TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kaspar Jesmin

Taastuvenergia ennustusmudelitele vajalikke ilmaandmeid pakkuva tööriista loomine

Bakalaureusetöö (9 EAP)

Juhendaja: Tiit Sepp, MSc

Tartu 2024

Taastuvenergia ennustusmudelitele vajalikke ilmaandmeid pakkuva tööriista loomine

Lühikokkuvõte:

Bakalaureusetöö eesmärk on luua süsteem, mis kiirendab ja lihtsustab ajalooliste ilmaandmete pärimist ja töötlemist. See hõlmab kasutajasõbraliku liidese loomist, mis suudab lahendada praeguste süsteemide puudujääke, sealhulgas aeglust, keerukust ja piiranguid.

Võtmesõnad:

Python, Docker, FastAPI, ilmaennustused, SQL, andmebaasid, päikeseenergia, gRPC

CERCS: P175 Informaatika

Development of a tool providing weather data for Renewable energy prediction models

Abstract:

The aim of this bachelor's thesis is to develop a system which allows for faster retrieval and processing of historical weather data. This includes creating a user-friendly API to address existing system drawbacks, such as slow processing, complexity, and limitations.

Keywords:

Python, Docker, FastAPI, weather forecasts, SQL, databases, solar energy

CERCS: P175 Informatics

Sisukord

Sissejuhatus

Ajaloolised ilmaandmed on olulised nii teadusuuringute kui ka praktiliste rakenduste jaoks, pakkudes väärtuslikku teavet näiteks energiatootlusmudelite loomiseks. Keeruliste liideste ja päringusüsteemide tõttu on andmed raskesti kättesaadavad. Käesolev töö keskendub nende probleemide lahendamisele, pakkudes efektiivsemat ja kasutajasõbralikumat viisi ERA5 andmete pärimiseks ja töötlemiseks. ERA5 on ECMWF-i (*European Centre for Medium-Range Weather Forecasts*) poolt loodud ilmaandmete andmestik.

Töö eesmärk on luua süsteem, mis kiirendab ja lihtsustab ajalooliste ilmaandmete pärimist ja töötlemist. See hõlmab kasutajasõbraliku liidese loomist, mis suudab lahendada praeguste süsteemide puudujääke, sealhulgas aeglust, keerukust ja piiranguid päringute suurusele.

Töö käigus luuakse vahekiht ECMWF-i poolt loodud rakendusliidesele. Vahekihti andmete hankimine on aeganõudev ning sõltuvalt soovitud ajaperioodi pikkusest võib võtta tunde või isegi päevi. Kui vahekihis on andmed olemas, siis on need läbi API kiirelt kättesaadavad.

Teises peatükis kirjeldatakse süvitsi ECMWF-i rakendusliidese probleeme ja sõnastatakse nende põhjal eesmärgid, mida peab loodav lahendus täitma. Kolmas peatükk kirjeldab ERA5 andmete olemust. Neljas peatükk selgitab rakenduses kasutatud tehnoloogiaid. Viies peatükk kirjeldab rakenduse arhitektuuri ning iga teenust eraldi. Seitsmes peatükk kirjeldab tehtud mõõtmisi ning tulemuste vastavust eesmärkidele. Järgnevad veel kokkuvõte ja võimalikud edasiarendused.

1. Mõisted ja terminid

API ehk rakendusliides (ingl *Application Programming Interface*): reeglid ja vahendid rakendusprogrammi suhtluseks [1].

JSON: lihtne inimloetav andmevahetusformaat, mis on tulnud javascriptist [2].

ECMWF ehk Euroopa Ilmaennustuse Keskus (ingl *European Centre for Medium-Range Weather Forecasts*): sõltumatu valitsustevaheline organisatsioon, põhiülesandeks on ülemaailmse ilmaennustuse koostamine kuni 15 päevaks ning selle jagamine liikmesriikidele [3].

YAML: inimloetav andmete serialiseerimise keel [4].

IDE ehk integreetirud arenduskeskkond (ingl *Integrated development environment*): Rakendus, mis aitab arendajal koodi kirjutada. Võib aidata näiteks rakenduse pakendamise, testimise, sõnade lõpetmise või refaktoreerimisega [5].

SQL ehk struktureeritud päringute keel (ingl *Structured Query Language*): Programmeerimiskeel relatsiooniliste andmebaaside halduseks ja kasutamiseks [6].

2. Eesmärgid ja nõuded

Tehnilised nõuded tulenevad vajadusest ajaloolistele ilmaandmetele kiirelt ja mugavalt ligi pääseda, mida ECMWF-i ametlik lahendus ei võimalda.
ERA5-st ilmaandmete allalaadimine on keeruline ja ajakulukas.
Järgnevalt on välja toodud lahendamist vajavad probleemid.

- Andmete allalaadimise kiirus.
    - Andmete allalaadimisel lisatakse päring järjekorda, suuri päringuid karistatakse ja need liiguvad järjekorras tahapoole. Eraldi karistatakse veel päringuid, mis laevad andmeid rohkem kui ühe kuu kohta korraga [7].
    - Kui päringuga hakatakse tegelema, siis olenevalt päringu suurusest võib töötlemine võtta tunde.
- Liides andmete tõmbamiseks on keeruline ning ajakulukas kasutada.
    - Ühel päringul on maksimaalseks lubatud suuruseks 60000 üksust [7].
    - Korraga on võimalik alla laadida vaid ühe aasta andmeid.
    - Kõik kellaajad, päevad ja kuud tuleb märkida ükshaaval.
    - Liides sisaldab palju erinevaid võimalikke välju, millest saab välja sorteerida need, mis on tihti kasulikud.
- Andmed ei pruugi olla vastavuses dokumentatsiooniga.
    - Andmete dokumentatsioon kirjeldab iga välja võimalikke vahemikke. On tekkinud probleem, et osad väärtused erinevad dokumentatsioonist.
    - Allalaadimisel on esinenud korduvaid ridu.

Oleks liialt ressursimahukas, kui iga töötaja andmeid iseseisvalt alla laeks ning end kõikide probleemide või omapäradega kurssi peaks viima. Seetõttu on vaja lahendust, mis teeb andmete kättesaamise lihtsamaks ja kiiremaks.
Kuna otse serverist ei ole võimalik andmeid kiiremini kätte saada, siis on otsustatud kasutada vahekihti, kuhu andmeid lisatakse soovitud piirkonna jaoks ühe korra. Näiteks on teada, et projekt vajab masinõppe mudelit erinevate Eesti päikeseparkide tootlikkuse ennustamiseks, siis käivitatakse andmete lisamine vahekihti soovitud ala ja ajaperioodi jaoks. Kui andmete lisamine on tehtud, siis on võimalik neid andmeid kiiresti kasutada.

Sealt tulenevalt peab rakendus täitma järgnevalt välja toodud nõuded.

- Andmete lisamine.
    - Andmetest, mis ei vasta dokumentatsioonile, peab teavitama kasutajat ja/või andmed automaatselt parandama.
    - Andmete lisamiseks tuleb luua kergesti kasutatav liides.
- Andmete kasutamine.
    - Peab võimaldama allalaadimist mitme aasta andmetele korraga.
    - Server peab kliendi päringuga tegelema hakkama koheselt – järjekorda klientidele ei tohiks tekkida.
    - Andmeid tuleb saata üle võrgu efektiivselt.
    - Andmete pärimiseks luua kergesti kasutatav liides.

Tulenevalt andmete kasutamise otstarbest ei pea lahendus omama kogu funktsionaalsust, mida pakub ECMWF teenus. Lahenduse andmed ei pea sisaldama kõiki välju, mida pakub ECMWF-i teenus, samuti ei pea suutma pakkuda kõiki ülemaailmseid andmeid aastast 1940 ning see lahendus peab toime tulema ainult paari paralleelse kasutajaga.

3. ERA5 Andmed

ERA5 andmestik on ECMWF-i viienda põlvkonna metoodikaga analüüsitud kliima- ja ilmaandmed kogu maailmast. ERA5 andmed on loodud kasutades ECMWF-i IFS(Integrated Forecasting System) ilmaennustusi, millel on rakendatud andmete assimilatsiooni. Andmete assimilatsiooni all peetakse silmas meetodit, kus eelmine ilmaennustus kombineeritakse vaatlusandmetega ja selle põhjal saadakse uued täpsemad ilmaandmed [8].

ERA5 pakub suurt hulka andmeid atmosfääri, mere ja maapinna kohta. Andmestikus leiduvad andmed on iga tunni kohta ning 31 km vahedega [9]. Andmestikku uuendatakse iga päev, viiepäevase viivitusega. Värskeimad andmed võivad olla erinevad lõplikust väljalaskest, suurte vigade korral teavitatakse kasutajaid. Lõplik väljalase tuleb kaks kuni kolm kuud peale esimest andmete avaldamist [10].

Andmeid pakutakse kahes formaadis: GRIB fail või eksperimentaalne NetCDF fail. On neli põhilist alam-andmestikku, tunnised andmed, kuised andmed, rõhutasemete andmed ja ühetasemelised andmed [10].

ERA5 on tasuta kasutamiseks [11].

4. Kasutatud tehnoloogiad

Kogu tarkvara on realiseeritud programmeerimiskeeles Python, kuna autoril on antud keelega kõige rohkem kogemusi, lisaks on lõppkasutajad andmeteadlased ja nende hulgas on Python domineeriv programmeerimiskeel. Rakendus on pakendatud Dockeri konteinerina, et oleks kasutatav nii pilvekeskkonnas kui potentsiaalsete klientide keskkondades.

4.1 Docker

Docker on platvorm rakenduste arendamiseks, tarnimiseks ja käitamiseks. Dockeriga saab pakendada ja käitada rakendust isoleeritud keskkonnas, mida kutsutakse konteineriks. Konteinerisse saab pakendada kõik tarkvaralised sõltuvused, et rakendust käitada ning seetõttu ei pea tuginema millelgi, mis on paigaldatud host-masinasse [12].

Konteinerid salvestatakse standardiseeritud formaati, mida saab jagada ning kasutada erinevate konteinerikäivtuskeskkondadega [13].

Konteineri käivitamise eelduseks on ainult Linuxi tuuma rakendusliidesed, mis on ühised kõikide konteinerikäivituskeskkondade vahel. Muudes operatsioonisüsteemides peale Linuxi kasutatakse ühilduvuskihti, milleks võib olla näiteks virtuaalmasin. Dockeri konteiner kasutab vähem ressursse kui virtuaalmasin, kuna ei emuleeri tervet masinat vaid kasutab host-masina süsteemi tuuma. Konteinerid on seetõttu vähema ressursikuluga ning lihtsamad hallata, see tähendab väiksemaid rahalisi kulusid [14].

Selleks, et käitada rakendust, mis koosneb mitmest konteinerist, on üheks võimaluseks Docker compose. Docker compose võimaldab defineerida konteinerid ja muud ressursid ning seadistada rakendust YAML failis [15].

4.2 Distribution Registry

Distribution Registry on rakendus, mis realiseerib Dockeri liidese Docker image salvestamiseks ja laadimiseks. Tegemist on avatud lähtekoodiga tarkvaraga [16].

Sama liidest realiseerib ka Docker Hub registry ja teised teenusepakkujad [17].

4.3 Docker image

Docker image on mall koos juhistega Dockeri konteineri loomiseks. Lihtsaim viis uute konteinerite loomiseks on olemasolevate avalike image'ite täiendamine oma seadistuste ja

failidega. Näiteks saab võtta aluseks Ubuntu image, paigaldada Apache veebiserver ja rakendus, mis kasutab veebiserverit ning nii luua rakendusele käituskeskkond [18].

4.4 Python

Python on üldotstarbeline objektorienteeritud interpreteeritav programmeerimiskeel. Python on lihtne ja kergesti loetav. Pythoni lihtsus teeb sellest keele, kus on võimalik väga kiiresti rakendusi luua [19].

Pythoni levinud kasutusaladeks on näiteks masinõpe ja serveripoolne veebiarendus. Pythonit kasutatakse tihti vahekihina, et panna omavahel suhtlema teistes keeltes loodud rakendused [20].

4.5 Pandera

Pandera on avatud lähtekoodiga projekt, mille eesmärk teha andmete valideerimist andmeraamidele ning seeläbi muuta andmetöötluse ahelaid rohkem loetavaks ja robustsemaks.

Pandera suudab valideerida näiteks pandase, polarsi, daski, modini, ja pyspark.pandase andmeraame [21].

4.6 FastAPI

FastAPI on modernne ja kiire veebiraamistik API-de ehitamiseks Python programmeerimiskeeles. Peamisteks FastAPI võimekusteks on:

- kasutamise lihtsus – intuitiivne, kiiresti õpitav ja hea toetus IDE-de poolt;
- kiire – üks kiireimaid Pythoni veebiraamistike;
- automaatselt koostatud OpenAPI dokumentatsioon.

FastAPI’t kasutavad ka tuntud ettevõtted nagu näiteks Microsoft, Uber ja Netflix [22].

4.7 SQLALchemy

SQLAlchemy on Pythoni teek, mis koosneb kahest põhilisest osast SQLAlchemy ORM ja SQLAlchemy Core. SQLAlchemy Core on samuti abstraktsioon SQL ees ning võimaldab SQL päringud läbi Python koodi. SQLAlchemy ORM on valikuline teek, mis on ehitatud SQLAlchemy Core peale [23].

SQLAlchemy toetab paljusid andmebaase, sealhulgas näiteks SQLite, PostgreSQL ja MySQL andmebaase [24].

4.8 Alembic

Alembic on andmebaasi migratsioonide tööriist, mis on tehtud SQLAlchemy loojate poolt. Alembic võimaldab automaatselt luua uusi migratsioone võrreldes omavahel andmebaasi hetkeseisu ja projekti koodis olevaid SQLAlchemy mudeleid.

Alembicu migratsioone saab kirjutata nii Pythonis kui ka puhtas SQL-is [25].

4.9 Protocol Buffers

Protocol Buffers on programmeerimiskeelest sõltumatu laiendus struktureeritud andmete serialiseerimiseks. Andmete struktuur tuleb ühe korra kirja panna ning seejärel on võimalik automaatselt loodud koodi abil andmeid kirjutada ja lugeda erinevates programmeerimiskeeletes [26].

Protocol Buffers kasutab binaarset andmete formaati, mis on kompaktsem ning mille tõttu on võimalik andmeid kiiremini kirjutada ja lugeda kui tekstil põhinevate formaatidega [27].

4.9.1 RPC

RPC ehk *remote procedure call* on tehnika, mille abil peaks üle võrgu tehtud päring jätma mulje nagu oleks see funktsiooni või meetodi väljakutse programmeerimiskeeles ühe protsessi siseselt. Järgnevalt on välja toodud mõned probleemid, mis sellega kaasnevad, sest üle võrgu liikuv päring on funktsiooni väljakutsest väga erinev.

- Kui kohalik funktsiooni väljakutse kas õnnestub või ebaõnnestub olenevalt funktsiooni sisendparameetritest, siis RPC väljakutse võib ebaõnnestuda probleemide tõttu, mille üle ei ole programmeerijal mingit kontrolli. Näiteks võib olla probleem internetiga või masin, mille poole pöördutakse on aeglane.
- Kui esimene päring ebaõnnestub ja seejärel päring uuesti sooritada, on võimalik, et võrgu taga olev masin lihtsalt ei tagastanud tulemust ja seetõttu võidi soovitud tegevust sooritada mitu korda.

Probleemidest hoolimata RPC laialt kasutusel ning uued RPC raamistikud on läbipaistvamad selle osas, et päring üle võrgu erineb kohalikust funktsiooni väljakutsest [28].

4.9.2 gRPC

gRPC on moderne avatud lähtekoodiga RPC raamistik.

Põhilised kasutusalad:

- efektiivne mikroteenuste omavaheline suhtlus;
- mobiilsete seadmete ja brauserite ühendamine *back-end* teenustega.

Mõned kasulikud võimekused:

- efektiivne suhtlus üle võrgu;
- toetatud teegid 11 programmeerimiskeeles;
- kahesuunaline voogedastus suhtlus üle http/2 [29].

gRPC teenused defineeritakse kasutades Protocol Buffersit [30].

4.10 Relatsiooniline andmebaas

Relatsiooniline andmebaas organiseerib andmed ridadest ja tulpadest koosnevate tabelitena. Enamasti on andmed jaotatud erinevate tabelite vahel, mida saab omavahel siduda võtmetulpade abil [31].

Relatsioonilised andmebaasid kasutavad Don Chamberlin ja Ray Boyce poolt loodud programmeerimiskeelt SQL. SQL keelega saab andmebaasi ridu kergelt lisada, uuendada, lugeda ja kustutada [31].

4.10.1 PostgreSQL

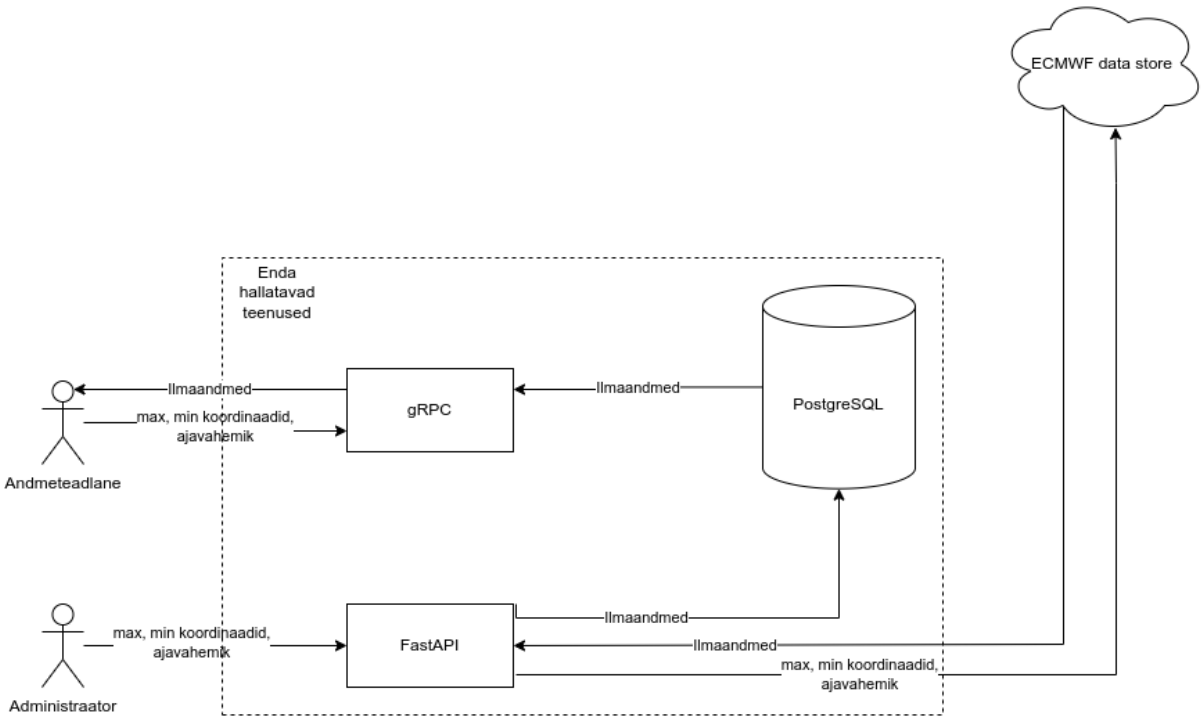
PostgreSQL on avatud lähtekoodiga relatsiooniline andmebaas. PostgreSQL'i on arendatud üle 35 aasta ning selle käigus on välja kujunenud kuvand paljude võimalustega töökindlast ja turvalisest andmebaasist. PostgreSQL'il on mitmeid võimekaid lisasid, nagu näiteks PostGIS, mis laiendab geograafiliste andmete haldamise võimalusi [32].

4.10.2 B- Tree indeks

B-Tree indeks on kasuks järjestatavate andmete puhul, mille pärimisel kasutatakse tihti võrdlusoperaatoreid (< , <= , = , >= , >). Nende operaatorite puhul kaalub PostgreSQL päringute planeerija b-tree indeksi kasutamist [33].

5. Arhitektuur

Rakendus koosneb kolmest põhilisest teenusest, milleks on PostgreSQL andmebaas, gRPC rakendusliides ja FastAPI rakendusliides. Joonisel 1 on kujutatud teenuste suhtlust kasutajaga ja teenuste omavahelist suhtlust.



Joonis 1. Rakenduse arhitektuur

5.1 gRPC rakendusliides

Üheks rakendusliideseks on gRPC, sest see võimaldab efektiivselt andmeid üle võrgu saata kasutades Protocol Buffers'it. Rakendusliideses kasutatavate teenuste ja andmete struktuuri määramine toimub .proto failides. Antud rakendusliidese sisendiks on sõnum, mis koosneb neljast koordinaadist, mis on määratletud kui vähim pikkus- ja laiuskraad, suurim pikkus- ja laiuskraad ning kahest alg- ja lõppajapunktist. Rakendusliidese väljundiks on sõnum, mis sisaldab ajapunkti, pikkus-ja laiuskraadi ning kõiki ilmaandmeid. Defineeritud on, et teenus saab ette sisendsõnumi ja tagastab väljundsõnumi.

Kui kasutaja edastab sõnumi teenusele, siis teenus tagastab vastussõnumiga kõik ilmaandmed, mille koordinaadid ja ajapunkt jäävad sisendsõnumis määratletud vahemikesse.

Selleks, et gRPC serverist andmeid pärida, peab olema ka kliendil ligipääs .proto failidele. See sai loodud git *submodule* abil. Kõik .proto failid on eraldi giti repositooriumis, millega tuleb ka selgitus koos käsurea käskudega, kuidas antud failidest saab luua Python programmeerimiskeele klassid. Kliendil ei ole kohustust kasutada Pythonit, gRPC tugi on paljudel programmeerimiskeeltel.

5.2 PostgreSQL andmebaas

Selleks, et andmeid hoida andmebaasis, on kasutusel tabel, mis sisaldab kõiki ilmaandmete välju, ajapunkti ning laius- ja pikkuskraadi. Pikkus ja laiuskraadid kasutavad b-tree indeksit, et päringud, mis sisaldavad suurem-väiksem võrdluseid, oleksid efektiivsemad.

Andmebaasiga oleks saanud kasutada PostGIS laiendust, mis oleks pakkunud rohkem võimalusi geograafiliste andmete pärimisel ning võimalik, et oleks olnud efektiivsem. Näiteks võimaldab PostGIS laiendus defineerida hulknurkasid, kuhu sisse peavad oodatud tulemused jääma. Otsustasin jääda siiski puhta PostgreSQL juurde, et hoida lahendus lihtsana, sest hiljem on alati võimalus võimekusi lisada.

Rakenduses kasutatakse SQLAlchemy ORM-i andmebaasiga suhtlemiseks. Selleks on defineeritud Pythoni klassid, mis sisaldavad kõiki andmebaasi väljasid.

Et võimaldada andmete struktuuri kontrollitud muutmist on kasutusel ka andmebaasi migratsioonid. Migratsioonide jaoks on kasutusel Pythoni teek Alembic, mis võimaldab koostada automaatselt uue migratsiooni võrreldes omavahel andmebaasi hetkeseisu ja SQLAlchemyga loodud Pythoni klasse.

5.3 FastAPI rakendusliides

FastAPI rakendusliides on kasutuses andmete lisamiseks andmebaasi. Et lisada andmeid andmebaasi, peab kasutaja andma API lõpp-punktile ette alg- ja lõppajapunkt ning neli koordinaati, mis on määratletud kui vähim pikkus- ja laiuskraad ning suurim pikkus-ja laiuskraad. Kui päring läheb edukalt läbi, siis tagastatakse kasutajale http olekukood 201 ja JSON sisuga {'result': 'successfully started downloading'}. Samal ajal jätkab server taustal töö lõpetamisega. Kui töö käigus midagi valesti läheb ja andmeid ei ole võimalik andmebaasi lisada, siis probleem logitakse.

6. Andmete valideerimine

ERA5 API-st laetavad andmed ei ole alati dokumentatsioonile vastavad. Probleemid, mida olen kohanud:

- esineb juhtumeid, kus väljade väärtused ei mahu dokumentatsioonis antud piiridesse. Näiteks sätestab dokumentatsioon vahemiku 0-1, andmetes võib olla nt -0.00000843 või 1.00000723;
- andmed saabuvad vigaselt – on olnud olukord, kus iga koordinaatiga rida on topelt ja igas teises reas on kõikide ilmaandmete väljades nullväärtused.

Probleemide lahenduseks on defineeritud Pandera klass “Era5Schema”. Era5Schema sisaldab kõiki välju, mis lähevad andmebaasi tabelisse. Iga välja juures on kirjas andmetüüp ja vastavalt vajadusele ka muud nõudmised välja kohta. Näiteks:

- hulk, mille sisse peab väärtus kuuluma;
- väärtuse kohustuslikkus – vahest võib väärtus ka puududa.

Nõuded tulevad ERA5 andmete dokumentatsioonist [9] ja kui andmeraam määratud skeemile ei vasta, siis viga logitakse.

7. Tulemused

7.1 Testkeskkond

Töös väljatoodud katsetused viidi läbi töö autori sülearvutis protsessoriga i5-10210U ning 16 GB vahemäluga. Arvutis töötasid samal ajal ka teised protsessid, mis võisid mõjutada katsetuste kestust,

Ajamõõtmised viidi läbi kasutades Pythoni standardteeki time. Töös esitatud näidisega samaväärseid katseid viidi läbi korduvalt ning tulemused jäid alati samasse suurusjärku.

7.2 gRPC

Näidiskatse andmed pärinevad Joonisel 2 väljatoodud piirkonnast 01.02.2022 – 31.08.2022 perioodil.

Joonis 2. valitud pindala

Näidiskatsel võttis päring gRPC kliendi poolt aega 50.35 sekundit. Serveris jaotus päring nii:

- 22.62 sekundit andmete pärimiseks kasutades SQLAlchemy ORM-i;
- 25.16 sekundit SQLAlchemy objektide muundamine gRPC objektideks.

Peale sõnumi kättesaamist kliendi poolt logitud viimase ajahetke ning serveris enne sõnumi saatmist viimase logitud ajahetke vahele jäi 2.44 sekundit. Seega 2.44 sekundi jooksul serialiseeriti andmed, saadeti üle võrgu ja deserialiseeriti.

Sõnum gRPC objektina oli 150409217 baiti ehk umbes 150 MB.

7.2.1 Võrdlus JSON API-ga

Kasutades objektil funktsiooni MessageToJson, siis saadud sõnum oli suurusega 636183601 baiti ehk umbes 636 MB. Seega gRPC'ga sõnumi saatmine mahu poolest on üle nelja korra efektiivsem kui oleks võrdväärse JSON-it tagastava API-ga.

Põhiline ajakulu tuleks siiski serialiseerimisest JSON-iks ja tagasi. Funktsiooni MessageToJson väljakutse võttis üle 200 sekundi aega. Kui eeldada, et tagasi muundamine võtab sama kaua aega, siis antud päring muutuks juba üle kuue minuti aeglasemaks. On võimalik, et sõnumi JSON-iks muundamist saab palju kiiremini teha, aga siiski ei saa see olla ligilähedal binaarse formaadiga, mida kasutab gRPC ja mille puhul võttis kõik kokku alla 3 sekundi.

8. Kokkuvõte

Töös käsitleti ajalooliste ilmaandmete kättesaadavuse probleeme ja töötati välja lahendus, mis lihtsustab ERA5 andmete allalaadimist ja kasutamist. Peamisteks saavutusteks oli kasutajasõbraliku ja efektiivse andmete pärimise süsteemi loomine, mis lahendab mitmeid ECMWF-i ametliku lahenduse puudusi.
Testides mõõdeti süsteemi jõudlust ja leiti, et gRPC rakendusliides on tõhus viis andmete üle võrgu saatmiseks, sest on märkimisväärselt kiirem ja mahult efektiivsem kui traditsioonilised JSON-põhised lahendused.

Andmeid on võimalik alla laadida pika perioodi kohta. Puuduvad ECMWF API piirangud, mis takistavad andmete laadimist enam kui aasta kohta korraga ning API liides on lihtsasti kasutatav.

Andmete valideerimiseks kasutati Pandera raamistikku, mis tagab, et andmed vastavad nõutud standarditele.

Rakendus on pakendatud Dockeri konteineritesse, et tagada selle lihtne kasutamine erinevates keskkondades.

Töö tulemusel loodi süsteem, mis kiirendab ja lihtsustab ajalooliste ilmaandmete kättesaamist, lahendades senised kitsaskohad.

9. Võimalikud edasiarendused

Töö edasiarenduseks on võimalusi mitmeid. Mõned kohad, kust oleks võimalik edasi minna:

- suuremate päringute puhul oleks hea kasutada voogedastust – see vähendaks mälukasutust ja võimaldada töötlemise kiiremat alustamist;
- uurida võimalusi geograafiliste andmete salvestamiseks ja pärimiseks PostGIS laiendusega;
- probleemid andmetega on tihti sarnased, mõningaid neist on võimalik lahendada automaatselt;
- FastAPI background task, mida kasutatakse andmebaasi andmete lisamiseks on mõeldud pigem väiksemate tööde jaoks, kasutusele tuleks võtta tööde järjekorrad ning eraldi teenus, mis töid täide viiks (nt teek Celery);
- lisada ühikteste ja integratsiooniteste;
- täiendada logimist - luua logide jaoks ühtne formaat ning lisada logimist ka gRPC teenusele.

Suurimate kitsaskohtade tuvastamiseks tarvis rakendusele päris kasutajaid. Kasutades rakendust pidevalt, on hästi aru saada, kuhu lisada logimist ja kuhu vajadusel tuua sisse ka teisi monitooringu tööriistu. Kasutades on näha, kas edasiarendused nimetatud kohtades on ka tegelikult vajalikud, ei ole mõistlik liiga vara alustada rakenduse optimiseerimisega. „„The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming.“ [34]“

10. Viidatud kirjandus

1. API. [Online]. [cited 2024 05 15. Available from: https://akit.cyber.ee/term/3088.
2. SQL. [Online]. [cited 2024 05 15. Available from: https://akit.cyber.ee/term/10850-json.
3. Keskkonnaagentuur. [Online].; 2020 [cited 2024 05 15. Available from: https://www.ilmateenistus.ee/definition/ecmwf/.
4. YAML. [Online]. [cited 2024 05 15. Available from: https://akit.cyber.ee/term/13870-yaml.
5. Amazon. [Online]. [cited 2024 05 15. Available from: https://aws.amazon.com/what-is/ide/.
6. SQL. [Online]. [cited 2024 05 15. Available from: https://akit.cyber.ee/term/8501-sql.
7. ECMWF. A new CDS soon to be launched - expect some disruptions. [Online].; 2024 [cited 2024 05 01. Available from: https://forum.ecmwf.int/t/a-new-cds-soon-to-be-launched-expect-some-disruptions/1607.
8. Hersbach , Bell B, Berrisford P, Hirahara S. The ERA5 global reanalysis. Quarterly Journal of the Royal Meteorological Society. 2020 May.
9. ERA5: data documentation. [Online]. [cited 2024 01 14. Available from: https://confluence.ecmwf.int/display/CKB/ERA5%3A+data+documentation.
10. ERA5 hourly data on single levels from 1940 to present. [Online]. [cited 2024 01 14. Available from: https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=overview.
11. ECMWF. Licence to Use Copernicus Products. [Online].; 2019 [cited 2024 01 14. Available from: https://apps.ecmwf.int/datasets/licences/copernicus/.

12. Docker. Use containers to Build, Share and Run your applications. [Online]. [cited 2024 01 14. Available from: https://www.docker.com/resources/what-container/.

13. Walli S. Demystifying the Open Container Initiative (OCI) Specifications. [Online].; 2017 [cited 2024 01 14. Available from: https://www.docker.com/blog/demystifying-open-container-initiative-oci-specifications.

14. Understanding the Docker Internals. [Online].; 2017 [cited 2024 01 14. Available from: https://medium.com/@BeNitinAgarwal/understanding-the-docker-internals-7ccb052ce9fe.

15. Docker. Docker Compose overview. [Online]. [cited 2024 01 14. Available from: https://docs.docker.com/compose/.

16. Cloud Native Computing Foundation. Distribution Registry. [Online].; 2023 [cited 2024 01 14. Available from: https://distribution.github.io/distribution/.

17. Docker. Registry. [Online]. [cited 2024 01 14. Available from: https://docs.docker.com/registry/.

18. Docker. Docker overview. [Online]. [cited 2024 01 14. Available from: https://docs.docker.com/get-started/overview/.

19. Python Software Foundation. What is Python? Executive Summary. [Online]. [cited 2024 01 24. Available from: https://www.python.org/doc/essays/blurb/.

20. Zola A. Python. [Online].; 2021 [cited 2024 01 14. Available from: https://www.techtarget.com/whatis/definition/Python.

21. The Open-source Framework for Precision Data Testing. [Online]. [cited 2024 05 14. Available from: https://pandera.readthedocs.io/en/stable/.

22. [Online]. [cited 2024 05 13. Available from: https://fastapi.tiangolo.com/.

23. Overview. [Online].; 2024 [cited 2024 05 13. Available from: https://docs.sqlalchemy.org/en/20/intro.html.

24. Features and Philosophy. [Online]. [cited 2024 05 14. Available from: https://www.sqlalchemy.org/features.html.

25. Sqlalchemy. [Online].; 2023 [cited 2024 05 14. Available from: https://github.com/sqlalchemy/alembic.

26. Google. Protocol Buffers. [Online].; 2024 [cited 2024 05 14. Available from: https://protobuf.dev/.

27. Bello G. What is Protobuf? [Online].; 2024 [cited 2024 05 14. Available from: https://blog.postman.com/what-is-protobuf/.

28. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. 1st ed.: O'Reilly Media; 2017.

29. Google. Who is using gRPC and why. [Online]. [cited 2024 05 14. Available from: https://grpc.io/about/.

30. Google. [Online]. [cited 2024 05 14. Available from: https://grpc.io/.

31. IBM. What is a relational database? [Online]. [cited 2024 01 04. Available from: https://www.ibm.com/topics/relational-databases.

32. PostgreSQL. About. [Online]. [cited 2024 05 13. Available from: https://www.postgresql.org/about/.

33. PostgreSQL. Index Types. [Online]. [cited 2024 05 13. Available from: https://www.postgresql.org/docs/current/indexes-types.html#INDEXES-TYPES-BTREE.

34. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms; 1968.

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Kaspar Jesmin,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose "Taastuvenergia ennustusmudelitele vajalikke ilmaandmeid pakkuva tööriista loomine", mille juhendaja on Tiit Sepp, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Kaspar Jesmin

15.05.2024