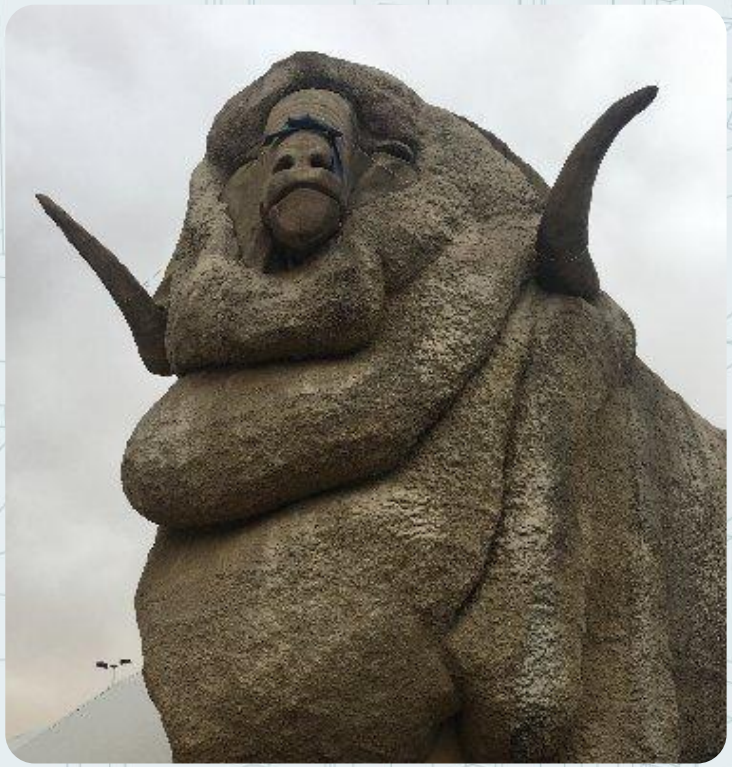
The Survey of Vectortile techniques: Static vs Dynamic

FOSS4G 2023 Prizren

MIERUNE | IGUCHI Kanahiro

https://docs.google.com/presentation/d/1_94vez5hCU7n9ZFICXfqjugOmix3G2puL_DtvqphU94/edit?usp=sharing

About me



GIS/OSS Enthusiast

IGUCHI Kanahiro

CTO at MIERUNE Inc.

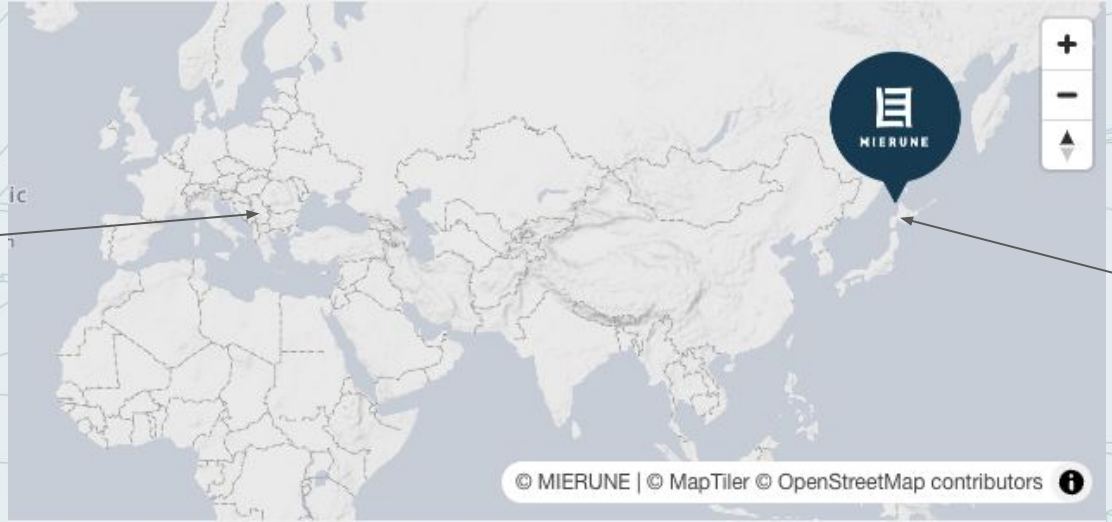
MapLibre User Group Japan

https://github.com/Kanahiro

MIERUNE is a tech company in Japan, specialized in Geospatial technologies.

MIERUNE means “It’s visible!” in Japanese.

About us



Prizren, Kosovo

Sapporo,
Hokkaido,
Japan

MIERUNE

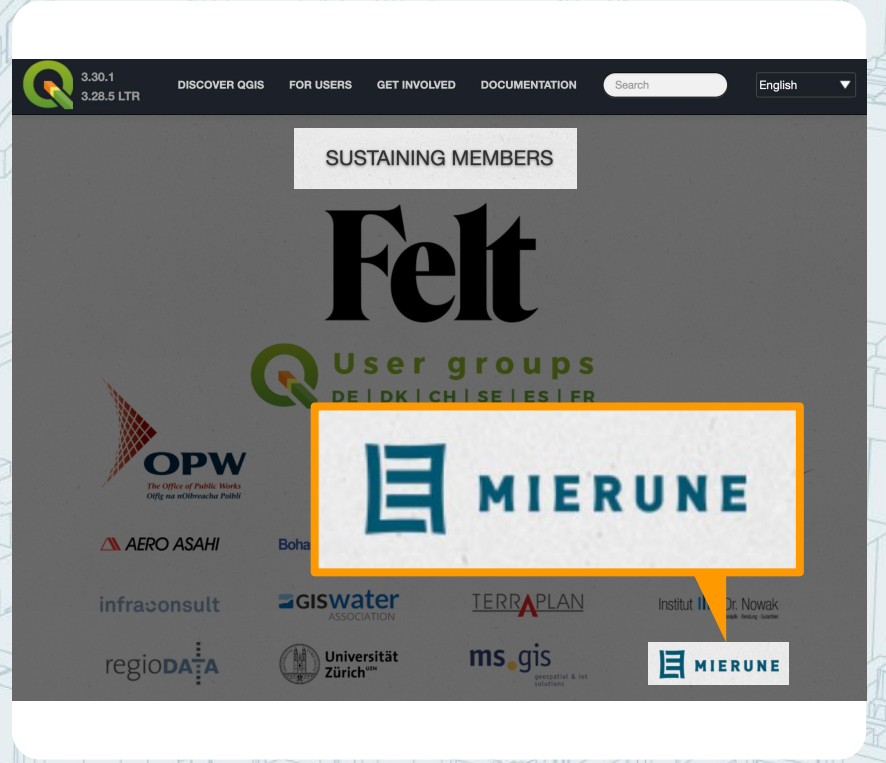
MIERUNE supports Communities

OSGeo・JP

QGIS

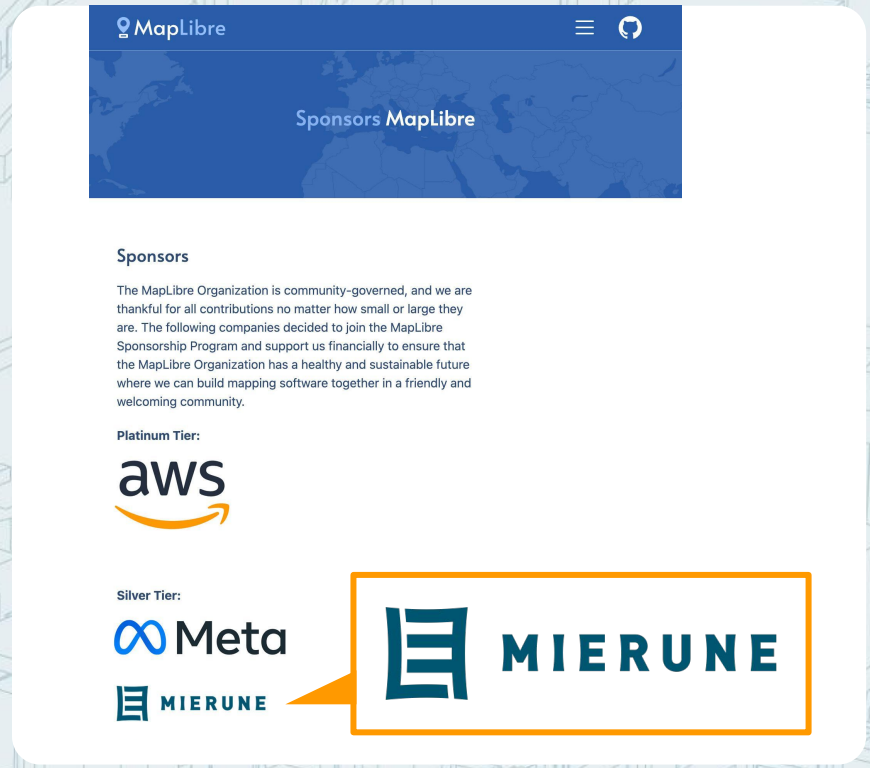
MapLibre

QGIS.org



https://qgis.org/en/site/

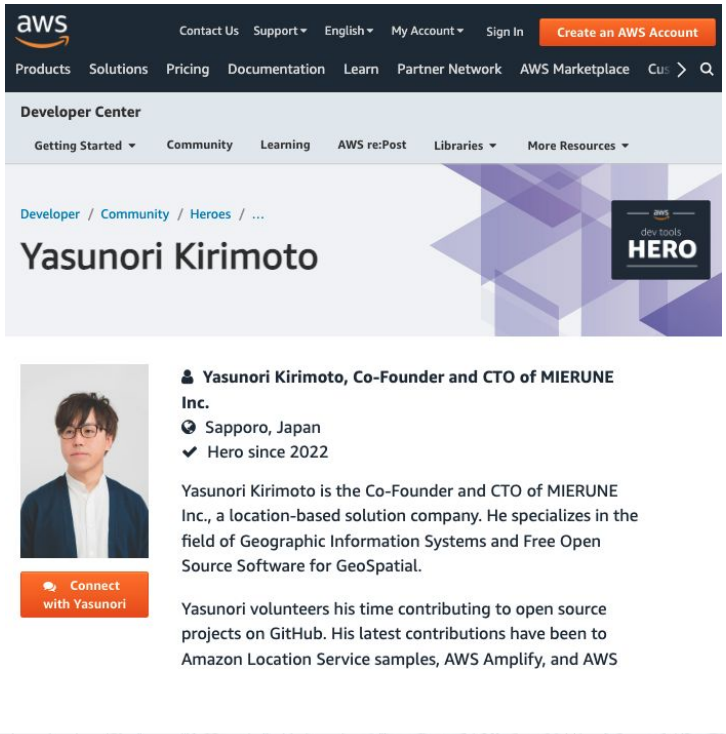
MapLibre Organization



https://maplibre.org/sponsors/

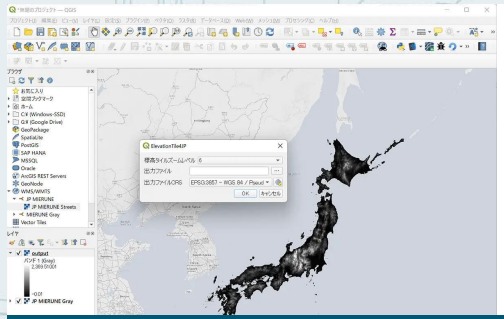
Community

aws HEROES

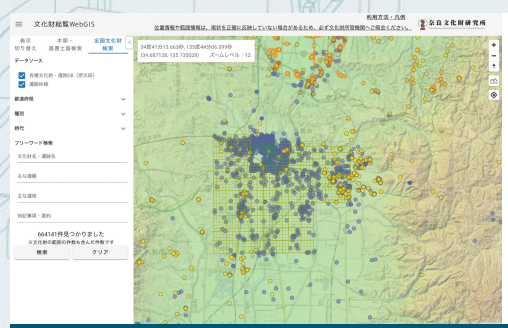


https://aws.amazon.com/developer/community/heroes/yasunori-kirimoto/?nc1=h_ls

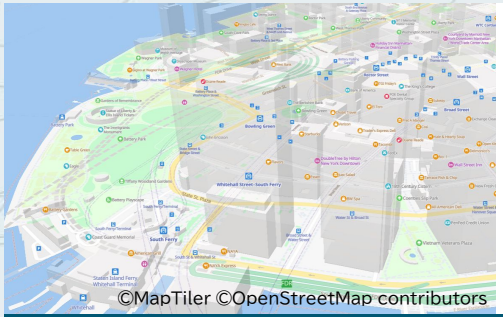
Solutions



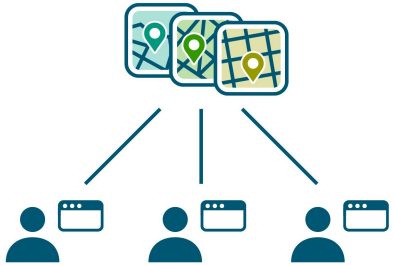
QGIS



WebGIS

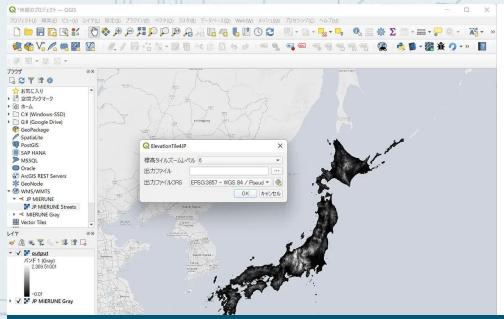


MapTiler



MIERUNE BASE

Solutions

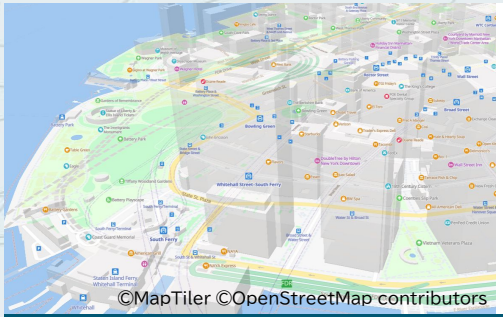


QGIS

QGIS

- Plugin Develop
- Consulting
- Seminar

WebGIS



maptiler

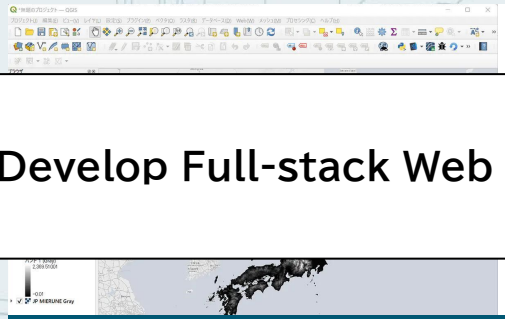
MapTiler

MIERUNE BASE

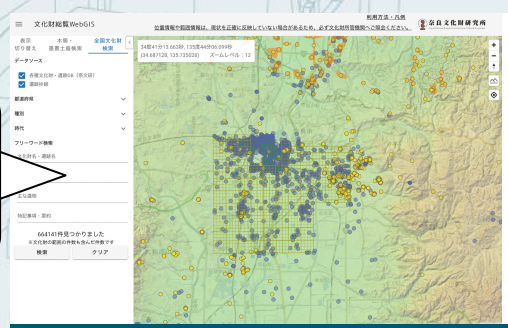
MIERUNE BASE

Solutions

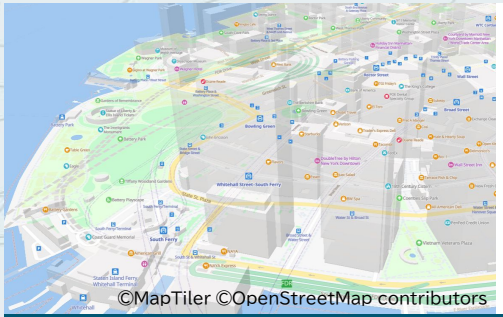
- Develop Full-stack Web GIS Apps



QGIS

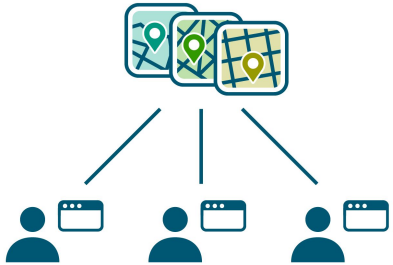


WebGIS



maptiler

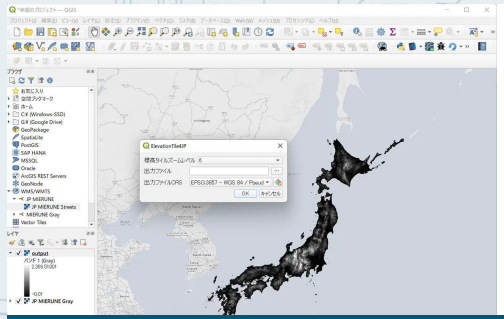
MapTiler



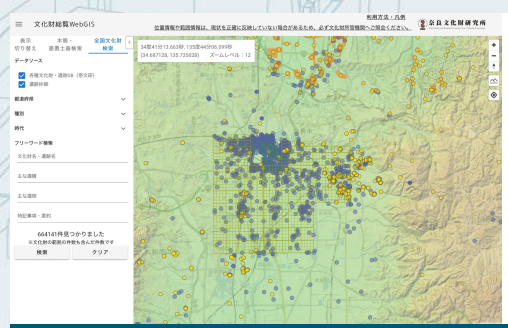
MIERUNE BASE

MIERUNE BASE

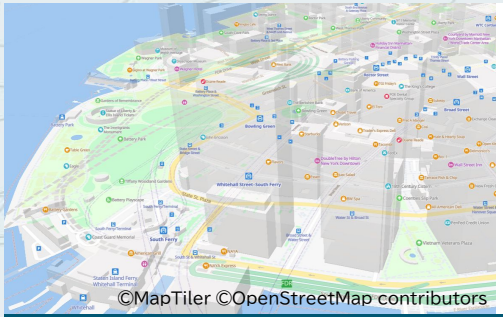
Solutions



QGIS



WebGIS

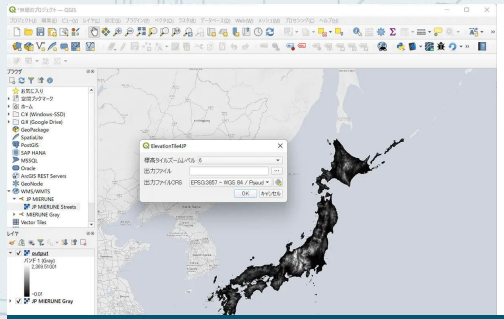


MapTiler

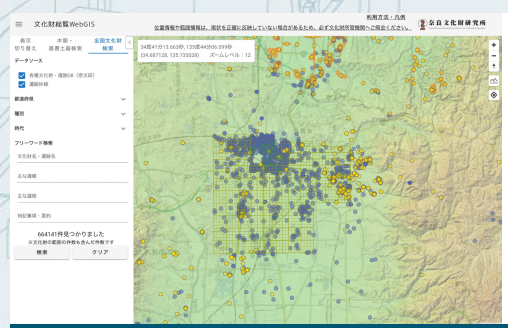
- As an agent in Japan, MIERUNE provide supports for MapTiler products for customers in Japan
- Create beautifully styled maps

MIERUNE BASE

Solutions



QGIS



WebGIS

- Geospatial SaaS
- Store, edit and serve your GIS data to/in/from cloud.

MapTiler

MIERUNE BASE

MIERUNE BASE

Change the World, with Geospatial.

Contents

MIERUNE

MIERUNE

01

Vectortile Techniques

Vectortile Techniques

- Vectortile techniques are grown in an ecosystem around of the specification: **MapboxVectorTile(MVT).**
- At first, Mapbox developed the spec and implementations - renderer, converter or some utilities.
  - Mapbox GL JS / Native and plugins
  - tippecanoe
  - geojson-vt…

Vectortile Techniques

MIERUNE

- Now the ecosystem have expanded beyond Mapbox organization and the vectortile techniques are widely used in FOSS4G.

Vectortile Techniques

MIERUNE

- In this talk,
  - I’ ll introduce softwares regarding to vectortile techniques
  - mainly about producing and serving tiles

Vectortile Techniques

- There are many softwares can be used for production.
- Introduce them, classifying three types:
  - producing tiles from your datasets
  - producing tiles from OpenStreetMap data
  - serving tiles

MIERUNE

Producing tiles from your datasets

- tippecanoe (felt/tippecanoe, mapbox/tippecanoe)
  - developed under Mapbox at first but now under Felt.com
  - good performance, simple interface and dependencies
  - very nice default settings and many options for specific usecases
  - de-facto standard in this category
- GDAL/OGR
  - supports MVT as a vector driver and can convert data into MVT
- PostGIS
  - support converting geometry into MVT in function ST_AsMVT
  - this function is fast enough to serve tiles dynamically.

MIERUNE

Producing tiles from OpenStreetMap

- OpenMapTiles(openmaptiles/openmaptiles)
  - opened the door of this category
  - parsing osm.pbf into PostGIS by pre-defined schema, then output as MVT
  - it may take some days for converting planet data
- Tilemaker(systemed/tilemaker)
  - references OMT schema
  - very fast converting than OMT
  - easy to use and customize
- Planetiler(onthegomap/planetiler)
  - references OMT schema
  - very fast converting than OMT: **for planet inside one-hour!!**
  - easy to use and customize
  - well documented and ready for production even planet scale

MIERUNE

Serving tiles

- Tileserver.gl (maptiler/tileserver-gl)
  - serves tiles from MBTiles
  - can serve raster tiles as styled images by server-side rendering
- **PMTiles (protomaps/PMTiles)**
  - cloud-optimzed file format
  - includes all tile data in one file and clients can get each tile by random access via HTTP Range-Request
  - similar to MBTiles but in this format you need no server implementation to serve tiles and well-compressed repeated-tiles.

Vectortile Techniques

- The cost for producing and serving vectortiles have been dramatically reduced by:
  - tippecanoe - de-facto standard converter
  - Planetiler - everyone can make OSM tiles at low cost
  - PMTiles - make the cost for hosting tiles much lower.
- Now you can utilize vectortiles in many situations at tiny cost!
- However, these techniques mainly focus on **“static” tiling.**

02

Static vs Dynamic

Static vs Dynamic

- **Static tiling** is one of the best way to serve any geospatial dataset because:
  - You can pre-process all tiles from a dataset
  - For serving tiles, you need only web-server or object-storage, these need not much cost.
  - A performance in serving is very good.

Static vs Dynamic

- Static tiling has a big problem cannot be solved in static approach: serving frequently updated dataset
  - once origin dataset updated, you have to re-process tiling (at worst entire tiles).
  - the process takes unacceptable costs when:
    - update is frequent
    - but you have to serve newest data.
- This is a typical situation we often encounter.

So, we need to consider “dynamic” tiling for next.

MIERUNE

03

Dynamic Tiling

Dynamic Tiling

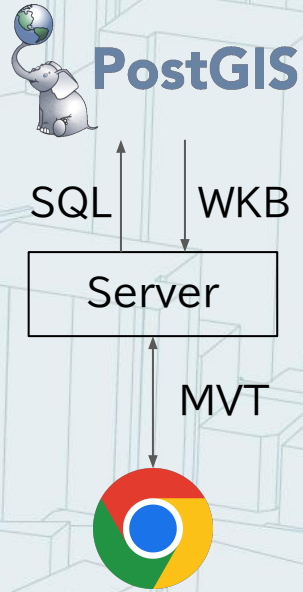
- Of course not only me think about dynamic tiling.
- There are some implementations serving tiles from PostGIS dynamically.
- They can be classified into two types:
  - based on ST_AsMVT
  - or not (native conversion)

Dynamic Tiling: ST_AsMVT

- ST_AsMVT is a function provided by PostGIS.
- It can be used for converting geometry into MVT binary.
- First implementation was in 2018 and improved many times. A performance have been improved.

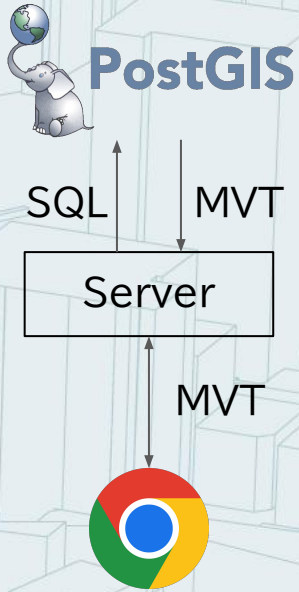
Native Conversion

- t-rex-tileserver/t-rex
  - written in Rust
  - early adapter in dynamic tiling
- go-spatial/tegola
  - written in Go
  - native converting as default but seems to support ST_AsMVT as optional



with ST_AsMVT

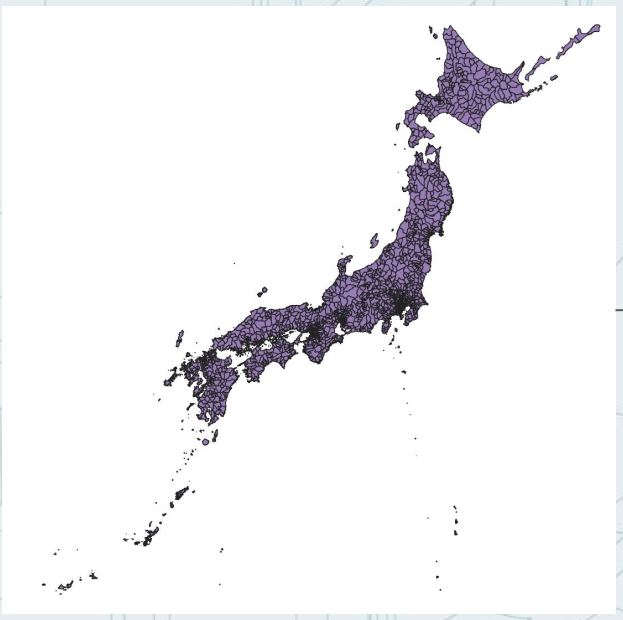
- CrunchyData/pg_tileserv
  - written in Go
  - easy to use with zero-config
  - customizable options
- **maplibre/martin (urbica/martin)**
  - written in Rust
  - easy to use with zero-config
  - customizable options
  - supports MBTiles/PMTiles
  - dynamic sprite for MapLibre Style Spec
- developmentseed/tipg(timvt)
  - built on FastAPI(Python), similar concept to titiler (server for COG)



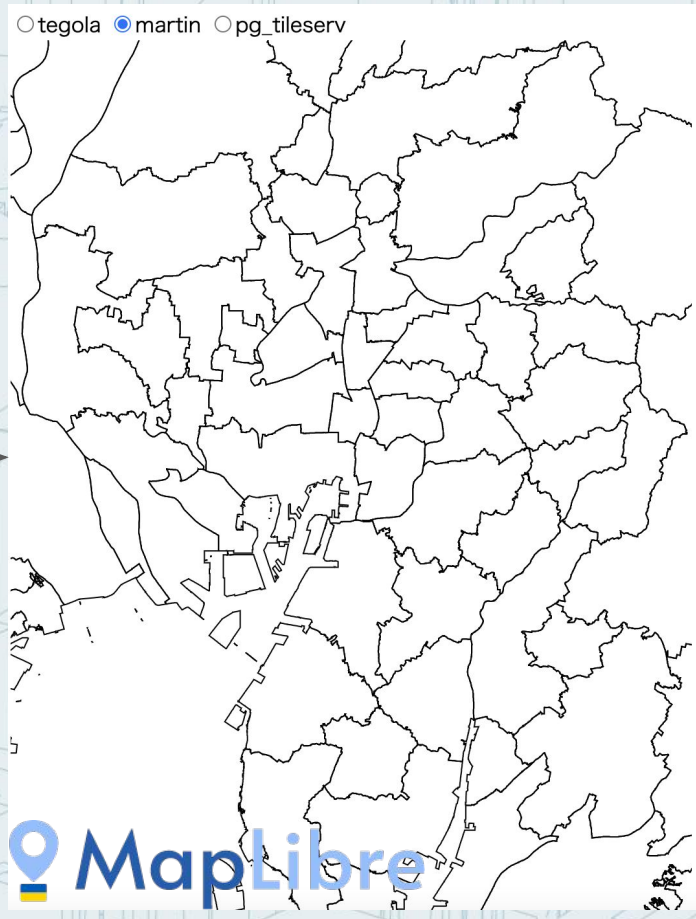
Dynamic Tiling:demo

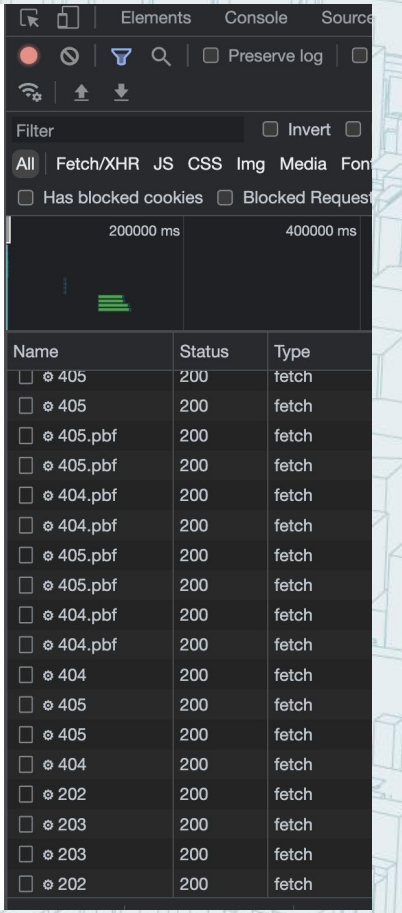
- demo
  - built test servers on docker compose on AWS EC2
    - PostGIS
    - tegola
    - martin
    - pg_tileserv
  - request tiles from clients: MapLibre GL JS
  - data: admin boundary in Japan with 120k polygons, 700MB(geojson)
  - codes:
    - https://github.com/Kanahiro/postgis-mvt-servers-demo

Dynamic Tiling:demo



admin boundary in Japan with
120k polygons





Dynamic Tiling:demo insight

- (not strict metrics, including network latency) average response time at zoom=10:
  - martin: 600ms+
  - pg_tileserv: 2sec+
  - tegola: 30sec+
- **ST_AsMVT is much faster than native converting!**
  - martin looks the fastest and enough to serve tiles dynamically!
- lower zoom means more features
  - more cost for tiling calculation and network
  - it is important to adjust/limit zoomlevel

Dynamic Tiling: pros cons

- comparing dynamic to static…
  - pros
    - can serve newest data equals to in DB
    - needs no tiling pipeline
  - cons
    - needs much larger computing cost on request
      - need fight to scalability
    - needs other softwares and more complexity
      - application server, caching…

Dynamic Tiling: Next ideas…

- avoid highly dependent on PostgreSQL computing.
  - newer implementations depend on ST AsMVT. As demo indicates, this function has a great performance!
  - However we should keep in mind the calculation(WKB -> MVT) is done on DB. This brings a high load on DB server.
  - It could be better to run such calculation on other cheaper, and more scalable runtime like AWS Lambda, ECS…
- wise caching
  - caching is necessary even in dynamic tiling but caching simply with only TTL might not be the best solution. Ideal way is to always use cached tiles unless a content of the tile modified.

Conclusion

- Vectortile techniques are matured in “static” tiling but not in “dynamic”.
- Dynamic tiling is necessary for serving newest data.
- There are some great production-ready implementations even in dynamic tiling.