Reproducibility review of: Random Data Distribution for Efficient Parallel Point Cloud Processing

Carlos Granell [orcid: 000-0003-1004-9695]

2024-05-23

This report is part of the reproducibility review at the AGILE conference. For more information see https://reproducible-agile.github.io/. This document is published on OSF at https://osf.io/ymc3t. To cite the report use

> Granell, C. (2024, April 18). Reproducibility review of: Random Data Distribution for Efficient Parallel Point Cloud Processing. https://doi.org/10.17605/osf.io/ymc3t

Reviewed paper

> Teuscher, B., and Werner, M.: Random Data Distribution for Efficient Parallel Point Cloud Processing, AGILE GIScience Ser., 5, 15, https://doi.org/10.5194/agile-giss-5-15-2024, 2024

Summary

The workflow provided was **partially reproduced** in the sense that I was able to reproduce the system reported in the paper *qualitatively*. That is, I did not reproduce the experiment or other results (figures, etc.) included in the article, but I was able to set up and run the system described in the paper and test it with some query samples. Therefore, by doing this, this report ensures that the system works as described in the manuscript.

Reproducibility reviewer notes

The manuscript included a DASA section, where a link to a public GitHub repository https://github.com/tum-bgd/2024-AGILE-RandomDataDistribution was included to the point cloud data management system. The repo included a `README` file, MPL-2.0 licence, supporting python scripts and sample data. I followed the instructions in the `README` file to build the Docker image, run the server, and load and query some data. I used an online arrow viewer to visualise the resulting `test.arrow` file. So, I could reproduce these instructions locally. Below details of the workflow followed.

I opened one terminal window (*server* terminal) to run the commands to start the server

Prepare sample data

The GitHub repository included a python script `scripts/ahn.py` to download some sample data.

```
python scripts/ahn.py --dataset AHN3 --filenames C_69AZ1
```

Problem: I ran `scripts/ahn.py` and got an error due to the *match statement*. I've got python 3.8.10 installed in my local machine and the [match statement was new in python 3.10] (https://docs.python.org/3/reference/compound_stmts.html#match). Easy fixed.

Start server

```
# build image
docker build --tag "crux_test" .
```

```
# run interactive
docker run -it --rm --env-file .env -e PORT=3000 -p 3000:3000 -v ./data:/crux/data crux_test
```

Check status

I opened another terminal window (*test* terminal) to run the following commands

```
$ curl -G '0.0.0.0:3000/status' | jq

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    75  100    75    0     0  37500      0 --:--:-- --:--:-- --:--:-- 37500
{
  "config": {
    "host": "172.17.0.2",
    "port": 3000,
    "coordinators": []
  },
  "workers": []
}
```

Load some data

```
$ curl -G '0.0.0.0:3000/load' -d 'uris=./data/AHN3/C_69AZ1.LAZ'
```

This command execution produced some log output captured in the *server* terminal

```
 2024-04-15T13:53:38.130486Z DEBUG tower_http::trace::on_request: started processing request
    at /usr/local/cargo/registry/src/index.crates.io-6f17d22bba15001f/tower-http-0.5.2/src/trace/on_request.rs:80
    in tower_http::trace::make_span::request with method: GET, uri: /load?uris=./data/AHN3/C_69AZ1.LAZ, version: HTTP/1.1

  2024-04-15T13:53:38.131252Z  INFO crux_server::handlers::load: LoadRequest {
    collection: Some(
        "default",
    ),
    uris: [
        "./data/AHN3/C_69AZ1.LAZ",
    ],
    delta: None,
    from: None,
    to: None,
    workers: None,
    store: None,
    compress: false,
}
    at crux-server/src/handlers/load.rs:59
    in tower_http::trace::make_span::request with method: GET, uri: /load?uris=./data/AHN3/C_69AZ1.LAZ, version: HTTP/1.1

  2024-04-15T13:53:38.131354Z  INFO crux_server::handlers::load: Processing uri ./data/AHN3/C_69AZ1.LAZ
```

```
    at crux-server/src/handlers/load.rs:185
    in tower_http::trace::make_span::request with method: GET, uri: /load?uris=./data/AHN3/C_69AZ1.LAZ, version: HTTP/1.1

  2024-04-15T13:53:38.158180Z  INFO crux_server::handlers::load: Set fraction to None
    at crux-server/src/handlers/load.rs:205
    in tower_http::trace::make_span::request with method: GET, uri: /load?uris=./data/AHN3/C_69AZ1.LAZ, version: HTTP/1.1

  2024-04-15T13:54:31.838130Z DEBUG tower_http::trace::on_response: finished processing request, latency: 53700 ms, status: 200
    at /usr/local/cargo/registry/src/index.crates.io-6f17d22bba15001f/tower-http-0.5.2/src/trace/on_response.rs:114
    in tower_http::trace::make_span::request with method: GET, uri: /load?uris=./data/AHN3/C_69AZ1.LAZ, version: HTTP/1.1
```

Query data

```
# query a random sample of the loaded data
$ curl -G '0.0.0.0:3000/points?p=0.001' --output test.arrow
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 10.3M  100 10.3M    0     0   986k      0  0:00:10  0:00:10 --:--:-- 2995k
```

This generated the `test.arrow` file, which I could inspect it in an online arrow viewer. *Update: Later version of the* ***README*** *file included a python snippet to ease arrow data visualisation.*

```
# query by x, y, z and importance bounds
$ curl -G '0.0.0.0:3000/points?bounds=174000,315000,0,0,174060,315060,1000,1' --output test_2.arrow
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2976k  100 2976k    0     0  4145k      0 --:--:-- --:--:-- --:--:-- 4139k
```

This generated the `test_2.arrow` file, which I could inspect it in a online arrow viewer.

Comments to the authors

The GitHub repository was properly described. Below some suggestions:

- It is strongly recommended to specify the required versions of the Python libraries used. Adding a `requirements.txt` is ideal for specifying the versions used. Otherwise, a statement in the `README` file may be sufficient too. *Note: Authors fixed it and included an statement of the python version required in the* ***README*** *file.*
- The initial version of the repository included two licenses (APACHE and MIT) and let users pick one of them. Authors justified it because of the "issue German universities have due to *Beihilferecht*. If things have commercial value, we are not allowed to share them outside university unless a market price is given". *Note: Authors chose Mozilla Public License Version 2.0.*
- It helps to add estimated execution time for time-consuming steps. I am referring in particular to build the image since it took about 20 minutes to complete.
- Last but not least, this reproducible report does not reproduce any figures, maps or experiments reported in the manuscript. However, I was able to set up an instance of the server described and ran some tests. Therefore, to a certain extent, this reproducibility report demonstrates the viability and functioning of the system presented in the manuscript. *Note: Authors improved the* ***README*** *file by adding more details about generated output and instructions to test the system. For example, they included a python snippet to convert the resulting Arrow file into a Pandas Dataframe for visualisation purpose.*