Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jaka Kokošar

Interaktivna podatkovna analitika v oblaku s programsko knjižnico Resolwe in programom Orange

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Blaž Zupan
Somentor: asist. dr. Miha Štajdohar

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Raziskovalni pristop k analizi podatkov zahteva orodja za interaktivno podatkovno analitiko in interaktivno vizualizacijo podatkov. Obstoječa orodja na tem področju dobro delujejo na manjših naborih podatkov in analitiko izvajajo lokalno, na osebnih računalnikih. Za večje podatke bi bilo potrebno razviti strežniške rešitve, ki podpirajo interaktivnost in interaktivne vmesnike na odjemalcih. V diplomski nalogi razvijte prototip take rešitve, ki naj temelji na sistemu za strežniško izvajanje procesov Resolwe in tega vključi v program za interaktivno podatkovno analitiko Orange.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Pregled področja **5**

2.1 Programski sistem za analitiko podatkov KNIME 6

2.2 Program za podatkovno analitiko Orange 7

2.3 Orodje za razvijanje strežniških analiz Snakemake 8

2.4 Gradnja delovnih procesov v okolju Resolwe 10

3 Arhitektura sistema **13**

3.1 Programske zahteve 13

3.2 Arhitektura programskega sistema 14

4 Programske rešitve **17**

4.1 Okolje Resolwe 17

4.2 Izvajanje procesov Resolwe v okolju Orange 22

5 Primeri uporabe **25**

6 Zaključek **29**

Literatura **31**

Povzetek

Naslov: Interaktivna podatkovna analitika v oblaku s programsko knjižnico Resolwe in programom Orange

Avtor: Jaka Kokošar

Na področju podatkovne analitike se v zadnjem obdobju soočamo z vse večjo količino podatkov, ki jih z novimi pristopi in metodami pridobivajo prav na vseh področjih človekovega udejstvovanja. Če se orodja za podatkovno analitiko želijo soočiti s tem problemom, se morajo temu primerno prilagoditi. V diplomskem delu predlagamo arhitekturo programskega sistema, ki omogoča interaktivno izvajanje analitičnih orodij v oblaku. Predlagana rešitev združuje dva odprtokodna projekta. Odprtokodni program Orange nudi intuitiven grafični vmesnik, kjer lahko s pomočjo vizualnega programiranja povezujemo različna analitična orodja. Odprtokodni projekt Resolwe pa omogoča učinkovito izvajanje analitičnih procesov v oblaku in nudi ogrodja za prenos podatkov in rezultatov na in iz strežnika. V delu predstavimo prototip predlagane programske rešitve, s katero pokažemo, da Resolwe lahko poganja podatkovno analitiko programa Orange. V delu opozorimo na omejitve kombinacije teh dveh sistemov in predlagamo rešitve, ki bi jih odpravile.

Ključne besede: analiza podatkov, vizualno programiranje, analitični procesi v oblaku, veliki podatki.

Abstract

Title: Interactive data analytics in the cloud through integration of Resolwe library and Orange data mining

Author: Jaka Kokošar

With new approaches and the development of new methods in all areas of human endeavour we are facing an increasing amount of obtained data. The field of data analytics must adjust accordingly. In the Thesis, we propose a new architecture that allows interactive execution of data analytics in the cloud. The proposed solution combines two open source projects. Orange, the open source data mining suit, offers an intuitive graphical interface to create data analysis workflows. Resolwe is an open source dataflow package that fully supports execution of workflows in the cloud. We developed a prototype that combines the two systems and demonstrate that Resolwe is a viable solution for running Orange's data analysis workflows in the cloud. We point to the current limitations of proposed architecture and propose possible solutions that would eliminate them in the future.

Keywords: data analytics, visual programming, analytical processes in the cloud, big data.

Poglavje 1

Uvod

Z vse večjim tehnološkim napredkom ter razvojem vse bolj učinkovitih metod in načinov pridobivanja podatkov se na vseh področjih znanosti, industrije in življenja srečujemo z vse večjo količino pridobljenih podatkov. Programska oprema za podatkovno analitiko je že dosegla neko mero zrelosti [11], vendar se tudi na tem področju soočamo s problemom obdelave velikih podatkov [7]. Četudi so osebni računalniki z zmogljivo strojno opremo vse bolj dostopni, zaradi omejenega delovnega spomina in procesorske moči niso primerni, da se soočijo z omenjeno problematiko.

Izvajanje zahtevnih analiz nad velikimi podatki zahteva uporabo strežniške strojne in programske opreme za vzporedno in porazdeljeno procesiranje. KNIME[1], eno izmed vodilnih orodij za podatkovno analitiko, ki sledi paradigmi vizualnega programiranja in je po principu precej podobnen programu Orange[2], ki smo ga uporabili v naši rešitvi. KNIME že ponuja rešitve, ki končnega uporabnika osvobodi poganjanja računsko zahtevnih operacij na osebnih računalnikih. Ob ustrezni namestitvi potrebnih razširitev je uporabniku programa omogočen prenos in izvajanje analiz na namenskih strežnikih. S strežniško razširitvijo program KNIME naslavljaja predvsem uporabnike v poslovnem okolju. Uporabnikom poenostavi skupinsko sodelovanje, uporabo

[1] `https://www.knime.com`

[2] `https://orange.biolab.si`

orodij za avtomatizacijo izvajanja in poročanje rezultatov analiz. Te analize na strežniku izvršijo vse procese danega postopka, od včitavanja podatkov, predobdelave, izbora podatkov do modeliranja. Postopki se v celoti izvršijo brez posredovanja uporabnika in brez vključevanja njegovih morebitnih interakcij.

Vključitev interakcije v postopek analize in modeliranje podatkov je ključnega pomena za razumevanje podatkov in raziskovalni tip analize [10]. Uporabnik naj bi tekom analize podatke lahko vizualiziral, poiskal zanj pomembne skupine podatkov, te potem razvrstil ali poiskal ustrezne modele in vpliv posameznih faktorjev preučeval interaktivno, skladno z različnimi kriteriji izbora podatkov. Interaktivni izbor podatkov bi moral prožiti ustrezne procese, ki sledijo v analitičnem delokrogu, ti pa bi po svoji izvršitvi morali osvežiti vizualne prikaze rezultatov. Tak, interkativni način dela s podatki, podpira programski sistem za analizo podatkov Orange, a je pri tem omejen na poganjanje analize na osebnem računalniku. V diplomski nalogi smo skušali raziskati način, s katerim Orange spremenimo tako, da interaktivna uporaba njegovega programskega vmesnika proži analitične procese na strežniku. Prototip takega analitičnega sistema smo razvili z uporabo knjižnice Resolwe, ki podpira izvrševanje analitičnih procesov na strežnikih in kontrolo procesa izvrševanja analitičnih korakov s strani odjemalca.

Menimo, da je uporabnost takih analitičnih sistemov pogojena z enostavnostjo njihove uporabe. Programu KNIME s strežniško razširitvijo dodamo nov nivo kompleksnosti. Uporabnik programa se mora poleg lokalnega zavedati še strežniškega okolja in poznati vse potrebne konfiguracije, da lahko poganja delovne procese. Naša vizija je ustvariti orodje, ki omogoči reševanje najtežjih problemov na podorčju podatkovne analitike, pri tem pa orodje ne predstavlja dodaten izziv uporabniku pri njegovi uporabi. Uporaba kompleksnih strežniških sistemov je v našem primeru v celoti izpostavljena prek grafičnega vmesnika Orange, s katerim ohranimo enostavnost gradnje analitičnih delotokov.

V nadaljevanju dela najprej bolj podrobno predstavimo nekaj sorodnih

rešitev in tehnologijo Resolwe. Sledi podrobnejši opis arhitekture predlaganega sistema in predstavitev programske rešitve, ki jo umesti v okolje programa Orange. Navedemo tudi nekaj primerov uporabe razvitega sistema na večjih podatkih in ocenimo odzivnost sistema ter možnost povzporedenja podatkovne analitike na strežniku. Zaključimo z oceno izvedljivosti projekta, kjer bi programski sistem Orange v celoti izvajal analitiko podatkov na strežniku. Predlagamo tudi ideje za možne izboljšave sistema in dodatke, ki naj bi bili implementirani v knjižnici Resolwe.

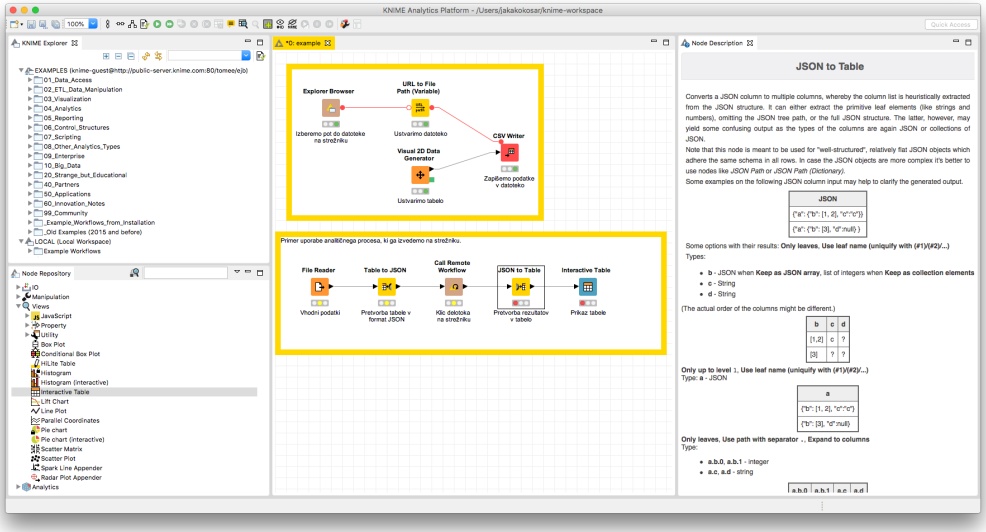
Poglavje 2

Pregled področja

V tem poglavju se osredotočimo na rešitve, ki so sorodne predlagani arhitekturi. Najprej si pogledamo program KNIME, ki je eno izmed vodilnih orodij za podatkovno rudarjenje in tudi neposredna konkurenca programu Orange. Oba omenjena programa združuje pristop k vizualnemu programiranju s katerim gradimo toke procesov, ki se izvajajo nad vhodnimi podatki. Gre za odprtokodni platformi, ki ponujata širok spekter orodij za delo nad podatki. Uporabnik lahko zajame in obdela podatke ter jih analizira in vizualizira, za kar ne potrebuje znanja programskih jezikov. V nadaljevanju teh orodij ne bomo primerjali, saj nas zanima predvsem njuna rešitev podatkovne analitike v oblaku. Na hitro se dotaknemo tudi že zelo uveljavljenega orodja Snakemake, ki rešuje probleme v isti domeni kot orodje Resolwe. Omenimo prednost predlaganega sistema, ko gre za integracijo v sistem grafičnega okolja Orange in ga podrobneje predstavimo.

2.1 Programski sistem za analitiko podatkov KNIME

KNIME [2] (*Konstanz Information Miner*) je odprtokodno okolje napisano v Javi. Osrednji del programskega sistema predstavlja platforma za podatkovno analitiko (*KNIME Analytics Platform*[1]), ki omogoča ustvarjanje delovnih procesov, izvajanje izbranih analiz ter vpogled v njihove rezultate.



Slika 2.1: Grafični vmesnik programa KNIME.

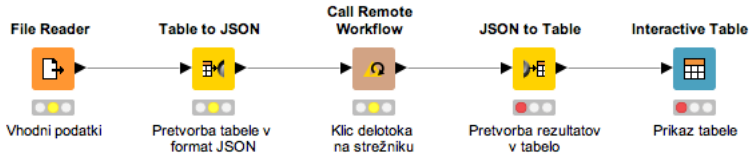
KNIME Server[2] je licenčna programska oprema, ki z uporabo spletnih tehnologij razširi osnovno delovanje platforme. Tri njene glavne značilnosti so:

- Poleg lokalnega delovnega prostora ima uporabnik na voljo svoj strežniški prostor na katerem lahko hrani podatke in delovne procese. Sodelovanje med uporabniki na projektih je tako zelo poenostavljeno, saj lahko

[1] `https://www.knime.com/knime-software/knime-analytics-platform`
[2] `https://www.knime.com/knime-software/knime-server`

z nastavljanjem pravic enostavno izpostavimo podatke in delovne procese drugim uporabnikom, ki jih lahko prenesejo v svoje okolje.

- Za izvajanje procesov skrbi prilagojena oblika analitične platforme, ki je nameščena brez gradnikov uporabniškega vmesnika. Tako podatki kot delovni procesi se morajo pred izvedbo prenesti na strežniški delovni prostor.



Slika 2.2: Klic strežniškega procesa iz odjemalčevega delotoka.

- Delovne procese lahko poganjamo tudi prek spletnega brskalnika z uporabo spletnega portala *KNIME WebPortal* in pa spletnih vmesnikov REST ali SOAP. Tako lahko delovne procese po potebi integriramo v druge programske sisteme.

2.2 Program za podatkovno analitiko Orange

Orange [3] je odprtokodni projekt, ki je nastal, na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Ponaša se s preprostim grafičnim vmesnikom prek katerega lahko združujemo orodja za vizualizacijo, podatkovno rudarjenje ter strojno učenje. Gradniki so osnovna komponenta s katerimi gradimo kompleskne analitične postopke. Z njimi lahko opravljamo tako preprosta opravila kot so branje in prikaz datotek, ter izvajanje kompleksnih algoritmov strojnega učenja in podatkovne analitike. Z interaktivnim vizualnim programiranjem gradnike prek njihovih vhodov in izhodov povezujemo v delotoke in spremljamo posamezne korake analize. Orange je z svojo preprostostjo in interaktivnostjo primeren tako za začetnike kot tudi bolj izkušene uporabnike.

Slika 2.3: Posnetek zaslona grafičnega vmesnika Orange s preprostim delotokom, ki prebere in prikaže podatke.

2.3 Orodje za razvijanje strežniških analiz Snakemake

Snakemake [9] je orodje za kompleksne analie na področju bioinformatike. Zagotavlja ponovljive analize ter omogoča porazdeljeno izvajanje le-teh in tako skuša čim bolj učinkovito izkoristiti računske vire, ki so nam na voljo. Z uporabo domensko specifičnega jezika, ki je podoben programskemu jeziku Python, določimo delovni proces v datoteki *Snakefile*. Delovni proces je sestavljen iz množice pravil, ki določijo, kako ustvariti izhodne podatke iz podatkov na vhodu. Oblika pravila predvideva določeno ime, vhodne in izhodne podatke ter ukaz ki se bo izvršil nad vhodnimi podatki. To je lahko operacija ukazne lupine ali pa koda v programskem jeziku Python.

```
import numpy as np
from Orange.data import Table
from MulticoreTSNE import MulticoreTSNE

rule all:
  input: "embedding.tab"

rule simple_filter:
 input:
   data = "data.tab"
 output:
   data = "data.filter.tab"
 run:
  table = Table(input.data)[:50]
  table.save(output.data)

rule tsne:
 input:
   data = "data.filter.tab"
 output:
   embedding = "embedding.tab"
 threads: 2
 run:
  t_sne = MulticoreTSNE(n_jobs=threads)
  table = Table(input.data)
  embedding = t_sne.fit_transform(table.X).astype(np.float32)
  np.savetxt(output.embedding, embedding, delimiter='\t')
```

Primer 2.1: Primer zapisa pravil v datoteki *Snakefile* v kombinaciji s programskim jezikom Python. Prikazana analiza zaporedno izvede pravila in vrne rezultat metode t-SNE.

Na primeru 2.1 lahko vidimo preprost primer uporabe orodja Snakemake v kombinaciji s programsko knjižnico Orange. Snakemake je orodje, ki se izvaja v ukazni lupini. Z ukazom `snakemake` poženemo pravilo `all`, ki kliče preostala pravila v pravilnem vrstnem redu. Lahko vidimo, da pravilo `tsne` pričakuje na vhodu datoteko, ki jo vrne pravilo `simple_filter`, zato ima v tem primeru prednost.

Ukaz `snakemake` lahko poženemo na osebnem računalniku ali pa na bolj kompleksni strežniški infrastrukturi brez dodatnih konfiguracij datoteke *Snakefile*. Žal pa sistem ne podpira interaktivno rabo, kjer bi lahko uporabnik na vsakemu koraku delovnega procesa spreminjal parametre in imel vpogled v vmesne rezultate. Ti koraki so nujni, če želimo ohraniti idejo interaktivnih delovnih procesov, kot jih uporablja Orange.

2.4 Gradnja delovnih procesov v okolju Resolwe

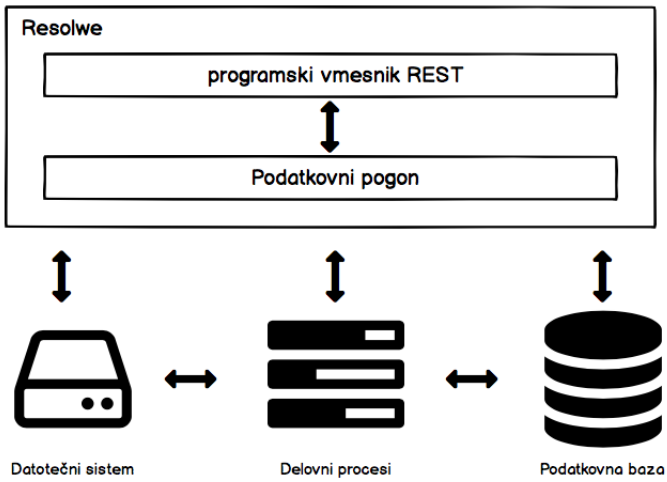
Resolwe [5] je odprtokodni projekt, ki ga ustvarja podjetje Genialis[3]. Gre za orodje, ki na učinkovit način omogoči gradnjo kompleksnih delovnih procesov za izvajanje zahtevnih podatkovnih analiz. V principu gre za knjižnico v programskem jeziku Python oz. aplikacijo spletnega ogrodja Django[4], ki jo lahko enostavno vključimo v projekt Django in tako omogočimo izvajanje delovnih procesov prek spletnega vmesnika REST. Prek spletnega vmesnika poteka vsa komunikacija s podatkovnim pogonom, ki skrbi za izvajanje delovnih procesov. Resolwe zna odpravljati odvisnosti med procesi in jih razporejati na gruče strežnikov ter tako izkoristiti računske vire, ki so nam na voljo.

[3] `https://www.genialis.com/`

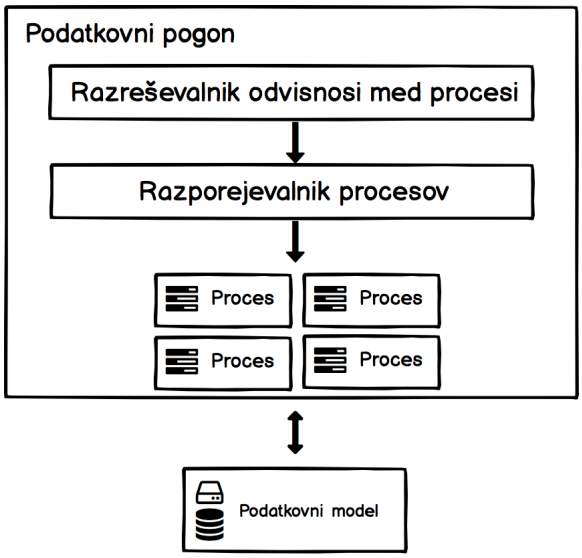
[4] `https://www.djangoproject.com`

Najbolj osnovna gradnika v arhitekturi sistema sta podatkovni objekt in proces. Podatkovni objekt opisuje podatke, izhodne statuse in rezultate procesov, ki so se izvedli nad vhodnimi podatki. Proces opisuje stanje podatkovnega modela med izvajanjem. Poleg programske kode, ki se bo izvedla nad podatki, hrani tudi različne metapodatke s katerimi razporejevalnik razrešuje odvisnosti med procesi.

Resolwe z modularnimi in prilagodljivimi deli sistema (slika 2.5) omogoča enostavno uporabo na osebnem računalniku in namestitev na kompleksne strežniške sisteme. Ne glede na to, kako je sistem postavljen, do njega dostopamo prek spletnega programskega vmesnika (primer 2.2). To nam omogoči lažji razvoj in integracijo sistema v programsko okolje Orange.



Slika 2.4: Arhitektura sistema Resolwe. Povzeto po [5].



Slika 2.5: Shema podatkovnega pogona. Povzeto po [5].

```
1  import resdk
2  username = 'admin'
3  password = 'secret'
4  url = 'http://127.0.0.1:8000/'
5
6  res = resdk.Resolwe(username, password, url)
7  proc = res.run('upload-file', input={'src': 'table.tab'})
8
9  res.data.all()
10 [Data <id: 43, slug: 'tabletab', name: 'table.tab'>]
11
12 res.data.get(id=43).download()
```

Primer 2.2: Primer uporabe orodja Resolwe. V danem primeru se prijavimo na strežnik, poženemo proces, ki naloži podatke na strežnik in izpišemo podatke, ki se nahajajo na strežniku.

Poglavje 3

Arhitektura sistema

Po pregledu obstoječih rešitev in tehnologij, ki so nam na voljo in so bile opisane v drugem poglavju (poglavje 2), tu navedemo zahteve predlagane arhitekture in jo bolj podrobno predstavimo.

3.1 Programske zahteve

Programsko okolje Orange [4] nam omogoča enostaven razvoj novih analitičnih gradnikov z uporabo programskega jezika Python. V le nekaj dneh je mogoče ustvariti prototip gradnika in ga testirati s preostalimi gradniki programa Orange, ki skupaj pokrivajo širok spekter področja podatkovne analitike. Program Orange je zanimiv tudi zato, ker omogoča interaktivno izvajanje analiz in vpogled v vmesne rezultate postopkov. Ker želimo ohraniti omenjene lastnosti programa Orange, smo si s prototipno rešitvijo, želeli odgovoriti na sledeča vprašanja:

- Ali lahko z uporabo orodja Resolwe poganjamo delovne procese značilne za program Orange? Kje se nahajajo podatki in kako z njimi upravljamo?
- Ali je mogoče spremljati napredek procesa, ki se je zagnal na strežniku? Povedano drugače, ali lahko, na zahtevo, dobimo vmesne rezultate in status zagnanega procesa?

- Ali lahko s predlagano arhitekturo poenostavimo razvoj gradnikov v programu Orange? Kakšen je minimalen vzorec kode, ki sproži analizo na strežniku ter prikaže rezultate in ali lahko to posplošimo na vse gradnike programa Orange?

- Ali se bo z izvajanjem procesov na strežniški aplikaciji izboljšala ali poslabšala zmogljivost gradnikov programa Orange? V kolikšni meri lahko pohitrimo računske operacije nad podatki? Je mogoče ohraniti odzivnost uporabniškega vmesnika med izvajanjem analiz?

- Ali lahko predlagano arhitekturo zapakiramo v instalacijski paket programa Orange in zaženemo strežniško aplikacijo lokalno, na uporabnikovem računalniku?

Na osnovi zgornjih vprašanj smo oblikovali sledeče programske zahteve:

- Podatki in analitična orodja se nahajajo na strežniški aplikaciji. Aplikacija na odjemalcu lahko vidi seznam podatkov in procesov, ki so ji na voljo.

- Gradniki programa Orange postanejo kontrolne enote za izvajanje procesov na strežniku. V vsakem trenutku lahko pridobimo informacijo o stanju izvajanja izbranega procesa in pri tem ohranimu odzivnost uporabniškega vmesnika.

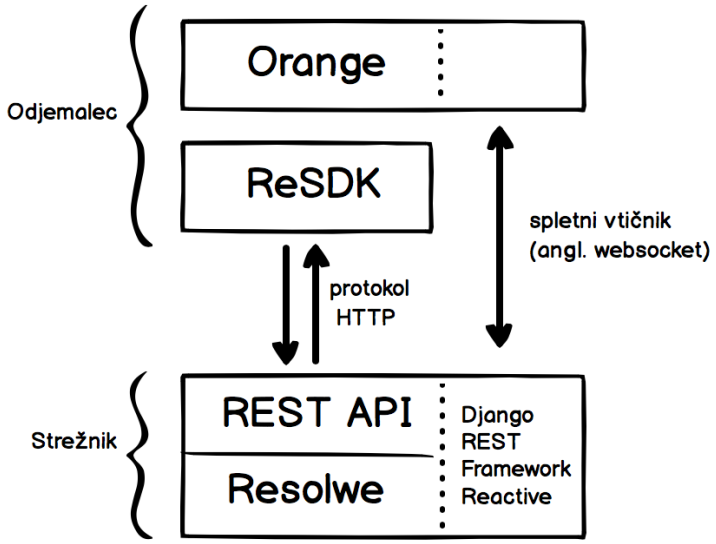
- Procese na strežniku lahko poganjamo v večjdernem okolju.

- Strežniško aplikacijo namestimo in poženemo na strani odjemalca.

3.2 Arhitektura programskega sistema

Grafični vmesnik programa Orange je zgrajen okoli pythonske knjižnice PyQt. Ta omogoča uporabo orodja Qt za razvoj grafičnih vmesnikov v programskem jeziku Python. Gradnike programa Orange lahko razumemo kot ovojnico čez

algoritme za podatkovno analitiko. Podatki, nad katerimi gradniki tipično izvajajo operacije, se shranjujejo v obliki podatkovne tabele imenovane Orange Data Table.



Slika 3.1: Integracija sistema Resolwe v okolje Orange. Funkcionalnosti sistema Resolwe so izpostavljene prek spletnega vmesnika REST, ki ga izrabljamo prek orodja ReSDK na strani odjemalca. Komunikacija med odjemalcem in strežnikom poteka prek protokola HTTP. Spletni vtičniki so komunikacijski kanal med gradnikom in procesom.

V predlagani arhitekturi sistema (slika 3.1) programu Orange odvzamemo orodja za podatkovno analitiko in jih umestimo v okolje sistema Resolwe. Gradniki grafičnega vmesnika Orange niso več ovojnice nad algoritmi za podatkovno analitko, temveč postanejo kontrolne enote za izvajanje analitike na strežniku. Glavni modul programa Orange postane orodje Python ReSDK[1]. Gre za knjižnico, ki omogoči interakcijo z spletnim vmesnikom strežnika Resolwe, v programskem jeziku Python. Prek nje se lahko prijavimo na strežnik,

[1] `https://resdk.readthedocs.io/en/stable/`

dostopamo do podatkov, izvajamo procese in gradimo kompleksne analize.

Osnovni podatkovni tip, nad katerim gradniki izvajajo operacije, postane podatkovni objekt in tako zamenja podatkovno tabelo Orange. Podatkovni objekt je vse, kar prenesemo ali pa ustvarimo s procesi na strežniku. V njem se hranijo zapisi o vhodnih in izhodnih parametrih ter informacije o procesu, ki ga je ustvaril. Poleg tega v podatkovnih objektih hranimo dodatne podatke, ki opisujejo objekt in omogočajo ponovljivost analiz. Ko v gradniku programa Orange poženemo proces, se med gradnikom in spletnim strežnikom vzpostavi povezava prek spletnega vtičnika (angl. *websocket*). Gradnik tako sprejme vsako spremembo, ki se zgodi nad podatkovnim objektom med izvajanjem procesa.

Predlagana arhitektura z uporabo orodja Resolwe in programom Orange naslavlja nekatere ključne arhitekturne koncepte sodobnih sistemov na področju podatkovne analitike [12]. S takim pristopom ohranimo koncept načrtovanja delovnih procesov prek vizualnega programiranja. Omogočamo interaktivno rabo vizualizacij in prikaz vmesnih rezultatov ter stanj analiz v izvajanju, prek grafičnega vmesnika Orange. Resolwe, kot podatkovni pogon predlagane arhitekture, omogoči razpolaganje z računskimi viri strežnikov. Z možnostjo povzporednega izvajanja algoritmov, lahko bistveno zmanjšamo čas izvajanja podatkovnih analiz.

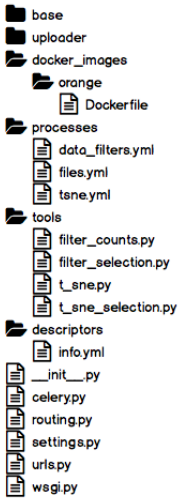
Poglavje 4

Programske rešitve

V tem poglavju se osredotočimo na programske rešitve, ki smo jih uporabili pri implementaciji naloge. Najprej si pogledamo, kako smo uporabili tehnologijo Resolwe. Pokažemo, kako določimo procese in pripravimo izvajalno okolje analitičnih orodij. V drugem delu pa pogledamo kako smo vse to vključili v programsko okolje Orange. Ker je Resolwe projekt, ki se še razvija in ker namen naloge ni podrobneje predstaviti notranje strukture sistema, se bomo osredotočili le na tiste dele, ki so potrebni, da lahko upravljamo s procesi in podatkovnimi objekti v programu Orange.

4.1 Okolje Resolwe

Struktura tipičnega projekta v sistemu Resolwe je prikazana na sliki 4.1. Tu nas zanimajo predvsem direktoriji, nad katerimi delamo v tipičnem razvojnem ciklu gradnika v okolju Orange. V njih hranimo orodja za delo nad procesi v okolju Resolwe. To so direktoriji `processes`, `tools` in `docker_images`.



Slika 4.1: Struktura direktorija projekta Resolwe.

4.1.1 Shema procesa

Priporočen način določitve procesa na strežniku Resolwe uporablja zapis v načinu YAML[1]. Poglejmo si primer procesa (primer 4.1), ki smo ga uporabili pri implementaciji naloge. Gre za proces s katerim prenesemo datoteko na strežnik Resolwe.

Polje imenovano `slug` predstavlja unikaten identifikator procesa. V polju `requirements` procesu nastavimo okolje v katerem se bodo izvajala orodja, ki jih potrebujemo za podatkovno analitko (več v podpoglavju 4.1.2). V nadaljevanju določimo nekaj meta podatkov procesa, kot sta ime in tip podatkovnega objekta, ki ga proces ustvari. Polja `input` in `output` določata vhodne in izhodne parametre procesa. Parametru moramo nastaviti ime in tip, lahko mu nastavimo tudi druga podprta polja (kot naprimer polje, ki parametru nastavi privzeto vrednost). Program, ki se bo izvedel ob zagonu procesa pa določimo v polju `run`. V času nastajanja tega dela je edini način pisanja programov v sistemu Resolwe podprt prek skriptnega jezika Bash.

[1] `http://yaml.org`

Ekipa, ki dela na projektu Resolwe, si prizadeva, da bi pisanje procesov bilo v celoti podprto v programskem jeziku Python.

```
- slug: data-table-upload
  name: Orange Data Table upload
  requirements:
    executor:
      docker:
        image: jakakokosar/resolwe_project:latest
  data_name: '{{src.file|default("?")}}'
  version: 0.0.1
  type: data:table:singlecell
  category: upload
  persistence: RAW
  description: Upload pickled Orange data table.
  input:
    - name: src
      label: Orange Data Table
      type: basic:file
  output:
    - name: table
      label: Orange Data Table
      type: basic:file
  run:
    runtime: polyglot
    language: bash
    program: |
      mv {{ src.file_temp }} {{ src.file }}
      re-save-file table {{ src.file }}
```

Primer 4.1: Primer datoteke, ki določa proces na strežniku Resolwe. V shemi navedemo izvajalno okolje, vhodne in izhodne podatke, ter programsko kodo, ki ustvari podatkovni objekt na strežniku.

4.1.2 Izvajalnik programske kode

Ko govorimo o procesih moramo vedeti, kje in kako se bo naša programska koda izvajala. Zaganjalnik procesa najprej pripravi izvajalno okolje v vsebniku Docker in šele nato izvede kodo procesa. Vsebniška slika vsebuje vsa orodja, ki bodo na voljo med izvajanjem procesa. V datoteki Dockerfile določimo zaporedje ukazov, ki kreirajo vsebniško sliko.

```
FROM docker.io/resolwe/base:ubuntu-18.04
RUN pip install numpy, scipy, orange3
```

Primer 4.2: Primer uporabe datoteke Dockerfile, kjer določimo vsa orodja, ki jih bo imel proces na voljo med izvajanjem.

Pomembno je, da izhajamo iz osnovne slike, ki nam jo ponuja Resolwe. Tako bomo imeli med izvajanjem procesa na voljo orodja, ki poenostavijo pisanje programov. V primeru 4.1 smo uporabili ukaz `re-save-file`, ki shrani datoteko na izhod procesa.

```
requirements:
  executor:
    docker:
      image: jakakokosar/resolwe_project:latest
  resources:
     cores: 4
     memory: 4096
```

Primer 4.3: V polju `requirements` določimo nastavitve, ki jih bo zaganjalnik procesa uporabil pri pripravi izvajalnega okolja procesa.

Vsakemu procesu lahko tudi dodelimo računske vire, ki mu bodo med izvajanjem na voljo. V primeru 4.3 smo proces nadgradili tako, da smo določili podpolje `resources`.

Pisanje procesov si lahko poenostavimo tako, da kličemo izvršljive datoteke, ki smo jih vstavili v direktorij `tools`. Glede na to, da program Orange razvijamo v programskem jeziku Python, je smiselno procese na strežniški

aplikaciji razvijati v istem programskem jeziku. Na primeru 4.4 lahko vidimo primer klica datoteke `t_sne.py`, v kateri nad vhodnimi podatki izvedemo metodo t-SNE (*t-distributed stochastic neighbor embedding*) in njen rezultat shranimo v obliki formata JSON[2]. Ker implementacija omenjene metode omogoča večjedrno izvajanje, ji prek vhodnega podatka (*requirements.resources.cores*) navedemo število procesorskih jeder, ki so na voljo procesu (primer 4.4).

```
1  run:
2    runtime: polyglot
3    language: bash
4    program: |
5      t_sne.py {{ data_table.table.file }} \
6               {{ pca_components }} \
7               {{ perplexity }} \
8               {{ iterations }} \
9               {{ init.embedding_json.embedding }} \
10              {{ requirements.resources.cores }}\
11              embedding.json
12     re-save embedding_json embedding.json
```

Primer 4.4: Vhodne podatke procesa uporabimo za klic metode t-SNE v procesu Resolwe, ki se požene iz ukazne lupine. Izračun metode shranimo na izhodno polje (`embedding_json`) procesa.

Seveda pa pri tem nismo omejeni le na programski jezik Python. Znotraj procesa lahko kličemo orodja v poljubnih programskih jezikih v kolikor nam to omogoča izvajalno okolje procesa. V našem primeru je vsebnik slike Docker, ki smo jo pripravili v primeru 3.1.

[2]https://json.org

4.2 Izvajanje procesov Resolwe v okolju Orange

Kot smo že omenili v poglavju 3.2, poteka vsa komunikacija med Orange in Resolwe prek Python knjižnice ReSDK. Da bi razvoj gradnikov poenostavili, smo ustvarili pomožni modul, ki služi kot ovojnica nad knjižnico ReSDK. Na ta način lahko zagotovimo konsistentno rabo ReSDK med vsemi gradniki v okolju Orange. Ker Resolwe ni del projekta Orange, se ne ta način zavarujemo pred nenadnimi spremembami na projektu Resolwe in se tako s primernim vzdrževanjem pomožnega modula izognemo večjim posegom v programsko kodo gradnikov.

```
from orangecontrib.resolwe.utils import ResolweHelper
res = ResolweHelper()
res.username
'admin'
 res.url
'http://127.0.0.1:8000/'
```

Primer 4.5: Prijava na strežnik Resolwe iz programskega okolja Orange.

Seznam datotek, ki se nahajo na strežniku Resolwe in so dostopne prijavlenemu uporabniku, lahko vidimo s klicom metode `list_objects` (primer 4.6). Datoteke lahko filtriramo, izbrišemo ali pa jih prenesemo iz strežnika, za lokalno rabo.

```
res.list_objects()
[Data <id: 1, slug: 'aml-8kpickle', name:'aml-8k.pickle'>,  Data <id: 2,
    slug: 'aml-1kpickle', name:'aml-1k.pickle'>]
```

Primer 4.6: Prikaz datotek, ki se nahajajo na strežniku Resolwe.

V primeru 4.7 lahko vidimo, da do pocesov lahko dostopamo prek metode `list_process` (primer 4.7). Proces, ki smo ga definirali v primeru 4.1, lahko poženemo prek pomožne metode `upload_table`, ki preveri vhodne po-

datke in jih pripravi za delo s procesi na strežniku. Ko smo podatke naložili na strežnik, nad njimi izvedemo proces metode t-SNE. Vhodni parameter procesa je podatkovni objekt, ki se je ustvatil, ko smo podatke naložili na strežnik. Rezultat procesa t-SNE je ustvaril nov podatkovni objekt, prek katerega dostopamo do rezultatov metode.

```
from Orange.data import Table
res.list_process()
[Process <id: 1, slug: 't-sne', name: 't-SNE'>, Process <id: 2 slug: 'data
    -table-upload', name: 'upload'>]

res.upload_table(Table('iris'))
res.list_objects()
[Data <id: 1, slug: 'aml-8kpickle', name:'aml-8k.pickle'>, Data <id: 2,
    slug: 'aml-1kpickle', name:'aml-1k.pickle'>, Data <id: 3, slug: '
    irispickle', name: 'iris.pickle'>]

selected_data = res.get_object(id=1)
inputs = {'data_table': selected_data, 'pca_components': 20 }

proc = res.run_process('t-sne', inputs)
Data <id: 3, slug: 't-sne-results', name: 't-sne results'>
result = res.get_json(proc, 'embedding json')
```

Primer 4.7: Delo s procesi na strežniku Resolwe. V prikazanem delotoku, naložimo datoteko na strežnik in nad njo poženemo proces t-SNE.

Uporaba modula *orangecontrib.resolwe.utils* je opcijska. Navedene primere lahko ponovimo z uporabo python knjižnice ReSDK. Vendar poleg že zgoraj omenjene prednosti (lažje vzdrževanje gradnikov) bomo z uporabo modula reševali tudi druge tipične probleme, s katerimi se srečujemo v razvojnem ciklu gradnika v okolju Orange. Vzpostavitev povezave prek spletnih vtičnikov in uporaba procesorskih niti sta vzorca, ki se pojavita pri vseh gradnikih. Zato je smiselno omenjene funkcionalnosti podpreti v pomožnem modulu, saj lahko s pravim pristopom zmanjšamo kompleksnost programske kode posameznih gradnikov.

Na primeru 4.8 si poglejmo uporabo orodja ReSDK v gradniku t-SNE. Klic funkcije `compute_tsne_embedding` sproži izvajanje metode, ki vrne rezultat v obliki, ki ga pričakuje metoda za izris vizualizacije. Če želimo omenjeno metodo pognati na strežniku Resolwe, moramo klic funkcije zamenjati s funkcijo, ki pokliče proces na strežniku. Proces vrne rezultat v pričakovani obliki, do njega dostopamo prek izhodnega parametra procesa, ki smo ga izvedli.

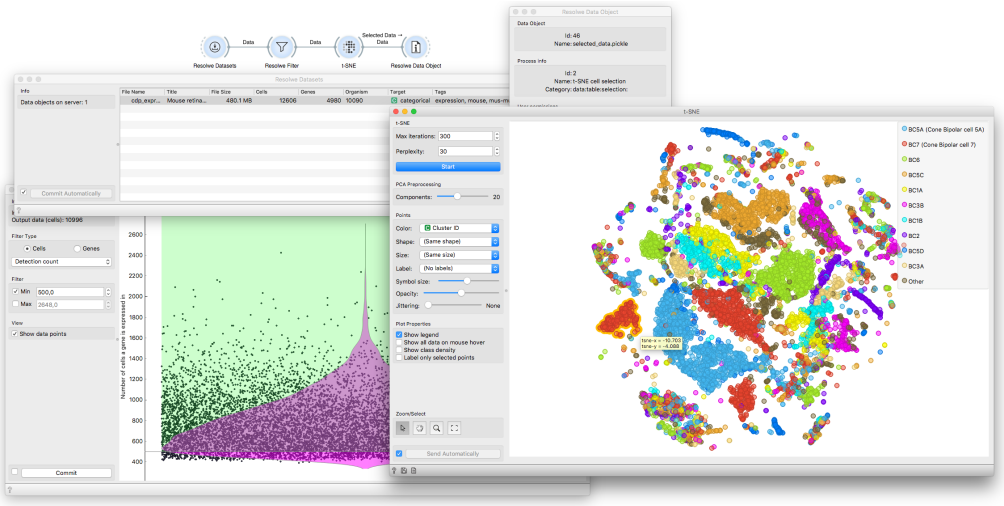
```
1 embedding = compute_tsne_embedding(inputs)
2 ------------ ali ---------------------------------------
3 proc = res.run_process('t-sne', inputs)
4 embedding = res.get_json(proc, 'embedding_json')
```

Primer 4.8: Primerjava klica metode t-SNE med trenutno in predlagano implementacijo gradnika t-SNE.

Poglavje 5

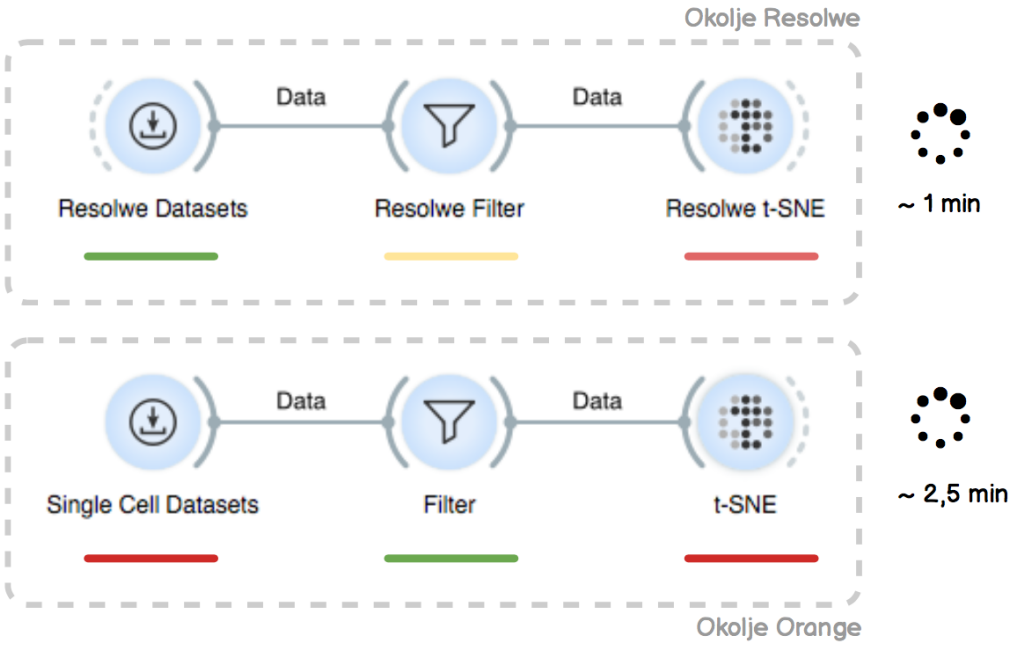
Primeri uporabe

Smiselnost uporabe predlagane arhitekture lahko ocenimo z implementacijo delovnih procesov, ki se tipično pojavijo v okolju Orange. V diplomskem delu smo razvili prototip dodatka za Orange, ki v celoti izvaja procese na strežniku Resolwe in uporablja ter obdeluje podatke s področja genomike posameznih celic [8]. Za pričujoče delo je tu pomembno le, da so to visokorazsežn podatki z velikim številom primerov.



Slika 5.1: Delovni proces v programu Orange z uporabo tehnologije Resolwe.

Tipičen proces v programu Orange (slika 5.1) lahko razdelimo na tri ključne korake: zajem, preobdelava in analiza vhodnih podatkov. Uporabili smo podatke, ki so tipični za analizo visokodimenzionalnih podatkov s področja funkcijske genomike [1]. V podatkih, ki jih uporabljamo v primeru je zbranih 13.000 celic, katerih odziv je opisan z izraženostjo več kot 5.000 genov [6]. Podatki na tem področju, so lahko še bistveno večji kot podatki, ki smo jih uporabili v zgornjem primeru. Problem s katerim se soočamo pri izvajanju tovrstnih analiz v programu Orange lahko ponazorimo že z uporabo razmeroma majhnega vzorca podatkov.



Slika 5.2: Uporabljene analitične sheme in časovna primerjava podatkovne analize med obstoječo in predlagano arhitekturo. V uporabljeni shemi najprej preberemo podatke in s pomočjo gradnika `Filter` izberemo celice, katerih število izraženih genov presega izbrano mejo. V izbranih celicah nato poiščemo skupine s pomočjo metode t-SNE (*t-distributed stochas- tic neighbor embedding*).

S primerom na sliki 5.2 v grobem predstavimo razliko v času in odzivnosti sistema, ko poženemo analizo nad omenjenimi podatki. Za kakršno koli resno testiranje odzivnosti sistema bi potrebovali drugačen pristop. Upoštevati bi morali tip internetne povezave in uporabljeno strojno oziroma programsko opremo, vendar bodo meritve v našem primeru zadostovale za prikaz razlik (kjer zelena predstavlja hitro, rdeča pa počasno operacijo) obstoječe in predlagane arhitekture programa Orange.

Orange ima velike težave kadar pride do uvoza velikih podatkov. Pretvorba v podatkovno tabelo Orange je časovno zelo draga operacija in precej zmanjša odzivnost sistema. Branje datotek v naši pilotni implementaciji se zato raje izvrši v gradniku `Resolwe Datasets`. Prednost predlagane arhitekture je, da podatki živijo na strežniku in so takoj na voljo procesu analize. Kako so podatki na strežniku predstavljeni je s stališča uporabnika nepomembno. Podatki so lahko pripravljeni za delo glede na tip analiz in procesov, ki jih bomo izvajali. V naslednjem koraku, z gradnikom `Filter`, izberemo nabor genov oziroma celic, ki prestanejo nastavljene filtre. Gre za preprosto operacijo (štetje po vrsticah oz. stolpcih), ki ni računsko zahtevna in ne bremeni sistema. Vendar pri predlagani arhitekturi dodamo še en nivo kompleksnosti. To je komunikacija med strežnikom in končnim uporabnikom. Zelo očitno je, da bo odzivnost sistema za odtenek slabša, ko gre za tako preproste operacije, saj v predlagani arhitekturi izvedemo precej več dela v primerjavi s trenutno arhitekturo. V zadnjem koraku izvedemo še najzahtevnejšo operacijo v našem delovnem procesu. Na strežniku uporabimo implementacijo metode t-SNE, ki omogoča povzporedno izvajanje (*Multicore t-SNE*[1]) in tako z izrabo računskih virov na strežniku bistveno zmanjšamo njen izvajalni čas oziroma klic metode t-SNE v primeru 4.4. Izvajanje delovnega procesa smo z uporabo predlagane arhitekture približali eni minuti in tako v primerjavi s trenutno arhitekturo bistveno pohitrili.

S stališča ponovljivosti analiz z uporabo tehnologije Resolwe prav tako pridobimo na odzivnosti in izvajalnem času. Če parametrov v posameznih

[1] `https://github.com/DmitryUlyanov/Multicore-TSNE`

gradnikih ne spreminjamo in ponovno poženemo delovni proces, nam Resolwe vrne že izračunane rezultate. S trenutno arhitekturo smo pa primorani ponoviti vse postopke analize.

Poglavje 6

Zaključek

Cilj diplomskega dela je bil razviti prototipno rešitev, kjer bi vso podatkovno analitiko programa Orange izvajali na strežniku s pomočjo programskega orodja Resolwe. S predlagano arhitekturo smo uspeli na preprostem primeru ustvariti delovni proces v grafičnem vmesniku programa Orange, kjer se vse računsko zahtevne operacije izvajajo na temu namenjeni strežniški arhitekturi. Velikost projekta v grobem znaša 6000 vrstic. Od tega je približno 1000 vrstic porabljenih za določanje procesov na projektu Resolwe. Večji del ostale kode je bilo potrebne za razvoj grafičnih vmesnikov v gradnikih Orange. Izvorna koda, skupaj z navodili namestitve, je dostopna na portalu Github in je razdeljena na dva ločena projekta[1,2].

Z izdelavo prototipa smo spoznali, kje so največje omejitve trenutne rešitve. Grafične komponente in vizualizacije v okolju Orange so globoko prepletene s podatkovno tabelo Orange. V predlagani arhitekturi podatki živijo na strežniški aplikiciji, zato je trenutna interakcija s takimi gradniki mogoča šele, ko podatke prenesemo s strežnika. Spremljanje napredka procesa v razvitem prototipu ni v celoti podprto. Trenutna rešitev intervalno pošilja zahtevke HTTP na strežnik, dokler ne dobi statusa o uspešno oziroma neuspešno zaključenem procesu. Kompleksnost programske kode gradnikov

[1] `https://github.com/JakaKokosar/orange3-resolwe`
[2] `https://github.com/JakaKokosar/resolwe_project`

bistveno zmanjšamo, saj je uporaba Python knjižnice ReSDK, s katero poganjamo podatkovno analitiko na strežniku Resolwe, precej enostavna za uporabo in omogoča učinkovito integracijo v okolje Orange. Smo pa otežili sam razvojni cikel gradnikov, saj se morajo razvijalci gradnikov seznaniti z orodij in omejitvami tako na strežniški aplikaciji kot tudi v okolju Orange. Predlagana arhitektura je odvisna od stanja in zrelosti projekta Resolwe. Da bi uresničili zahteve, ki smo jih navedli v poglavju (3.1), je potreben razmislek o dodatnih funkcionalnosti:

- predstavitev podatkov na strežniški aplikaciji tako, da lahko z njimi interaktivno uporavljamo prek grafičnega vmesnika programa Orange,
- omogočiti podporo spletnih vtičnikov prek orodja ReSDK,
- poenostaviti razvoj procesov z uporabo programskega jezika Python.

V diplomskem delu smo pokazali, da je s pravim pristopom možno združiti dva odprtokodna projekta Resolwe in Orange in tako omogočiti interaktivno podatkovno analitiko v oblaku. Prihodnost projekta je odvisna predvsem od sodelovanja med razvijalci na projektih Resolwe in Orange, ki lahko skupaj ustvarijo moderen produkt, ki bo kos tudi najzahtevnejšim nalogam.

Literatura

[1] T. Barrett, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, M. Holko, A. Yefanov, H. Lee, N. Zhang, C. L. Robertson, N. Serova, S. Davis, and A. Soboleva. NCBI GEO: archive for functional genomics data sets—update. *Nucleic Acids Research*, 41(D1):D991–D995, 2013.

[2] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. KNIME-the konstanz information miner: version 2.0 and beyond. *ACM SIGKDD Explorations Newsletter*, 11(1):26–31, 2009.

[3] J. Demšar and B. Zupan. Orange: Data mining fruitful and fun-a: historical perspective. *Informatica*, 37(1), 2013.

[4] J. Demšar, B. Zupan, G. Leban, and T. Curk. Orange: From experimental machine learning to interactive data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 537–539. Springer, 2004.

[5] Genialis. Resolwe documentation. `https://resolwe.readthedocs.io/en/stable/index.html`, 2018. [Pridobljeno 25.7.2018].

[6] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research*, 37(1):1–13, 2009.

[7] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money. Big data: Issues and challenges moving forward. In *2013 46th Hawaii International Conference on System Sciences*, pages 995–1004, Jan 2013.

[8] T. Kalisky and S. R. Quake. Single-cell genomics. *Nature methods*, 8(4):311, 2011.

[9] J. Köster and S. Rahmann. Building and Documenting Workflows with Python-Based Snakemake. In *German Conference on Bioinformatics 2012*, volume 26 of *OpenAccess Series in Informatics (OASIcs)*, pages 49–56, 2012.

[10] D. Sacha, M. Sedlmair, L. Zhang, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, 268:164 – 175, 2017.

[11] S. Slater, S. Joksimović, V. Kovanovic, R. S. Baker, and D. Gasevic. Tools for educational data mining: A review. *Journal of Educational and Behavioral Statistics*, 42(1):85–106, 2017.

[12] A. Starič, J. Demšar, and B. Zupan. Concurrent software architectures for exploratory data analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(4):165–180, 2015.