The Move from Raster Tiles to Vector Tiles

Two Architectures for Making Lots of Maps Dynamically
Outline

- Who We Are & What We Do
- Data Maps - Raster-based Architecture
- Categorizations - Vector-based Architecture
- Raster vs Vector Trade-Offs
Empower Engine

- The mapping platform solution for Democrats and other Progressive organizations.
- Co-founded by 2 in 2013, now a team of 5.
- I have two degrees in CS and have been writing campaign software since 2003. I have been working in geospatial since 2013 (our founding).
- Our data maps were used nation-wide in Canada in Justin Trudeau’s October landslide.
Campaigns Organize Geographically

- Where voters live is key
- Organizers each have their own turf
- Volunteers work best with their neighbors
- Seeing data on maps is motivating
First Product: Data Maps

Make tons of choropleth maps.
Rescale colors each for user.
Local variation matters.
Second Product: Categorizations

Dynamically categorize all districts within an arbitrary boundary

Optionally look at related data on each category as you assign things
<table>
<thead>
<tr>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raster</strong> - A set of horizontal lines composed of individual pixels, used to form an image on a CRT or other screen.</td>
</tr>
<tr>
<td><strong>UTF</strong> - Unicode Transformation Format</td>
</tr>
<tr>
<td><strong>Vector</strong> - An array of data ordered such that individual items can be located with a single index or subscript.</td>
</tr>
<tr>
<td><strong>Raster Tiles</strong> = <strong>Image Tiles</strong> - JPEG images of map data</td>
</tr>
<tr>
<td><strong>UTF Grid Tiles</strong> - Json-encoded tiled data that can be used with raster tiles for interactivity</td>
</tr>
<tr>
<td><strong>Vector Tiles</strong> - Binary encoding of tiled data</td>
</tr>
<tr>
<td><strong>MVT</strong> = Mapbox Vector Tile is most efficient &amp; most standard</td>
</tr>
</tbody>
</table>
Outline

● Who We Are & What We Do
● Data Maps - Raster-based Architecture
● Categorizations - Vector-based Architecture
● Raster vs Vector Trade-Offs
Example: Ethnicity Plurality

Ethnicity Plurality

Census Blocks are colored according to the race/ethnicity with the greatest share of the population in that block. Darker color indicates greater share. Lighter colored blocks are likely to be more diverse than darker blocks. Note that percentage does not reflect overall population. Hover over a block to compare the percentage based colors to population counts.

<table>
<thead>
<tr>
<th>Census Block</th>
<th>Block 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>White Population</td>
<td>44</td>
</tr>
<tr>
<td>Black Population</td>
<td>30</td>
</tr>
<tr>
<td>Latino Population</td>
<td>8</td>
</tr>
<tr>
<td>Asian and Pacific Islander Population</td>
<td>28</td>
</tr>
<tr>
<td>Other Combined Population</td>
<td>4</td>
</tr>
</tbody>
</table>
Example: Model Score
Example: Penetration

2014 OFW: % of Doors Attempted

Percent of door targets attempted in the 45th Legislative District by Organizing for Washington (2014 Coordinated Campaign)

<table>
<thead>
<tr>
<th>Precinct</th>
<th>ELSIE</th>
</tr>
</thead>
<tbody>
<tr>
<td># in Universe</td>
<td>155</td>
</tr>
<tr>
<td># Attempted</td>
<td>117</td>
</tr>
<tr>
<td># in Universe Per Sq Mile</td>
<td>546.05</td>
</tr>
<tr>
<td># Attempted Per Sq Mile</td>
<td>414.37</td>
</tr>
<tr>
<td>% Attempted</td>
<td>75.48</td>
</tr>
<tr>
<td>Sq. Miles</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Dataset: Universe Tiers 1 and 2
Penetration: Door Attempts Since 2014-08-06
Within Legislative District 45
Boundary
Summary Stats
# in Universe 10,409
Example: Density of List

Identified Liberals

http://lpca.ca/geo.liberal.ca

# Voters Per Sq Km

1
31
52
84
127
2851

No district shown = No data

Dataset
IDed as Liberal

Within
59035 Vancouver

Boundary
East

Summary Stats

# Voters
1,134

Poll
59035:1360

# Voters
22

# Voters Per Sq Km
140.80

Sq. Kilometers
0.16

Poll
Current

Field
LPC/PLC

Search for Location
Technical Challenges

● Many data sets
  ○ Some static (e.g. census data), some aggregated and nightly updating (e.g. private field data)

● Tons of potential boundaries & granularities
  ○ Granularity - what set of districts is being colorized
  ○ Precincts/hexagons/blocks in CDs/Cities/Custom Districts

● Non-technical users make new maps in web request/response cycle

● Distribute maps to sets of users and rescale

● Update maps when data updates nightly
Architecture

- Render with Leaflet JS
- TileStache for custom image and UTF grid tiles
- Data stored in Postgres/PostGIS
- Cache tiles on S3 (built into TileStache)
- Cache table per map with shapes and data
  - TileStache does no joins and generates tiles from the smallest possible table
  - Compute color bins from data in cache table
  - Recreate table when data has been updated since map viewed
  - Update map version to invalidate the S3 tile cache
- Custom TileStache Wrapper for a dynamic config
  - Tile path is http://<tile server>/map_id/version/z/x/y
  - https://gist.github.com/JulieGoldberg/6926274
## Sample Map Cache Table Definition

<table>
<thead>
<tr>
<th>map_caches.map_2270</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>district_id</td>
</tr>
<tr>
<td>double</td>
<td>attribute_15230</td>
</tr>
<tr>
<td>double</td>
<td>attribute_15231</td>
</tr>
<tr>
<td>double</td>
<td>attribute_15232</td>
</tr>
<tr>
<td>geometry(MultiPolygon, 3857)</td>
<td>geometry</td>
</tr>
<tr>
<td>varchar(100)</td>
<td>long_name</td>
</tr>
<tr>
<td>integer</td>
<td>colorize_attribute_id</td>
</tr>
<tr>
<td>double</td>
<td>colorized_value</td>
</tr>
</tbody>
</table>
Make Map Cache Tables Efficiently

- Don’t do any geometric queries to determine what data will be on a map.
- Pre-calculate all possible granularities with all possible boundaries.
  - Census Tracts in Minneapolis? Here’s the list.
- When asked to pull any data layer in any boundary at any granularity, we join on indexed integer IDs.
- Cache table for each possible granularity, so the table we copy shapes from is as small as possible.
Cache tables can always be recreated....

Hi, this is your son's school. We're having some computer trouble. Oh, dear—did he break something? In a way—

Did you really name your son Robert'); DROP TABLE Students;-- ? Oh, yes. Little Bobby tables, we call him.

Well, we've lost this year's student records. I hope you're happy. And I hope you've learned to sanitize your database inputs.
Data Maps Live Demo....
Outline

- Who We Are & What We Do
- Data Maps - Raster-based Architecture
- Categorizations - Vector-based Architecture
- Raster vs Vector Trade-Offs
Example: Cook House Ratings
Example: Flag on District (PCO)
Example: Organizer Turf
Technical Challenges

● Live Editing of Map in Browser - Requires Vectors!
● Tons of potential boundaries & even more granularities than data maps
  ○ Precincts/Cities/Custom Districts in States/CDs/Cities/Custom Districts
● Non-technical users create and edit maps in web request/response cycle
Architecture

● Render with Mapbox GL
● Dynamic Edits saved with AngularJS
  ○ Send JSON back and forth and rerender just parts of the page
  ○ How single page apps are built
● Mapzen TileStache for dynamic MVT vector tiles
● Data stored in Postgres/PostGIS
● Cache tiles on S3 (built into TileStache)
● Custom TileStache Wrapper for a dynamic config
  ○ Tile path is http://<tile server>/granularity/z/x/y
  ○ https://gist.github.com/JulieGoldberg/6926274
Less Caching than for Data Maps

- District cache table & associated tile set per potential granularity
  - Many categorizations reuse the same cache table
  - JS filtering to only show districts on current map
- Still need to precompute all districts in all boundaries for efficient creation
Sometimes Data is Associated

- Associated data shown on bottom and summed.
- Unlike Data Maps, no binning based on data.
- Can send all the data on page load with district id.
- District tiles contain id.
- Mapbox GL will match it to shapes based on id.
Categorizations Live Demo....
Outline

● Who We Are & What We Do
● Data Maps - Raster-based Architecture
● Categorizations - Vector-based Architecture
● Raster vs Vector Trade-Offs
Ease of Generating Tiles

- Standard TileStache generates Image and UTF Grid tiles and has S3 caching out of the box.
- SpatialDev’s PGRestAPI generates MVT tiles, but it doesn’t cache or handle multiple DB schemas.
- MapBox server doesn’t generate dynamic tiles.
- Only Mapzen’s fork of TileStache (non-standard) generates MVT tiles.
Caching Required

- Dynamic maps mean we always pre-compute all possible districts in all possible boundaries.
- Tile creation should never involve joins or where clauses on big tables.
  - Vector tiles generated from cache table of all districts in specific group
  - All Iowa Precinct categorizations use the same cache table and same cached tiles
- Data maps’ cache table per map could go away if done with vector tiles.
  - We’d make a temp table to compute bin endpoints.
Computation on Browser

- Vectors allow dynamic styling!
- Vectors have fluid zoom levels.
- BUT it’s computationally intensive for the browser.
- Categorization list page is just text.

Static images could work too but are annoying to generate.

- We render 30 map previews for data maps list page.
Image & UTF Grid
Tiles: Data Maps

Vector Tiles: Categorizations
Conclusions

- For dynamic tiles, query against smallest possible database table
- Image tiles are more efficient to render on the browser
- Dynamic vector tiles are harder to generate
- Vector tiles are more flexible and reusable
Questions?

Contact: julie@empowerengine.com