Rendering maps without Database

Thomas Skowron
Previously

(if you happen to speak German)

„Überblick über Rendering-Techniken und Software“
FOSSGIS-Konferenz 2017

„Pipelinebasierte Erzeugung von Karten“
FOSSGIS-Konferenz 2018
Turning OSM Data into a graphical map
Bitmap/Vector Tiles,
Maps with larger extent, ...
State of the art
PostgreSQL

Do we really need all of that stuff?

+ SQL
+ ACID, MVCC & transactions
+ Indexes
+ Role permission management
+ scriptable
+ fail over
+ ...

- Performance (PostGIS)
- operational cost
- memory consumption

Do we really need all of that stuff?
Attempts to improve the situation...
Pre-rendered Vector Tiles

All features already baked in, flexibility mostly gone

Client Based Renderer
Alternative Approaches
tippecanoe
Clever features to keep vector tiles small
has a gazillion of options
still very limited to filtering
Tilemaker
flexibility through lua scripting
not scalable to larger extracts
But why does one tool need to do everything?
Generally, we are all doing almost the same stuff.
Step 1

Convert OSM data into geo data
Step 2
Filter
Step 3

Transform/map data
Step 4

Convert into target format
Suggestion:

parse | map-reduce | render
But how?
With **Tools**, which each do one thing well
and a portable data format
Let’s do Shapefiles!
Let’s do Shapefiles!
Let’s do OSMPBF!
Let’s do OSMPBF!
What does a suitable file format need?
Performance

linear writes, parallelizable reads
Scalable
small to huge data sets
Tag structures

No tables no more!
Future proof
adaptable to change
Shapefile

Performance moderate
Scalable no, 2 GB size limit
Tag Structure no
Future Proof no
GeoJSON

**Performance**  moderate

**Scalable**  moderately, single threaded

**Tag Structure**  yes

**Future Proof**  limited
GeoPackage

Performance  bad (SQLite)
Scalable  moderately
Tag Structure  yes
Future Proof  yes
We need something new
There is no progress without change
How would a new file format look like?

- binary
- blocks, streamable
- single stream, not multiple files
- not SQLite
- not overly obscure
- open and extendable
Suggestion
Based on
Protocol Buffers and WKB
Open Spec on https://thomas.skowron.eu/spaten/
Reference implementation in Go
github.com/thomersch/grandine/lib/spaten
Around 50% smaller than GeoJSON*

* YMMV
Version 0
Feedback and Ideas are welcome
What could we do with it?
grandine-spatialize -in planet.osm.pbf -mapping roads.yml |
grandine-tiler -out tiles/roads/ -zoom 14
osmium export -f spaten planet.osm.pbf |
gradine-converter -mapping roads.yml |
grandine-tiler -out tiles/roads/ -zoom 14

(not yet)
osmium export -f spaten planet.osm.pbf | your-tool-here -fancify | magic-renderer

(not yet)
Interchangeable tools
Future
Greater flexibility with less programming work
Faster processing with less hardware
Less points of failure
There is still lots to do

*Data format, tools, markup, …*
Let’s build the future together!
And now let’s discuss!