epivizfileserver Documentation

Jayaram Kancherla

Jul 26, 2021

Contents

Epiviz file Server is a scalable data query and compute system for indexed genomic files. In addition to querying data, users can also compute transformations, summarization and aggregation using NumPy functions directly on data queried from files.

Since the genomic files are indexed, the library will only request and parse necessary bytes from these files to process the request (without loading the entire file into memory). We implemented a cache system to efficiently manage already accessed bytes of a file. We also use dask to parallelize computing requests for query and transformation. This allows us to process and scale our system to large data repositories.

This blog post (Jupyter notebook) describes various features of the file server library using genomic files hosted from the NIH Roadmap Epigenomics project.

The library provides various modules to

- Parser: Read various genomic file formats,
- Query: Access only necessary bytes of file for a given genomic location,
- Compute: Apply transformations on data,
- Server: Instantly convert the datasets into a REST API
- Visualization: Interactive Exploration of data using Epiviz (uses the Server module above).

Note:

- The Epiviz file Server is an open source project on GitHub
- Let us know what you think and any feedback or feature requests to improve the library!

CHAPTER 1

Contents

1.1 Installation

1.1.1 Using PyPI

To install the package from PyPi,

```
pip install epivizfileserver
```

1.1.2 Development Version

To install the development version from GitHub: Install using pip

```
pip install git@github.com:epiviz/epivizFileParser.git
```

you can also clone the repository and install from local directory using *pip*

Note: If you don't have sudo rights to install the package, you can install it to the user directory using

```
pip install --user epivizfileserver
```

1.2 Tutorial

This blog post (Jupyter notebook) describes various features of the file server library using genomic files hosted from the NIH Roadmap Epigenomics project.

Note: This post describes a general walkthrough of the features of the file server. More usecases will be posted soon!

1.2.1 Import Measurements from File

Since large data repositories contains hundreds of files, manually adding files would be cumbersome. In order to make this process easier, we create a configuration file that lists all files with their locations. An example configuration file is described below -

1.2.2 Configuration file

The following is a configuration file for data hosted on the roadmap FTP server. This contains data for ChIP-seq experiments for the H3k27me3 marker in *Esophagus* and *Sigmoid Colon* tissues. Most fields in the configuration file are self explanatory.

```
[
    {
        url: "https://egg2.wustl.edu/roadmap/data/byFileType/signal/consolidated/
↪macs2signal/foldChange/E079-H3K27me3.fc.signal.bigwig",
        file_type: "bigwig",
        datatype: "bp",
        name: "E079-H3K27me3",
        id: "E079-H3K27me3",
        annotation: {
            group: "digestive",
            tissue: "Esophagus",
            marker: "H3K27me3"
        }
    }, {
        url: "https://egg2.wustl.edu/roadmap/data/byFileType/signal/consolidated/
↪macs2signal/foldChange/E106-H3K27me3.fc.signal.bigwig",
        file_type: "bigwig",
        datatype: "bp",
        name: "E106-H3K27me3",
        id: "E106-H3K27me3",
        annotation: {
            group: "digestive",
            tissue: "Sigmoid Colon",
            marker: "H3K27me3"
        }
    }
]
```

Once the configuration file is generated, we can import these measurements into the file server. We first create a *MeasurementManager* object which handles measurements from files and databases. We can then use the helper function *import_files* to import all measurements from this configuration file.

```
mMgr = MeasurementManager()
fmeasurements = mMgr.import_files(os.getcwd() + "/roadmap.json", mHandler)
fmeasurements
```

1.2.3 Query for a genomic location

After loading the measurements, we can query the object for data in a particular genomic region using the *get_data* function.

```
result, err = await fmeasurements[1].get_data("chr11", 10550488, 11554489)
result.head()
```

The reponse is a tuple, DataFrame that contains all results and an error if there is any.

1.2.4 Compute a Function over files

We can define and create new measurements that can be computed using a *Numpy* function over the files loaded from the previous step.

Note: you can also write a custom statistical function, that applies to every row in the DataFrame. It must follow the same syntax as any *Numpy* row-apply function.

As an example to demonstrate, we can calculate the average ChIP-seq expression for *H3K27me3* marker.

```
computed_measurement = mMgr.add_computed_measurement("computed", "avg_ChIP_seq",
↪"Average ChIP seq expression",
                                                      measurements=fmeasurements, computeFunc=numpy.
↪mean)
```

After defining a computed measurement, we can query this measurement for a genomic location.

```
result, err = await computed_measurement.get_data("chr11", 10550488, 11554489)
result.head()
```

1.2.5 Setup a REST API

Often times, developers would like to include data from genomic files into a web application for visualization or into their workflows. We can quickly setup a REST API web server from the measurements we loaded -

```
from epivizfileserver import setup_app
    app = setup_app(mMgr)
    app.run(port=8000)
```

The REST API is an asynchronous web server that is built on top of SANIC.

1.2.6 Query Files from AnnotationHub

We can also use the Bioconductor's AnnotationHub to search for files and setup the file server. We are working on simplifying this process.

Annotation Hub API is hosted at https://annotationhub.bioconductor.org/.

We first download the annotationhub sqlite database for available data resources.

```
wget http://annotationhub.bioconductor.org/metadata/annotationhub.sqlite3
```

After download the resource database from AnnotatiobnHub, we can now load the sqlite database into python and query for datasets.

```
import pandas
import os
import sqlite3

conn = sqlite3.connect("annotationhub.sqlite3")
```

(continues on next page)

(continued from previous page)

```
cur = conn.cursor()
cur.execute("select * from resources r JOIN input_sources inp_src ON r.id = inp_src.
↪resource_id;")
results = cur.fetchall()
pd = pandas.DataFrame(results, columns = ["id", "ah_id", "title", "dataprovider",
↪"species", "taxonomyid", "genome",
                                          "description", "coordinate_1_based",
↪"maintainer", "status_id",
                                          "location_prefix_id", "recipe_id",
↪"rdatadateadded", "rdatadateremoved",
                                          "record_id", "preparerclass", "id",
↪"sourcesize", "sourceurl", "sourceversion",
                                          "sourcemd5", "sourcelastmodifieddate",
↪"resource_id", "source_type"])
pd.head()
```

For the purpose of the tutorial, we will filter for Sigmoid Colon ("E106") and Esophagus ("E079") tissues, and the ChipSeq Data for "H3K27me3" histone marker files from the roadmap epigenomics project.

```
roadmap = pd.query('dataprovider=="BroadInstitute" and genome=="hg19"')
roadmap = roadmap.query('title.str.contains("H3K27me3") and (title.str.contains("E106
↪") or title.str.contains("E079"))')
# only use fc files
roadmap = roadmap.query('title.str.contains("fc")')
roadmap
```

After filtering for resources we are interested in, we can load them into the file server using the *import_ahub* helper function.

```
mMgr = MeasurementManager()
ahub_measurements = mMgr.import_ahub(roadmap)
ahub_measurements
```

The rest of the process is similar as described in the beginning of this tutorial.

1.3 Workspaces or Usecases setup using the File Server

The following workspaces have been setup using the Epiviz File Server

1. BICCN/NEMO Miniatlas Mouse MOp Dataset
2. BICCN Cross Species Dataset

1.4 Deployment

1.4.1 Using In built Sanic server (for development)

Sanic provides a default asynchronous web server to run the API. As a working example, checkout the Roadmap project from the Usecases section.

```
app = setup_app(mMgr)
app.run("0.0.0.0", port=8000)
```

1.4.2 Deploy using gunicorn + supervisor (for production)

Setup virtualenv and API

This process assumes the root API directory is */var/www/epiviz-api*

Setup virtualenv either through pip or conda

```
cd /var/www/epiviz-api
virtualenv env
source env/bin/activate
pip install epivizfileserver
```

A generic version of the API script would look something like this (add this to */var/www/epiviz-api/epiviz.py*)

```
from epivizfileserver import setup_app, create_fileHandler, MeasurementManager
from epivizfileserver.trackhub import TrackHub

# create measurements to load multiple trackhubs or configuration files
mMgr = MeasurementManager()

# create file handler, enables parallel processing of multiple requests
mHandler = create_fileHandler()

# add genome. - for supported genomes
# check https://obj.umiacs.umd.edu/genomes/index.html
genome = mMgr.add_genome("mm10")
genome = mMgr.add_genome("hg19")

# load measurements/files through config or TrackHub

# setup the app from the measurements manager
# and run the app
app = setup_app(mMgr)

# only if this file is run directly!
if __name__ == "__main__":
    app.run(host="127.0.0.1", port=8000)
```

Install dependencies

1. Supervisor (system wide) - http://supervisord.org/
2. Gunicorn (to the virtual environment) - https://gunicorn.org/

```
# if using ubuntu
sudo apt install supervisor

# activate virtualenv that runs the API
source /var/www/epiviz-api/env/bin/activate
pip install gunicorn
```

Configure supervisor

Add this configuration to */etc/supervisor/conf.d/epiviz.conf*

This snippet also assumes epiviz-api repo is in */var/www/epiviz-api*

```
[program:gunicorn]
directory=/var/www/epiviz-api
environment=PYTHONPATH=/var/www/epiviz-api/bin/python
command=/var/www/epiviz-api/env/bin/gunicorn epiviz:app --log-level debug --bind 0.0.
↪0.0:8000 --worker-class sanic.worker.GunicornWorker
autostart=true
autorestart=true
stderr_logfile=/var/log/gunicorn/gunicorn.err.log
stdout_logfile=/var/log/gunicorn/gunicorn.out.log
```

Enable Supervisor configuration

```
sudo supervisorctl reread
sudo supervisorctl update

service supervisor restart
```

Note: check status of supervisor to make sure there are no errors

Add Proxypass to nginx/Apache

(the port number here should match the binding port from supervisor configuration

for Apache

```
sudo a2enmod proxy
sudo a2enmod proxy-http

# add this to the apache site config
ProxyPreserveHost On
<Location "/api">
    ProxyPass "http://127.0.0.1:8000/"
    ProxyPassReverse "http://127.0.0.1:8000/"
</Location>
```

for nginx

```
# add this to nginx site config

upstream epiviz_api_server {
    server 127.0.0.1:8000 fail_timeout=0;
}

location /api/ {
    proxy_pass http://epiviz_api_server/;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_redirect off;
}
```

1.5 License

1.6 Contributors

- Jayaram Kancherla <jayaram.kancherla@gmail.com>
- Yifan Yang <yang7832@umd.edu>
- Hector Corrada Bravo <hcorrada@gmail.com>

1.7 Changelog

1.7.1 Version 0.1.2

- Package pushed to PyPi
- Updated readme and documentation

1.7.2 Version 0.1

- First release!

1.8 epivizfileserver

1.8.1 epivizfileserver package

Subpackages

epivizfileserver.client package

Submodules

epivizfileserver.client.EpivizClient module

Module contents

epivizfileserver.handler package

Submodules

epivizfileserver.handler.HandlerNoActor module

epivizfileserver.handler.handler module

epivizfileserver.handler.utils module

Module contents

epivizfileserver.measurements package

Submodules

epivizfileserver.measurements.measurementClass module

epivizfileserver.measurements.measurementManager module

Module contents

epivizfileserver.server package

Submodules

epivizfileserver.server.request module

epivizfileserver.server.utils module

Module contents

epivizfileserver.trackhub package

Submodules

epivizfileserver.trackhub.TrackHub module

Module contents

Submodules

epivizfileserver.cli module

Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search