines rakto - reikšmių poras. „Reduce“ fazė apdoroja tarpinius duomenis ir sukuria galutinę išvestį.
3. „Hadoop“ klasterio konfigūravimas:
   1. Atsisiųsti „VirtualBox“ virtualizacijos technologiją. „WinSCP“ ir „PuTTY“ klientus.
   2. Atsisiųsti „Hortonworks“ duomenų platformą „Hortonworks“ smėliadėžėje savo anksčiau atsisiųstai virtualizacijos technologijai.
   3. Atsidarius virtualizacijos technologiją reiktų importuoti atsisiųstą failą ir paleisti virtualią mašiną. Tinkamai pasileidus virtualiai mašinai turite matyti vaizdą esantį (5 pav.)



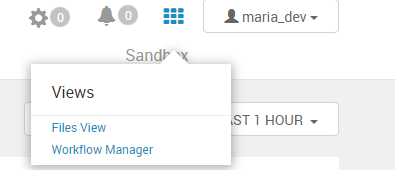
8 pav. Vaizdas paleidus „Hortonworks HDP Sandbox“ naudojant „VirtualBox“

* 1. Tuomet reikėtų atsidaryti hosts failą administratoriaus teisėmis, kuris yra šioje c:\Windows\System32\drivers\etc\hosts direktorijoje ir pridėti tokią eilutę {IP-Address} localhost sandbox-hdp.hortonworks.com sandbox-hdf.hortonworks.com
  2. Naršyklėje reikėtų įvestį adresą <http://localhost:1080>. Jeigu atsidaro langas, koks yra (6 pav.) tuomet viskas pasileido tinkamai.



9 pav. „Sandbox“ paleidimas

* 1. Paspaudus „Launch Dashboard“ reikės įvesti prisijungimo duomenis, kurie pagal nutylėjimą yra maria\_dev ir maria\_dev. Atsidariusiame puslapyje viršutiniame dešiniajame kampe reiktų paspausti mygtuką „Files View“ kuris yra pavaizduotas (7 pav.).



10 pav. „Files View“ mygtukas

* 1. Atsidariusiame lange („Ambari“) reikėtų nueiti į direktoriją adresu: user/maria\_dev ir joje sukurti naują bylą į kurią reikia įkelti   
     „MapReduce“ kodą.
  2. Atsidarius „WinSCP“ reiktų suvesti tokius duomenis: „Host name“ –„ localhost“, „Port number“ - 2222, „User name“ – „maria\_dev“, „Password“ – „maria\_dev“. Prisijungus reikėtų nueiti į /home/maria\_dev direktoriją ir joje susikurti savo bylą (tokiu pačiu pavadinimu kaip „Ambari“), į kurią reikės įkelti tą patį „MapReduce“ kodą.
  3. Vis dar esant „WinSCP“ atsidaryti sesiją „PuTTY“ terminale, suvesti anksčiau minėtus prisijungimus.
  4. Failams, kurie turi būti paleidžiami, laboratorinio darbo atveju t.y. „map.py“ ir „reduce.py“ turi būti uždėtos „execute“ teisės. Galima pasitikrinti ar tokios teisės uždėtos, jeigu „PuTTY“ terminale įvedus „ls“ komandą (reikia būti sukurtos bylos direktorijoje, tą galima padaryti įvedus komandą „cd“ ir nurodžius direktoriją) ir jeigu paleidžiamieji failai yra žalios spalvos, tuomet tokios teisės yra uždėtos jeigu ne reikėtų paleisti komandą: „chmod +x {{failo pavadinimas}}“.
  5. Norint paleisti „MapReduce“ programavimo modelį reikėtų paleisti šią komandą: „mapred streaming -input /user/maria\_dev/test/AA/lab1/duom\_cut.txt -output /user/maria\_dev/test/AA/lab1/out1 -mapper map.py -reducer red.py -file map.py -file red.py“ (vietoj „/test/AA/lab1“ turėtų būti tokia pati direktorija kaip ir „Ambari“ sistemoje) . Kur už „-input“ parametro reikia įvesti direktoriją iki duomenų rinkinio, už „-output“ parametro yra nurodoma direktorija išvesčiai, už „-mapper“ parametro yra nurodomas failas, kuriame yra aprašyta „Map“ fazė, už „-reducer parametro“ yra nurodomas failas, kuriame yra aprašyta „Reduce“ fazė, už „-file“ parametro yra prikabinamas „map.py“ ir „reduce.py“ failai, tuo atveju jeigu platforma nerastų reikiamų failų.
  6. Jeigu viskas pasileido gerai, tuomet „PuTTY“ terminale turėtumėt matyti vaizdą esantį (8 pav.).



11 pav. „PuTTY“ terminalo vaizdas, kuomet „MapReduce“ modelis sėkmingai įvykdytas

Tuomet nuėjus į „Ambari“ direktoriją, kuri buvo nurodyta kompandoje kaip išvesties direktorija, turėtumėt matyti naujai sukurta aplankalą, kurio viduje yra du failai (9 pav.) – „\_SUCCESS“ ir „part-00000“, pastarajame yra „MapReduce“ programavimo modelio rezultatas.



12 pav. Sėkmingai paleistas „MapReduce“ modelis per „Hadoop“ sistemą

# Antrojo uždavinio sprendimas

Šiame skyriuje bus aptartas antro laboratorinio darbo sprendimas, aprašytas „DAG“ programavimo modelio veikimas, bei aptartos „Spark“ transformacijos ir veiksmai, kurie buvo panaudoti antrajame laboratoriniame darbe.

## „DAG“ programavimo modelis

Šiame skyriuje bus pateiktas „DAG“ programavimo modelio veikimas kaip iliustracija naudojant antrojo laboratorinio darbo medžiagą. Šis modelis pasižymi acikliškumu t.y. jame nėra ciklų ar kilpų. Šio modelio duomenų priklausomybės rodo priklausomybę tarp operacijų. Jos rodo, kad operacijai įvesti reikia ankstesnės operacijos išvesties duomenų. Tai leidžia „DAG“ modeliui užfiksuoti priklausomybes ir užtikrinti, kad operacijos būtų vykdomos tinkama tvarka. Šis modelis taip pat pasižymi topologine tvarka t.y. operacijos išdėstomos linijine tvarka taip, kad nė viena operacija nepriklauso nuo vėlesnės operacijos. „DAG“ programavimo modelio struktūra susideda iš šių elementų: pirminis „RDD“, transformacijos, veiksmai ir išvestis.

### Pirminis „RDD“

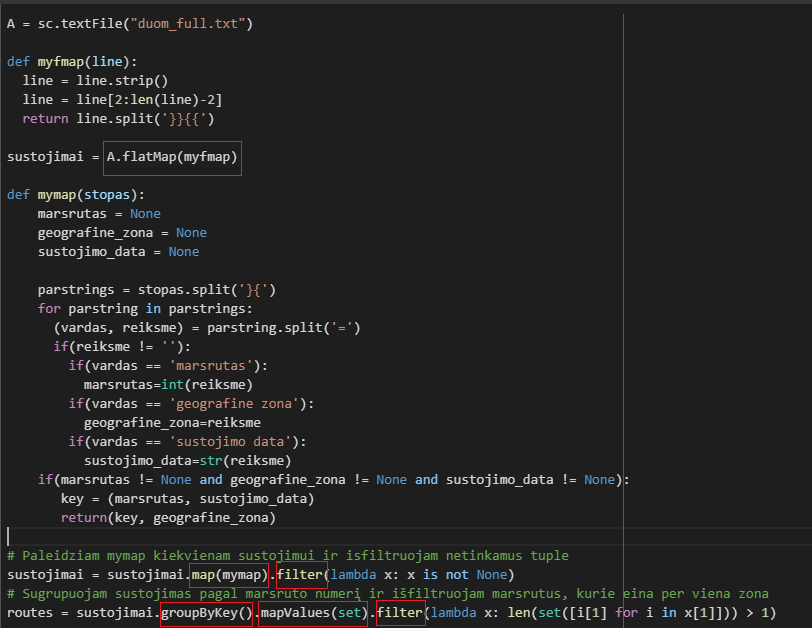
Pirminiu „RDD“ vadiname „RDD“ (Atsparūs paskirstytieji duomenų rinkiniai), kuriame yra įvesties duomenys. Pavyzdžiui (10 pav.), pirminį „RDD“ galima nuskaityti iš failo.



13 pav. Pirminis „RDD“

### Transformacijos

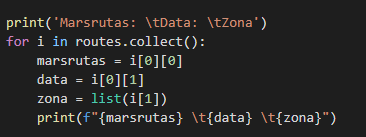
Transformacijos tai yra funkcijos, kurios yra taikomos „RDD“ rinkiniui. Šios transformacijos gali apimti tokias operacijas kaip „map()“, „filter()“, „join()“ ar bet kurią kitą „Spark“ sistemoje prieinamą transformaciją. Kiekviena transformacija sukuria naują „RDD“ rinkinį. Pavyzdžiui, (11 pav.) buvo pritaikytos penkios transformacijos („flatMap()“, „map()“, „filter()“, „groupByKey()“, „mapValues()“) duomenų sutvarkymui ir norimo rezultato gavimui.



14 pav. „RDD“ transformacijos

### Veiksmai

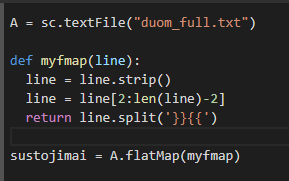
Veiksmai atvaizduoja veiksmo operaciją, taikomą transformuotam „RDD“ rinkiniui. Veiksmai inicijuoja „DAG“ modelio vykdymą, nurodydami „Spark“ atlikti skaičiavimus ir grąžinti rezultatą. Veiksmų pavyzdžiai: „collect()“, „count()“, „reduce()“, „save()“ ar bet kuris kitas „Spark“ prieinamas veiksmas. Pavyzdžiui (12 pav.) veiksmas „collect()“ paleidžia „DAG“ modelio vykdymą ir surenka rezultatus, grąžindamas juos kaip sąrašą.



15 pav. „RDD“ veiksmai

### Duomenų priklausomybės

Duomenų priklausomybės apibūdina ryšį tarp „RDD“ rinkinių pagal jų įvesties ir išvesties duomenis. Yra dviejų tipų priklausomybės: Siauros priklausomybės: kiekvieną tėvinės „RDD“ rinkinio skaidinį naudoja ne daugiau kaip vienas antrinės „RDD“ skaidinys. Siauros priklausomybės leidžia efektyviau apdoroti duomenis, nes nereikia permaišyti duomenų ar bendrauti tinkle. Plačios priklausomybės: kiekvieną tėvinės „RDD“ rinkinio skaidinį gali naudoti keli antrinės „RDD“ skaidiniai. Plačios priklausomybės dažnai susijusios su operacijomis, kurioms reikia permaišyti duomenis arba palaikyti ryšį tinkle. Skaitant kodą duomenų priklausomybę galima stebėti netiesiogiai, naudojant „RDD“ rinkinių taikomų transformacijų ir veiksmų seką. Norint pastebėti duomenų priklausomybes reikia ieškoti transformacijų, kurios keičia iš ankstesnių „RDD“ sukurtus naujus „RDD“ rinkinius. Pavyzdžiui (13 pav.) „RDD“ pavadinimu “sustojimai” yra išvesta iš „RDD“ rinkinio pavadinimu „A“ per „flatMap()“ transformaciją. Tai rodo duomenų priklausomybę tarp šių dviejų „RDD“.



16 pav. Duomenų priklausomybės

### Acikliškumas

Acikliškumas „DAG“ modelyje reiškia, kad duomenų apdorojimo sraute nėra ciklų ar kilpų. Kitaip tariant, tai reiškia, kad „Spark“ programoje tarp „RDD“ rinkinių ir transformacijų/veiksmų nėra žiedinių priklausomybių. Skaitant kodą acikliškumą galima stebėti netiesiogiai, naudojant „RDD“ taikomų transformacijų ir veiksmų seką. Jei kode nėra rekursinių ar savireferentinių operacijų, dėl kurių galėtų atsirasti ciklų, „DAG“ modelis pagal nutylėjimą yra acikliškas.

### Topologinė tvarka

„DAG“ modelyje topologinė tvarka reiškia linijinę seką, kuria „RDD“ rinkiniai ir transformacijos/veiksmai atliekami duomenims apdoroti. Ji rodo tvarką, kuria turi būti atliekamos operacijos, kad būtų patenkintos priklausomybės tarp „RDD“.

## „Spark“ transformacijos ir veiksmai

Šiame skyriuje bus aprašytos „Spark“ transformacijos ir veiksmai, kurie buvo panaudoti antrajame laboratoriniame darbe. Pakomentuoti transformacijų tipai, bei duomenų siuntimai dėl transformacijų.

Yra du „Apache Spark RDD“ operacijų tipai - transformacijos ir veiksmai. Transformacija yra funkcija, kuri iš esamų „RDD“ sukuria naują „RDD“ rinkinš, bei kai norima dirbti su tikruoju duomenų rinkiniu, tuomet atliekamas veiksmas. Kai jis yra paleidžiamas po rezultato, naujas „RDD“ nėra formuojamas kaip transformacijos metu.

### „RDD“ rinkinių transformacijos

„Spark“ transformacijos yra funkcija, kuri iš esamų „RDD“ sukuria naują „RDD“ rinkinį. Ji priima „RDD“ kaip įvestį ir sukuria vieną ar daugiau „RDD“ rinkinį kaip išvestį. Kiekvieną kartą ji sukuria naują „RDD“, kai taikoma bet kokia transformacija. Taigi, tokiu būdu „RDD“ rinkinio įvestis negali būti pakeista, nes „RDD“ yra nekintamos prigimties. Transformacijos yra vykdomos, kai yra iškviečiamas veiksmas. Jos nėra vykdomos iš karto. Po transformacijos gautas „RDD“ visada skiriasi nuo savo pirminio „RDD“ rinkinio. Jis gali būti mažesnis (pavyzdžiui taikant, „filter()“, „count()“, „distinct()“, „sample()“ transformacijas), didesnis (pavyzdžiui taikant, „flatMap()“, „union()“, „cartesian()“ transformacijas) arba tokio paties dydžio (pavyzdžiui taikant, „map()“ transformaciją). Yra dviejų tipų transformacijos: siauros ir plačios. Siauros transformacijos metu visi elementai reikalingi vieno skirsnio įrašams apskaičiuoti, yra viename tėviniame „RDD“ skirsnyje. Rezultatams apskaičiuoti yra naudojamas ribotas skirsnio poaibis. Siauros transformacijos yra: „map()“, „filter()“ rezultatas. Taikant plačiąją transformaciją, visi elementai, reikalingi vieno skirsnio įrašams apskaičiuoti, gali būti daugelyje „RDD“ rinkinio skirsnių. Dalis gali būti daugelyje tėvinės „RDD“ dalių. Plačios transformacijos yra „groupbyKey()“ ir „reducebyKey()“ rezultatas.

### „RDD“ rinkinių veiksmai

„Spark“ programoje veiksmas yra operacija, kuri sukelia skaičiavimus „RDD“ ir grąžina rezultatą. Transformacijos sukuria „RDD“ vieną iš kitos, tačiau kai norima dirbti su tikruoju duomenų rinkiniu, tuomet yra atliekami veiksmai. Kai jis yra atliekamas po rezultatom naujas „RDD“ nesukuriamas kaip transformuojant. Taigi, veiksmai yra „RDD“ operacijos, kuriomis gaunamos ne „RDD“ reikšmės. Veiksmo reikšmės yra saugomos tvarkyklėse arba išorinėje saugojimo sistemoje.

### Duomenų siuntimas dėl transformacijų

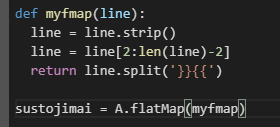
Kai „RDD“ rinkiniams yra taikomos transformacijos, duomenys perduodami tinkle, tame pačiame mazge arba tarp klasterio mazgų. Perduodamų duomenų kiekis priklauso nuo transformacijos tipo ir „RDD“ dydžio. Atliekant siaurąsias transformacijas paprastai perduodama mažiau duomenų nei plačiąsias, nes jos gali būti atliekamos lygiagrečiai kiekvienam skirsniui, Kita vertus, plačioms transformacijoms reikia perduoti daugiau duomenų, nes duomenis reikia išmaišyti per visą tinklą.

### Naudotos „Spark“ transformacijos ir veiksmai

Šiame skyriuje bus aprašytos „Spark“ transformacijos ir veiksmai, kurie buvo panaudoti antrajame laboratoriniame darbe. Šiai užduočiai išspręsti buvo panaudotos penkios transformacijos ir vienas veiksmas.

1. „flatMap()“

Naudojant „flatMap()“ funkciją, kiekvienam įvesties elementui turime daug elementų išvesties „RDD“ rinkiių. Laboratorinio darbo atveju ši transformacija taikoma „A“ „RDD“ rinkiniui (14 pav.) (kuris yra sukuriamas skaitant iš tekstinio failo), siekiant suplokštinti eilutes, padalijant jas pagal skirtuką „}}{{„. Pirminis duomenų rinkinys yra matomas (15 pav.). Kaip pasikeičia duomenys po „flatMap()“ transformacijos panaudojimo galima pamatyti (16 pav.).



17 pav. „flatMap()“ panaudojimas antrajame laboratoriniame darbe

A picture containing text, screenshot

Description automatically generated

18 pav. Pirminis duomenų rinkinys

A screen shot of a computer screen

Description automatically generated with low confidence

19 pav. Duomenys po „flatMap()“ panaudojimos

1. „map()“

„map()“ funkcija iteruoja kiekvieną „RDD“ rinkinio eilutę ir padalina ją į naują „RDD“. Naudojant „map()“ transformaciją reikia įvesti bet kokią funkciją ir pastaroji bus taikoma kiekvienam „RDD“ elementui. Naudojant „map()“ yra galimybė lanksčiai pasirinkti, kad „RDD“ rinkinio įvesties ir išvesties tipai gali skirtis vienas nuo kito. Pavyzdžiui, įvesties „RDD“ tipas gali būti „String“, o pritaikius funkciją „map()“, grįžtantis „RDD“ rinkinys gali būti „Boolean“ tipo. Laboratorinio darbo atveju (17 pav.) ši transformacija yra taikoma kiekvienam „sustojimo“ „RDD“ elementui naudojant funkciją „mymap()“, kad būtų išskirta svarbi informacija ir sukurta rakto ir vertės pora. Rezultatą po „map()“ panaudojimo galima pamatyti (18 pav.)



20 pav. „map()“ ir „filter()“ panaudojimas antrajame laboratoriniame darbe

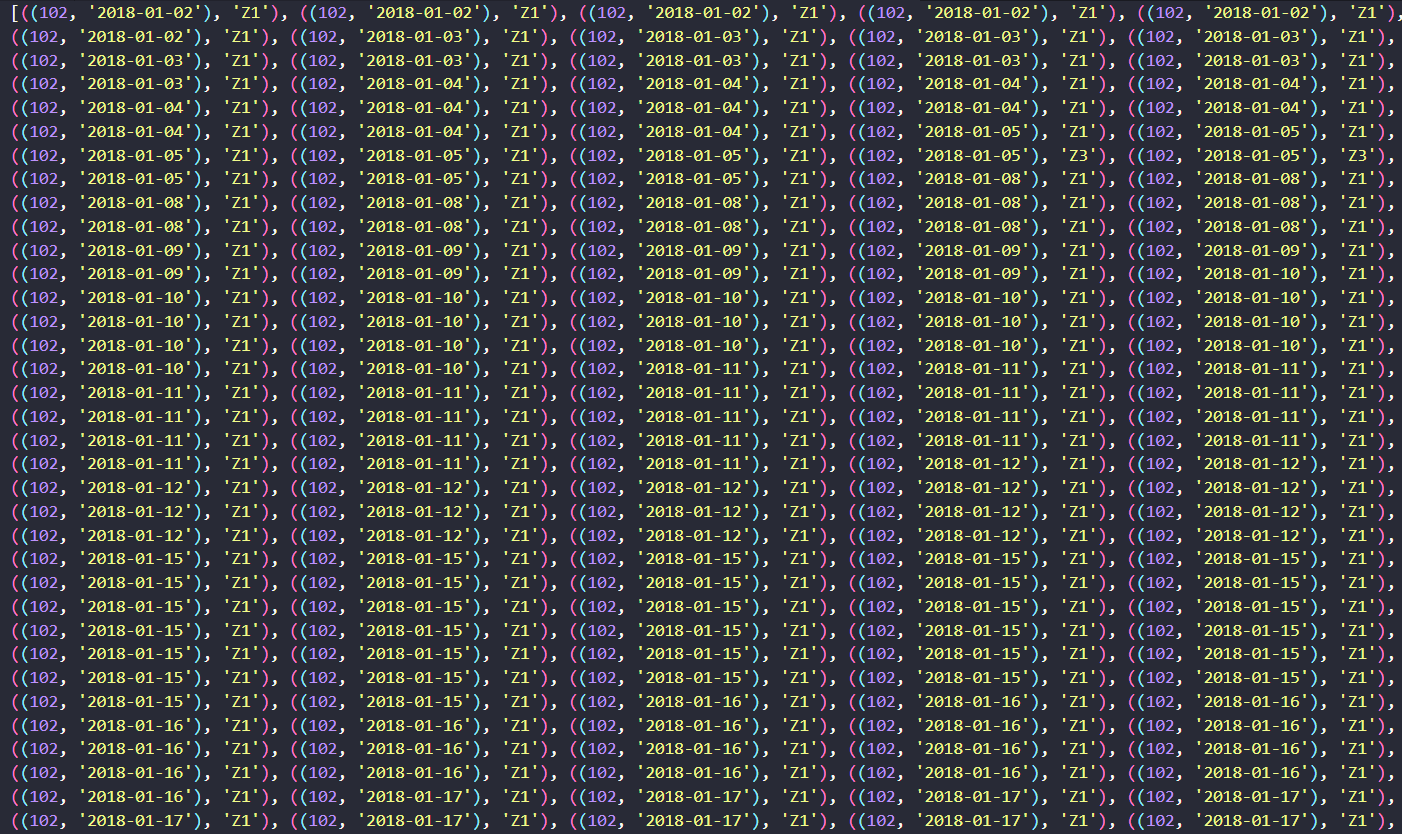
A screen shot of a computer screen

Description automatically generated with low confidence

21 pav. Duomenys po „map()“ panaudojimo

1. „filter()“

„filter()“ funkcija grąžina naują „RDD“, kuriame yra tik elementai, atitinkantys sąlygą. Tai siaura operacija, nes ji neperkelia duomenų iš vieno skirsnio į daugelį skirsnių. Laboratorinio darbo atveju (17 pav.) ši transformacija yra atliekama kiekvienam sustojimo „RDD“ rinkinio elementui, kuris iš „RDD“ pašalina reikšmes kurios yra „None“. Taip pat (20 pav.) taikoma „mapValues()“ rezultatui, kad būtų išfiltruoti visi raktai, kurių skirtingų geografinių zonų skaičius yra mažesnis arba lygus 1. Rezultatus po „filter()“ panaudojimo galima matyti (19 pav.) ir



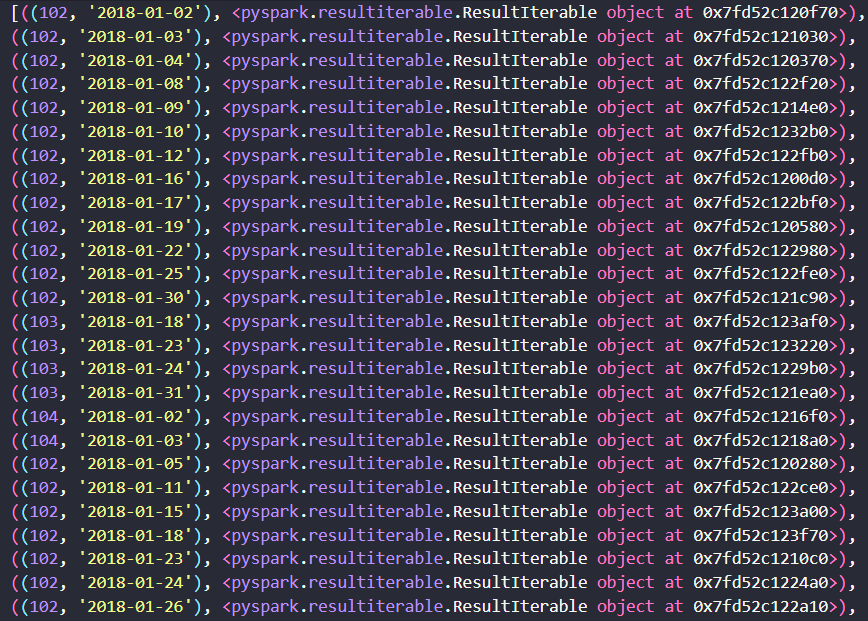
22 pav. Duomenys po „filter()“ panaudojimo kai yra išfiltruojamos „None“ reikšmės

1. „groupByKey()“

Naudojant funkcija „groupByKey()“ duomenų rinkiniui, sudarytam iš (k, v) porų, duomenys išmaišomi pagal rakto „k“ reikšmę kitame „RDD“. Laboratorinio darbo atveju (20 pav.) ši transformacija yra taikoma filtruotam „sustojimai“ „RDD“ rinkiniui, siekiant sugrupuoti elementus pagal raktą. Kaip duomenys atrodo „groupByKey()“ panaudojimo galima matyti (21 pav.)



23 pav. „goupByKey()“, „mapValues()“, „filter()“ panaudojimas antrajame laboratoriniame darbe



24 pav. Duomenys po „groupByKey“ panaudojimo

1. „mapValues()“

Transformacija „mapValues()“ naudojamas poros „RDD“ reikšmėms transformuoti, o raktai išlieka nepakitę. Laboratorinio darbo atveju (20 pav.) ši transformacija yra taikoma transformacijos „groupByKey()“ rezultatui, kad reikšmės būtų paverstos rinkiniu. Kaip pasikeitė duomenys po „mapValues()“ galima matyti (22 pav.)

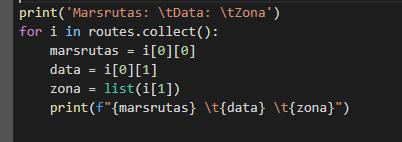
A picture containing screenshot, colorfulness

Description automatically generated

25 pav. Duomenys po „mapValues()“ panaudojimo

1. „collect()“

Veiksmas „collect()“ yra įprasta ir paprasčiausia operacija, kuri grąžina „RDD“ turinį į tvarkyklės programą. Laboratorinio darbo atveju (23 pav.) šis veiksmas taikomas galutiniam „RDD“ rinkiniui – ‚routes“, kad rezultatas būtų grąžintas kaip sąrašas. Rezultatus po „collect()“ galima pamatyti (24 pav.)



26 pav. „collect()“ panaudojimas antrajame laboratoriniame darbe

A picture containing text, screenshot, font, number

Description automatically generated

27 pav. Duomenys po „collect()“ panaudojimo

# Trečiojo uždavinio sprendimas

Šiame skyriuje bus aprašytas trečiasis laboratorinis darbas. „Python“ programavimo kalbos pagalba vizuolizuota duomenų aproksimacija su tiesinės regresijos modeliu. Pateiktos ir pakomentuotos „Python“ kalbos bibliotekos ir komandos, kurios buvo panaudotas rezultatui gauti. Pakomentuoti duomenų pertvarkymai tarp skirtingų formatų/duomenų struktūrų. Pakomentuotas atvejis jeigu kodas būtų vykdomas klasteryje, kokios kodo vietos būtų atsakingos už duomenų siuntimus tarp serverio mazgų arba lokalaus kompiuterio.

## Duomenų aproksimacijos vizualizacija

Duomenų aproksimacija – tai trūkstamų ar neišsamių duomenų įvertinimo ar aproksimacijos procesas, grindžiamas turima informacija. Jis apima įvairių matematinių ir statistinių metodų taikymą siekiant užpildyti spragas ir pagrįstai spėti apie nežinomas reikšmes. Duomenų aproksimacija dažnai naudojama tais atvejais, kai nėra išsamių duomenų arba juos brangu gauti. Yra įvairių duomenų aproksimacijos metodų ir algoritmų, kurie priklauso nuo duomenų pobūdžio ir konkrečios sprendžiamos problemos. Kai kurie dažniausiai naudojami metodai yra šie: interpoliavimas, regresija, išlyginimas, matricų pildymas ir imputavimas.

Šiame darbe bus kalbama apie duomenų aproksimacijos metodą – regresija. Jos modeliais siekiama atspindėti pagrindinius duomenų dėsningumus ir tendencijas pritaikant matematinę funkciją žinomiems duomenų taškams. Taikant regresiją siekiama rasti geriausiai tinkančią liniją arba kreivę, kuri minimizuotų prognozuojamų ir faktinių reikšmių skirtumą. Ši linija arba kreivė yra aproksimuoti duomenys ir gali būti naudojami naujiems arba trūkstamiems duomenų taškams prognozuoti. Yra įvairių regresijos modelių tipų, tačiau dažniausiai yra naudojamos tiesinės regresijos modelis. Tiesinėje regresijojje daroma prielaida, kad tarp įvesties kintamųjų ir išvesties kintamojo yra tiesinis ryšys.

Trečiajame laboratoriniame darbe buvo tirta tiesinė priklausomybė parametro „BendraKaina“ nuo parametro „siuntu skaicius“, todėl buvo pritaikytas tiesinės regresijos modelis. Naudojant „Python“ programavimo kalbą buvo vizuolizuota duomenų aproksimacija.

## Naudotos „Python“ bibliotekos ir komandos

### „Matplotlib“

„Matplotlib“ biblioteka yra populiari duomenų vizualizavimo priemonė „Python“ programavimo kalba, kuri suteikia daugybę funkcijų statiniams, animuotiems ir interaktyviems grafikams kurti. Ši biblioteka plačiai naudojama duomenų analizės srityje, įskaitant tiesinės regresijos modelių rezultatų vizualizavimą. Tiesinės regresijos kontekste „Matplotlib“ dažnai naudojamas faktiniams duomenų taškams ir aproksimaciją vaizduojančiai regresijos tiesei ar kreivei nubraižyti. „Matplotlib“ siūlo įvairias pritaikymo parinktis, leidžiančias audotojams nustatyti spalvas, etiketes, pavadinimus, ašis, legendas ir dar daugiau, kad vizualizacijos būtų aiškesnės ir estetiškesnės. Trečiame laboratoriniame darbe „Matplotlib“ biblioteka buvo naudojama tiesinės regresijos grafiko braižymui (30 pav.). Šio grafiko braižymui buvo panaudotos šios komandos:

* + - „figure()“- ši funkcija buvo panaudotas kartu su „figsize“ parametru (25 pav.). Šioje eilutėje sukuriamas naujas figūros objektas su konkrečiu figūros dydžiu – 10 colių pločio ir 5 colių aukščio. Parametras „figsize“ yra neprivalomas.



28 pav. „figure()“ komandos/funkcijos panaudojimas

* + - „scatter()“- ši funkcija buvo panaudota kuriant sklaidos grafiką (26 pav.). Kaip „x“ reikšmės naudojamos „labels“, o kaip „y“ reikšmės – „values“. Argumentas „s=10“ nustato, kad kiekvieno taško žymeklio dydis būtų 10.



29 pav. „scatter()“ komandos/funkcijos panaudojimas

* + - „xlabel()“ – ši funkcija buvo panaudota „x“ ašies etiketei nustatyti (27 pav.). Šioje eilutėje grafiko „x“ ašies etikėtė nustatoma kaip „BendraKaina“. Joje yra pateikiamas „x“ ašies vaizduojamo kintamojo aprašymas arba pavadinimas.



30 pav. „xlabel()“ komandos/funkcijos panaudojimas

* + - „ylabel()“ – ši funkcija buvo panaudota „y“ ašies atiketei nustatyti (28 pav.). Šioje eilutėje grafiko „y“ ašies etiketė nustatoma kaip „SiuntuSkaicius“. Joje yra pateikiamas „y“ ašies vaizduojamo kintamojo apraųymas arba pavadinimas.



31 pav. „ylabel()“ komandos/funkcijos panaudojimas

* + - „show()“ - ši funkcija buvo panaudota grafiko nubraižymui (29 pav.). Šioje eilutėje sukurtas grafikas yra rodomas ekrane. Šią eilutę yra būtina įtraukti norint vizualizuoti grafiką.



32 pav. „show()“ komandos/funkcijos panaudojimas

A picture containing screenshot, text, line, diagram

Description automatically generated

33 pav. Grafikas nubraižytas naudojant „Matplotlib“ biblioteką

### „Pandas“

„Pandas“ yra galinga duomenų tvarkymo ir analizės biblioteka „Python“ programavimo kalba. Joje pateikiamos duomenų struktūros, pavyzdžiui „DataFrame“, kurios leidžia lengvai tvarkyti struktūrizuotos duomenis ir jais manipuliuoti. „Pandas“ siūlo daugybę funkcijų ir metodų, skirtų duomenų valymui, išankstiniam apdorojimui, apibendrinimui ir analizei. Trečiame laboratoriniame darbe „Pandas“ biblioteka buvo panaudota „Spark DataFrames“ duomenų stuktūros konvertavimui į „Pandas DataFrames“ naudojant funkciją „toPandas()“ (31 pav.). Rezultatus prieš duomenų stuktūros konvertavimą galima pamatyti (32 pav.), o po – (33 pav.).



34 pav. „Pandas“ bibliotekos panaudojimas



35 pav. duomenys prieš „Pandas“ bibliotekos panaudojimą

A screenshot of a computer screen

Description automatically generated with low confidence

36 pav. duomenys po „Pandas“ bibliotekos panaudojimo

### „pyspark.ml“

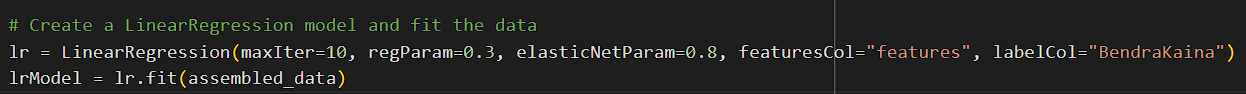
„Pyspark.ml“ yra „Python“ biblioteka, kuri yra „Apache Spark“ mašininio mokymosi bibliotekos dalis. Jojie pateikiama aukšto lygio „API“, skirta kurti ir vykdyti keičiamo mastelio mašininio mokymosi „pipelines“ didelės apimties paskirstytiems duomenims naudojant „Spark“ paskirstytųjų skaičiavimų galimybes. Šioje bibliotekoje siūlomi įvairūs algoritmai ir pagalbinės priemonės, skirtos įvairios mašininio mokymosi užduotims, įskaitant klasifikavimą, regresiją, klasterizavimą, matmenų mažinimą ir požymių išskyrimą. Joje laikomasi nuoseklios „API“ struktūros, todėl nesudėtinga keisti skirtingus algoritmus ir metodus be didelių kodo pakeitimų. Vienas iš pagrindinių „puspark.ml“ komponentų yra „DataFrame“ pagrįstų mašininio mokymosi „pipeline“ koncepcija. Jis susideda iš kelių etapų, pavyzdžiui, išankstinio duomenų apdorojimo, požymių inžinerijos, modelio mokymo ir vertinimo. Kiekvieną etapą atvaizduoja transforatorius arba vertintojas. Transformatoriai transformuoja vieną „DataFrame“ į kitą, o vertintojai yra algoritmai arba modeliai, kuriuos galima pritaikyti „DataFrame“, kad būtų sukurtas transformatorius. Trečiajame laboratoriniame darbe „pyspark.ml“ biblioteka buvo naudojama mašininiam mokymuisi.

* Duomenų pirminis apdorojimas: Duomenys iš anksto apdorojami naudojant įvairias operacijas, pavyzdžiui, filtravimą, atvaizdavimą ir sujungimą, kad jie būtų transformuoti į mašininiam mokymuisi tinkamą formatą. Duomenų požymiams surinkti į vieną požymių vektorių naudojama „pyspark.ml.feature()“ klasė „VectorAssembler“. Ji paima įvesties stulpelius, trečiojo laboratorinio darbo atveju – stulpelį „SiuntuSkaicius“, ir sukuria naują stulpelį pavadinimu „features“ (32pav.), kuriame yra surinktas požymių vektorius.



37 pav. „pyspark.ml.frature()“ bibliotekos panaudojimas duomenų pirminiui apdorojimui

* Modelio mokymas: trečiojo laboratorinio darbo atveju, požymių ir tikslinio kintamojo „BendraKaina“ ryšiui modeliuoti pasirinktas tiesinės regresijos algoritmas. Sukuriamas klasės „LinearRegression“ iš „pyspark.ml.regression“ egzempliorius (angl. „instance“), nurodant norimus parametrus, tokius kaip maksimalus iteracijų skaičius, reguliarizacijos parametras ir elastingo tinklo parametras. Modelis (36 pav.) apmokytas naudojant „fit()“ metodą (33 pav.) iš anksto apdorotiems duomenims „assembled\_data“.



38 pav. „pyspark.ml.feature()“ bibliotekos panaudojimas modelio mokymuisi



39 pav. Sudarytas modelis

* Modelio įvertinimas ir rodikliai: Siekiant įvertinti apmokyto modelio našumą, apskaičiuojamos įvairios metrikos. Apmokyto modelio atributas „summary“ suteikia prieiga prie įvairių vertinimo metrikų ir apmokyto modelio savybių. Trečiojo laboratorinio darbo atveju buvo apskaičiuotos šios metrikos: mokymo metu atliktų iteracijų skaičius, tikslo istorija, liekanos, vidutinė kvadratinė paklaida (RMSE) ir R kvadrato vertė (37 pav.). Trečiojo laboratorinio darbo metu sudaryto modelio rezultatus galima pamatyti (38 pav.)

A picture containing text, screenshot, font

Description automatically generated

40 pav. Modelio įvertinimo rodikliai naudojant „pyspark.ml.feature()“

A screenshot of a computer

Description automatically generated



41 pav. Sudaryto modelio įvertinimo rodikliai

### „pyspark.sql“

„pyspark.sql“ yra „PySpark“ bibliotekos modulis, kuriame pateikiama aukšto lygio sąsaja darbui su struktūrizuotais ir pusiau struktūrizuotais duomenimis naudojant „Spark“. Šiame modulyje siūloma „DataFrame API“, kuri yra paskirstytų duomenų rinkinių abstrakcija, leidžianti naudotojams atlikti į „SQL“ panašias operacijas su didelės apimties duomenų rinkiniais. Šis modulis naudoja galingą „Spark“ paskirstytųjų skaičiavimų variklį, kad būtų galima efektyviai apdoroti ir analizuoti struktūrizuotus duomenis, Jis palaiko įvaieius duomenų šaltinius, tokius kaip „CSV“, „Parquet“, „JSON“ ir duomenų bazes, todėl galima sklandžiai integruoti įvairius duomenų formatus. „pyspark.sql“ palaiko „SQL“ užklausas, todėl vartotojai gali išreikšti sudėtingas duomenų manipuliacijas naudodami „SQL“ sintaksę. Trečiajame laboratoriniame darbe „pyspark.sql“ modulis buvo naudojamas:

* Duomenų skaitymas: kodas skaito duomenis iš tekstinio failo naudodamas „spark.read.option().csv()“ (39 pav.). Ši funkcija yra „pyspark.sql“ modulio dalis ir grąžina „DataFrameReader“ objektą sukurtą iš „CSV“ failo. Rezultatus po „pyspark.sql“ panaudojimo galima pamatyti (40 pav.).



42 pav. „pyspark.sql“ modulio panaudojimas duomenų skaitymui



43 pav. Rezultatas po „pyspark.sql“ panaudojimo duomenų skaitymui

* Duomenų transformavimas: Kai duomenys perskaitomi į „DataFrame“ struktūra, įvairūs transformacijos veiksmai atliekami naudojant „DataFrame“ metodus. Trečiajame laboratoriniame darbe „drop()“ metodas (41 pav.) naudojamas nereikalingiems stulpeliams pašalinti. Metodas „withColumn“ naudojamas naujam stulpeliui pridėti ir stulpelio duomenų tipui konvertuoti (42 pav.). Rezultatus prieš „drop()“ funkcijos panaudojimą galima matyti (40 pav.) ir (42 pav.), o po - (43 pav.). Duomenis prieš „withColumn()“ panaudojimą galima matyti (43 pav.), o po – (45 pav.).



44 pav. „pyspark.sql“ modulio naudojimas nereikalingiems stulpeliams šalinti



45 pav. Duomenys prieš „drop()“ panaudojimą





46 pav. Duomenys po „drop()“ panaudojimo



47 pav. „pyspark.sql“ modulio „withColumn“ metodo naudojimas



48 pav. Duomenys po „withColumn()“ panaudojimo

* „DataFrames“ sujungimas: Kodas atlieka sujungimo operaciją tarp dviejų „DataFrames“, naudodamas sujungimo metodą. Trečiajame laboratoriniame darbe sujungiami duomenys pagal bendrą stuleplį („ID“) (46 pav.) ir sukuriamas naujas „DataFrame“, kuriame pateikiami sujungti duomenys. Duomenis prieš sujungimą galima pamatyti (47 pav.), o po – (48 pav.).



49 pav. „join()“ metodo panaudojimas



50 pav. Duomenys prieš „join()“ panaudojimą



51 pav. Duomenys po „join()“ panaudojimo

## Duomenų pertvarkymai

Duomenų pertvarkymas tarp skirtingų formatų ir duomenų struktūrų yra dažna duomenų apdorojimo ir analizės užduotis. Jis apima duomenų konvertavimą iš vieno pateikimo būdo į kitą, dažnai tam, kad būtų galima efektyviai apdoroti duomenis arba patenkinti tolesnių užduočių reikalavimus. Duomenims konvertuoti galima naudoti įvairius metodus ir priemones. Duomenų pertvarkymas paprastai apima tokias operacijas kaip duomenų analizė, filtravimas, agregavimas, atvaizdavimas ir restruktūrizavimas. Trečiajame laboratoriniame darbe buvo panaudoti keli duomenų pertvarkymų žingsniai:

* Duomenų įkėlimas: darbo kodas pradedamas duomenų įkėlimui iš tekstinio failo naudojant „spark.sparkContext.textFile()“ (49 pav.). Ši funkcija naudojamas failui nuskaityti ir sukurti „RDD“ rinkinį, kur kiekviena eilutė tampa „RDD“ elementu (50 pav.).



52 pav. Duomenų įkėlimo metu naudojami duomenų struktūros pakeitimai



pav. Duomenys po duomenų įkėlimo pertvarkymo

* Duomenų transformacija: funkcija „parsinam()“ kiekvieną „RDD“ eilutę suskaido į sąrašą, pašalindama tam tikrus simbolius, o „parsinam2()“ toliau kiekvieną eilutę suskaido į rakto ir vertės poras ir išgauna konkrečias raktų, trečiojo laboratorinio darbo atveju „marsrutas“, „sustojimo duomenys“, „Masinos tipas“ ir „siuntu skaicius“, reikšmes. Transformuoti duomenys grąžinami kaip „tuples“ duomenų struktūra (51 pav.).



54 pav. Duomenys po „parsinam()“ ir „parsinam2()“ funkcijų panaudojimo

* Duomenų konvertavimas: trečiojo laboratorinio darbo kodas konvertuoja „RDD“ rinkinį (52 pav.) į „Spark DataFrames“ duomenų struktūrą (53 pav.), kad būtų galima toliau apdoroti duomenis. Tai yra padaroma „RDD“ iškvietus „toDF()“ (54 pav.) ir nurodžius stulpelių pavadinimus,.



55 pav. „RDD“ rinkinys prieš konvertavimą naudojant „toDF()“ funkciją



56 pav. Duomenys po konvertavimo į „Spark DataFrames“



57 pav. Duomenų konvertavimas į „RDD“ į „Spark DataFrames“ duomenų struktūrą

* Duomenų vizualizavimas: laboratorinio darbo kodas konvertuoja kintamąjį „assembled\_data“ iš „pyspark DataFrame“ (55 pav.) duomenų struktūros į „Pandas DataFrame“ (56 pav.) naudojant „toPandas()“ funkciją (57 pav.).



58 pav. Duomenų stuktūra prieš „toPandas()“ naudojimą



59 pav. Duomenų struktūra po „toPandas()“ naudojimo



60 pav. Duomenų konvertavimas norint vizualizuoti duomenis

## Duomenų siuntimas

Vykdant kodą klasteryje, duomenų perdavimas tarp serverio mazgų arba lokalaus kompiuterio yra labai svarbus paskirstytųjų duomenų apdorojimo aspektas. Programoje „Spark“ duomenų perdavimas vyksta įvairiuose skaičiavimo etapuose. Pavyzdžiui, skaitant duomenis iš išorinių šaltinių, tokių kaip failai ar duomenų bazės, duomenis gali tekti perkelti iš saugyklos vietos į klasterio mazgus. Panašiai, atliekant transformacijas ar veiksmus su paskirstytųjų duomenų rinkiniais, tarp mazgų keičiamasi duomenimis, kad būtų lengviau atlikti lygiagretųjį apdorojimą. Atliekant tokias operacijas, kaip perskirstymas, agregavimas ir sujungimas, tarp mazgų gali būti perkeliama daug duomenų. Be to, renkant rezultatus ar perduodant duomenis į lokalųjį kompiuterį tolesnei analizei ar vizualizacijai, duomenys paprastai perkeliami iš paskirstytojo klasterio į vietinį kompiuterį. Trečiojo laboratorinio darbo atveju, šios kodo vietos būtų atsakingos už duomenų siuntimą tarp serverio mazgų arba lokalaus kompiuterio, jeigu kodas būtų vykdomas klasteryje:

* Tekstinio failo skaitymas: Paveikslėlyje () esančioje kodo eilutėje skaitomas tekstinis failas iš nurodytos vietos, todėl failas gali būti perkeliamas iš paskirstytos failų sistemos arba lokalios failų sistemos į „Spark“ klasterį.