https://is.gd/XIkcK6

Serving Dynamic Vector Tiles using ST_AsMVT()

paul.ramsey@crunchydata.ca

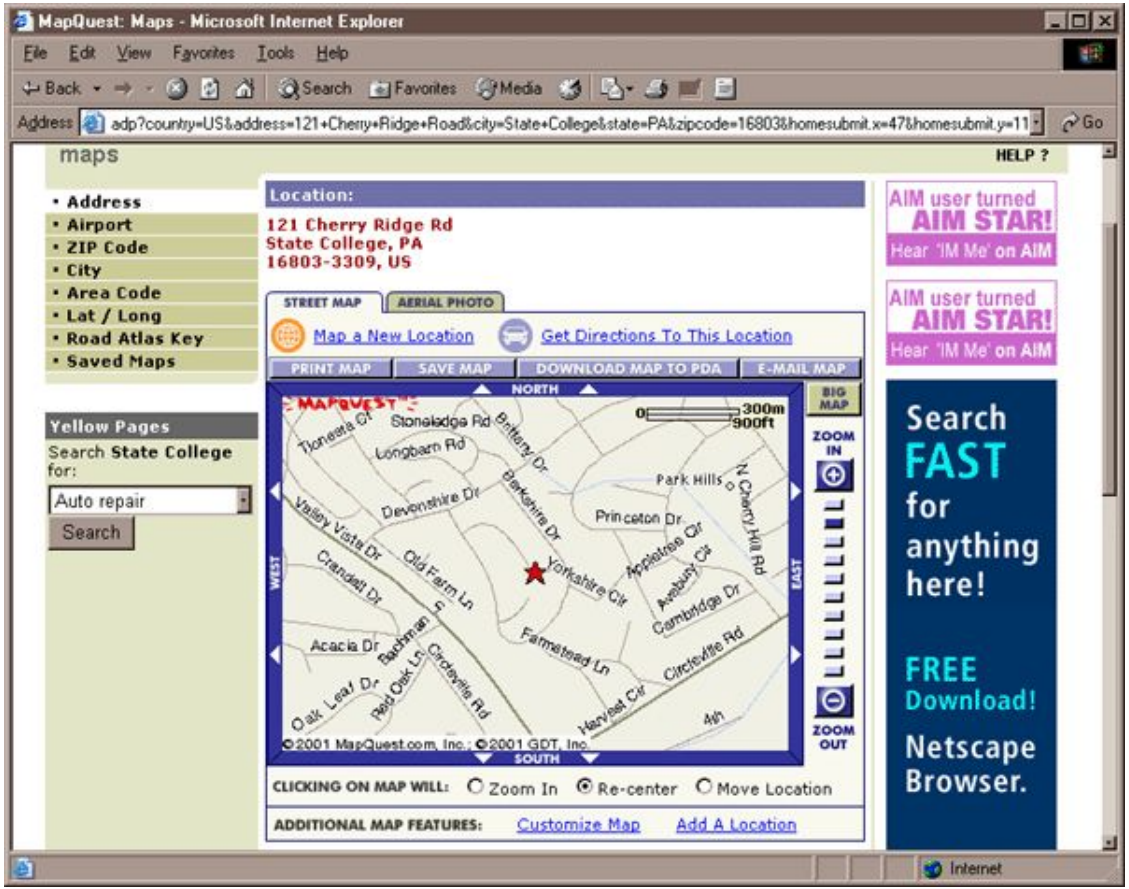
PostGIS Day STL - November 14, 2019

crunchy data

A Brief History of Map Tiles

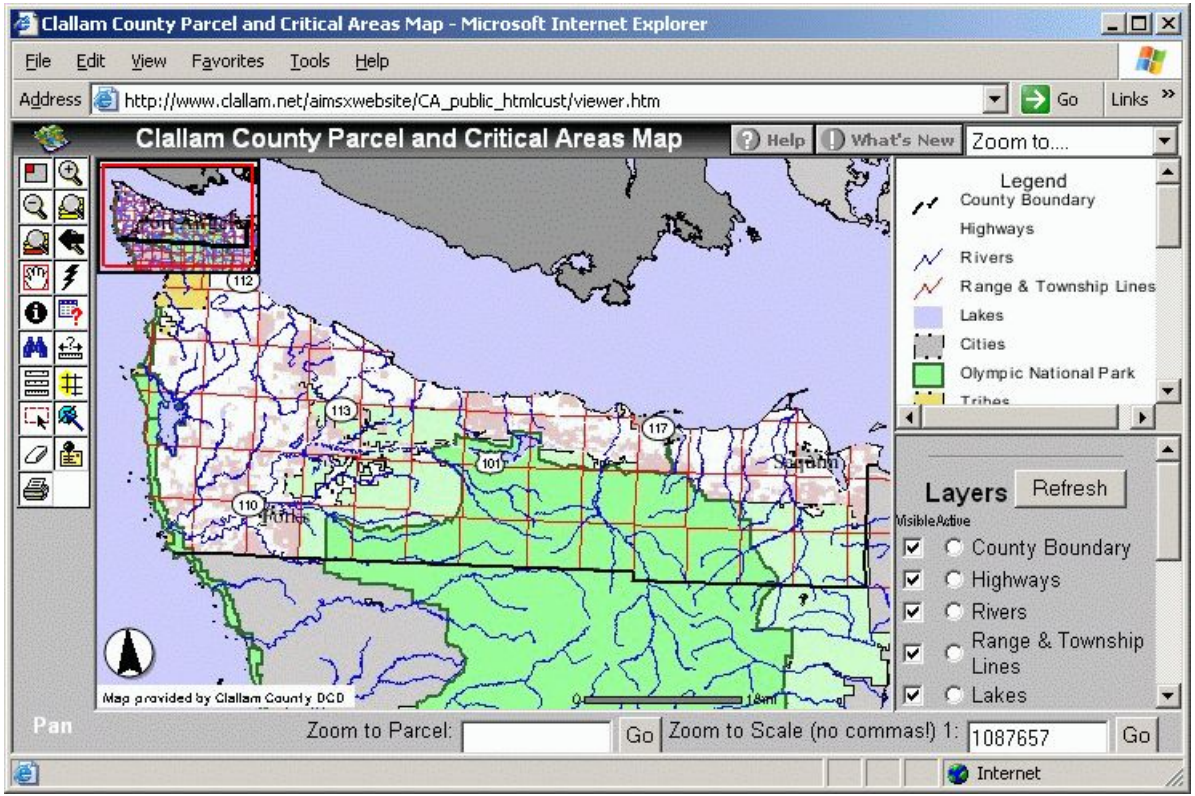
crunchydata

Pre-2000 - Static tiles, static HTML



- Computing cycles are expensive, server and client
- Bandwidth narrow
- Web mapping is just getting started
- Base map is large quantity of data to handle
- Render once as tiles
- Tiles are static and cacheable
- One tile at a time!

Circa 2000 - Dynamic map, mostly static HTML

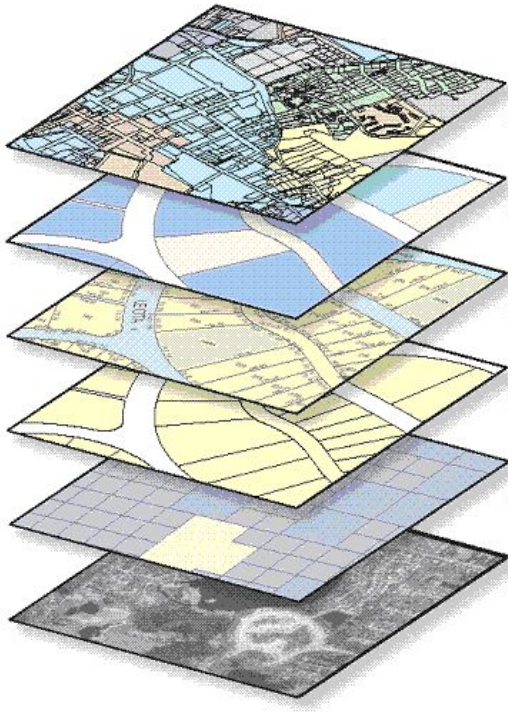


- ArcIMS and OGC WMS
- CPU cycles cheaper, some JS
- Dynamically render, suitable for custom data
- Low-usage, niche audiences
- Render all data into one map image
- Tiles are **not cacheable**

2005 - The Day the Earth got Slippy

- Heavier client-side JS
- View/load multiple tiles at a time
- Network becomes bottleneck
- Illusion of speed
- Tiles are cacheable
- Render base map once, add dynamic overlay (JS) on client

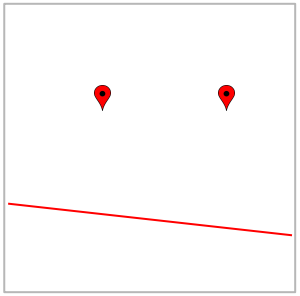
Before 2005:
A map has many layers



After 2005:
A map has **two** layers

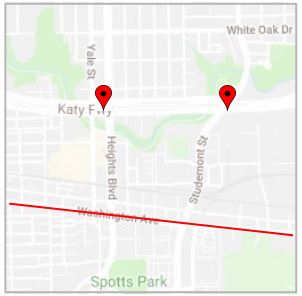
Base map

Static



Layer of interest

Dynamic



Composited on client

crunchy data

Modern Web Map Architecture

Base map

- Non-changing data
- Lots of layers composited
- Lots of data in each tile
- Generated in advance
- Served statically
- Highly cacheable
- Highly re-usable between applications

Layer of interest

- Changing data
- One or a handful of layers
- Very little data in each tile
- Generated on-the-fly
- Sometimes cacheable
- Unique to particular application

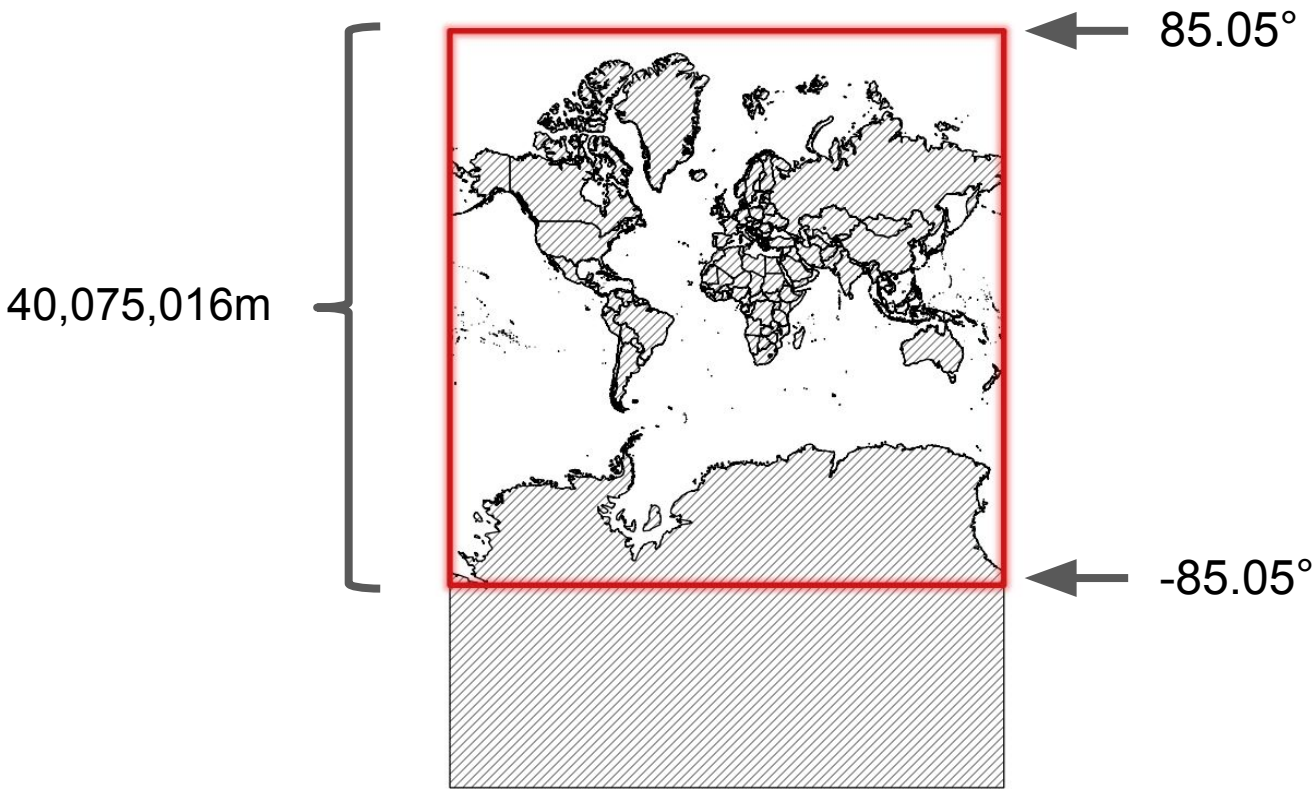
Map Tile Addressing

Tile Addressing

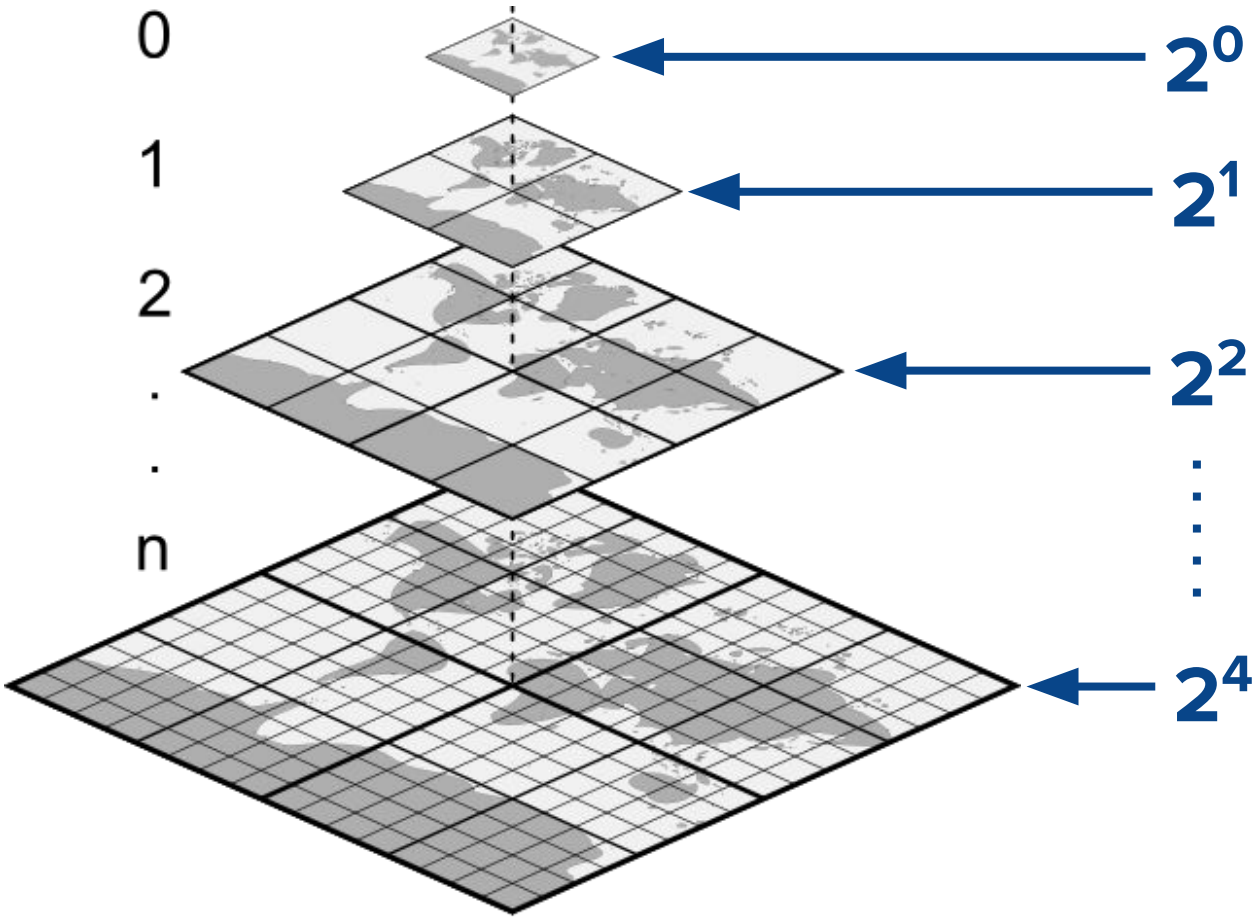
- Google / XYZ
  - http://tile.server/{z}/{x}/{y}.{ext}
  - http://tile.server/2/3/2.png
- Bing
  - http://tile.server/{L1}{L2}...{Ln}.png
  - http://tile.server/31.png
- OGC WMTS
  - http://tile.server/maps.cgi?service=WMTS&request=GetTile&version=1.0.0&layer=etopo2&style=default&format=image/png&TileMatrixSet=WholeWorld_CRS_84&TileMatrix=10m&TileRow=2&TileCol=3

Web Mercator (**EPSG:3857**, EPSG:900913)

"the **earth is** not just **flat**, it is also a **perfect square**"



Powers of Two



Assumptions

- Projection
  - Mercator
- Bounds (Mercator)
  - 20037508
  - -20037508
- Bounds (Geographic)
  - -180,180
  - -85.05,85.05

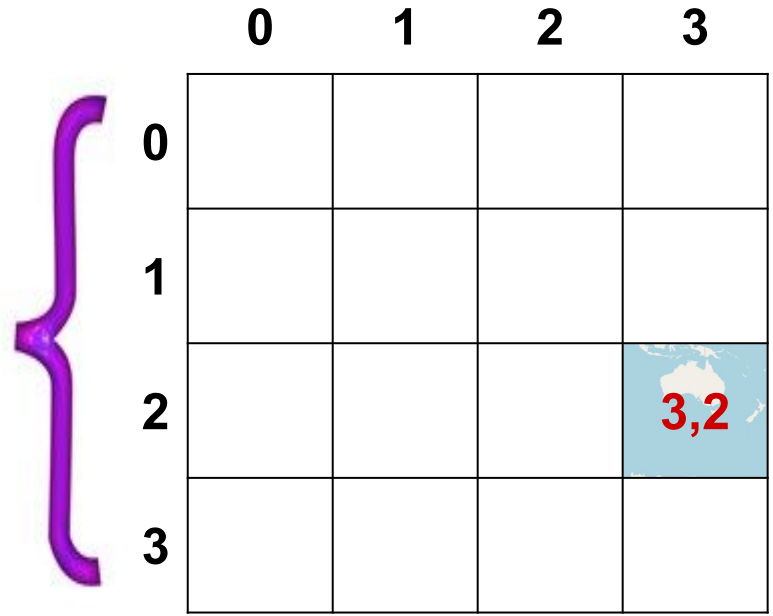
Powers of Two

	0	1	2	3
0			2,0	3,0
1			2,1	3,1
2				
3				

	0	1
0		1,0
1		

Tile Address to Geographic Bounds

- **Tile Address** (XYZ)
- with **Projection**
- with **Bounds**
- yields geographic extent



Mercator / EPSG:3857

Map Tile Formats

Image Formats

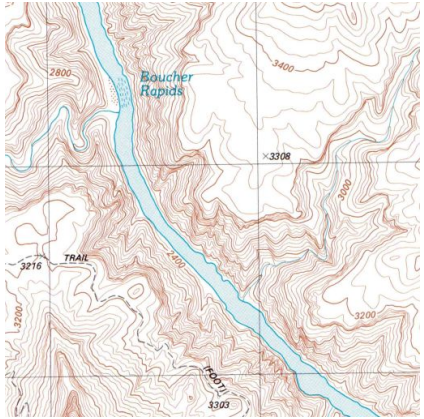
- Key goal is performance aka **Size**
  - Smaller is always better
- Google / XYZ
  - http://tile.server/{z}/{x}/{y}.{ext}
- PNG for cartographic maps
  - Lots of identical colors

- JPEG for imagery
  - Lots of color gradients

Image Resolution

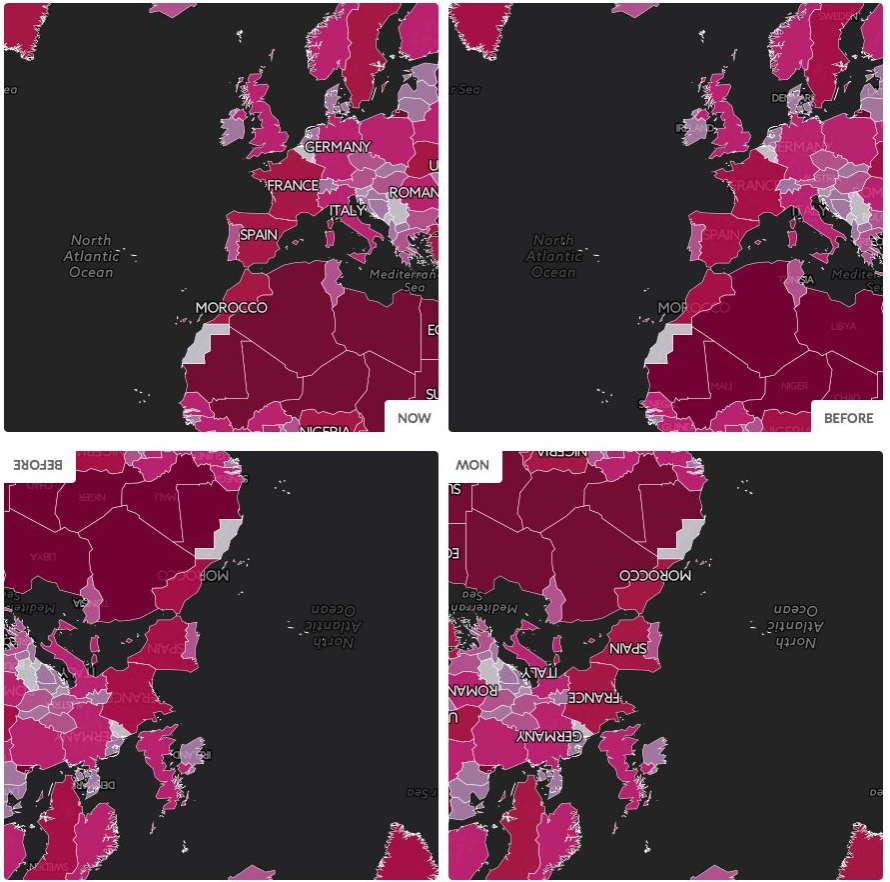
- Powers of two
  - 256 x 256

- Retina displays
  - 512 x 512



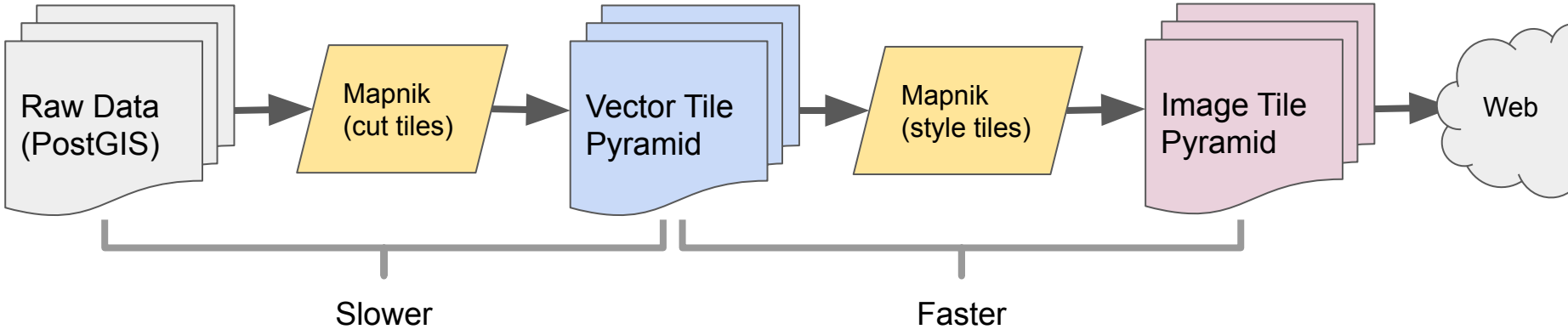
Limitations of Imagery

- More resolution requires more pixels
  - Size!
  - 512 x 512 = 262144 pixels
- To label or not to label!
  - Burned in labels conflict with other layers
  - Label sandwich more attractive but still conflict
- Labels are static and tied to one orientation
  - No map rotations allowed



Enter Vector Tiles

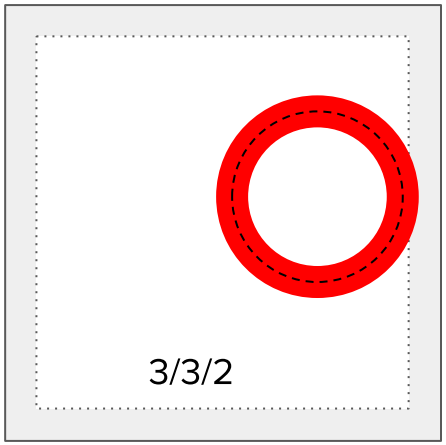
- ~2010 Google Android maps app starts using vectors
  - more map, less bandwidth
  - dynamic labelling
- ~2013 Google Maps web client starts using vectors
- ~2014 Mapbox publishes MVT specification
  - read/write implementation in Mapnik
  - server-side rendering!

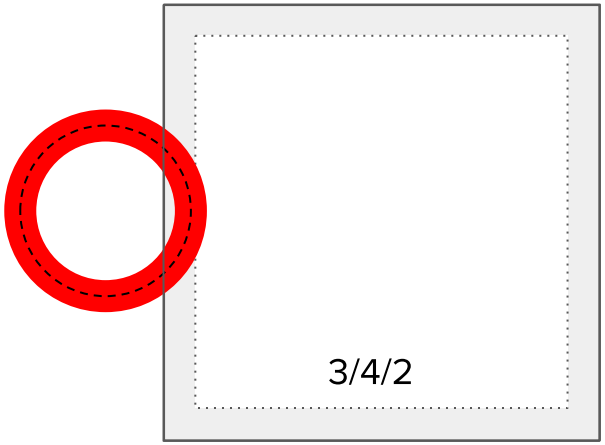


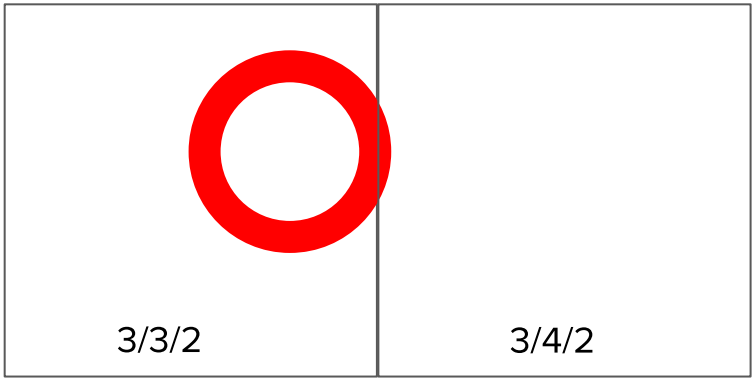
Baking a Vector Tile

- Tile request for 2/2/3.pbf (z/x/y.ext)
- Calculate data bounds
  - Tile bounds plus gutter
- Retrieve any features that intersect data bounds
- Simplify data to tile resolution
  - 4096 x 4096
- Quantize data to internal tile grid
  - Integer coordinates
- Confirm/repair validity of geometry
- Clip to data bounds
- Re-calculate coordinates using offsets
- Encode in protocol buffer format
  - Geometry **and** attribution
- Return tile PBF file

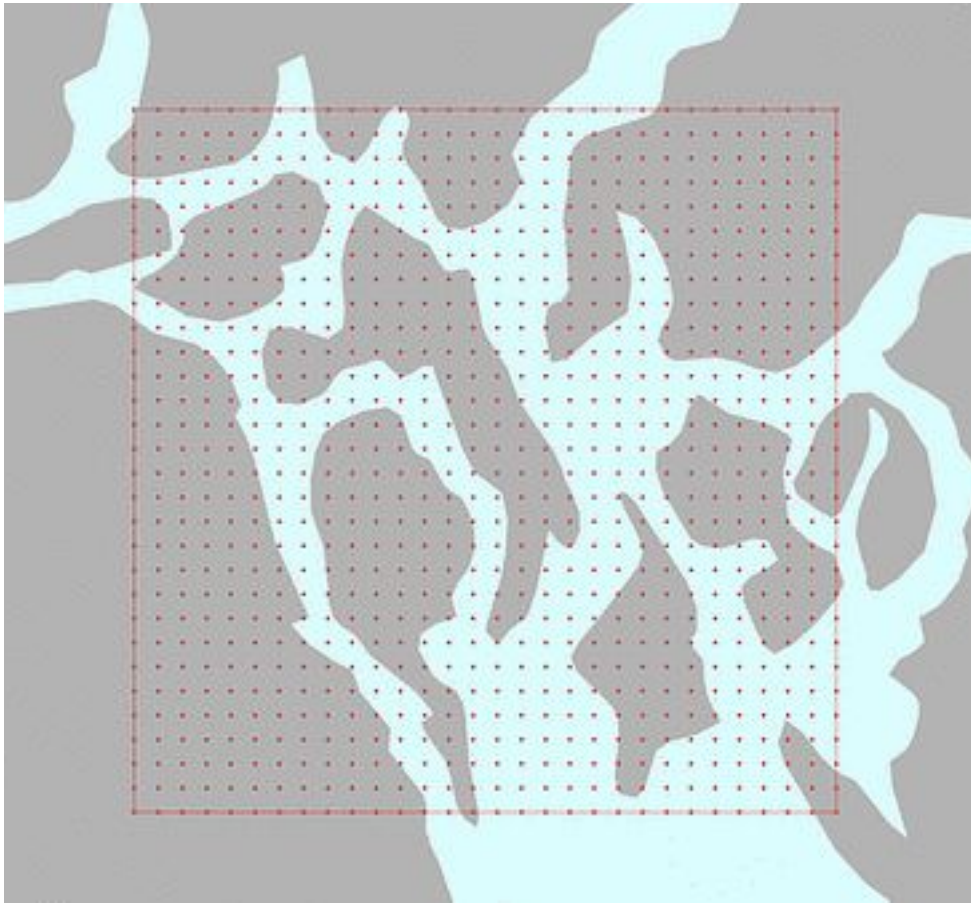
Bounds + Gutter







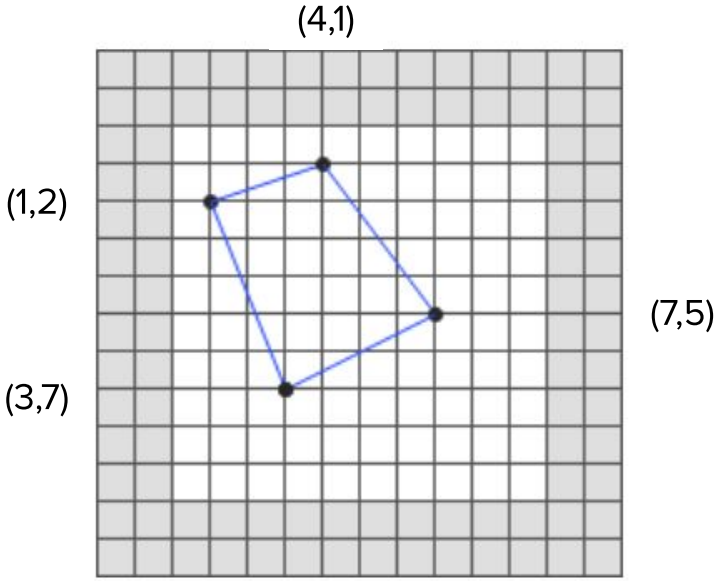
Simplify and Quantize



- Convert double coordinates to local integers
  - 8 byte doubles
  - 2 byte integers
- Remove unneeded coordinates
  - Co-linear coordinates
  - Near-duplicate coordinates
- Clip to gutter and confirm / repair validity of geometry

Produce MVT Tile Coordinates

- Enforce winding order
  - “Right hand rule”
  - CCW exterior rings
  - CW interior rings
- Convert coordinates into deltas
  - 1000 1000,
    1003 1004,
    1010 1000,
    1000 1000
  - 1000 1000
    +3 +4
    +7 -4
    -10 0




```
MoveTo(1,2)
LineTo(3,-1)
LineTo(3,4)
LineTo(-4,2)
ClosePath()
```

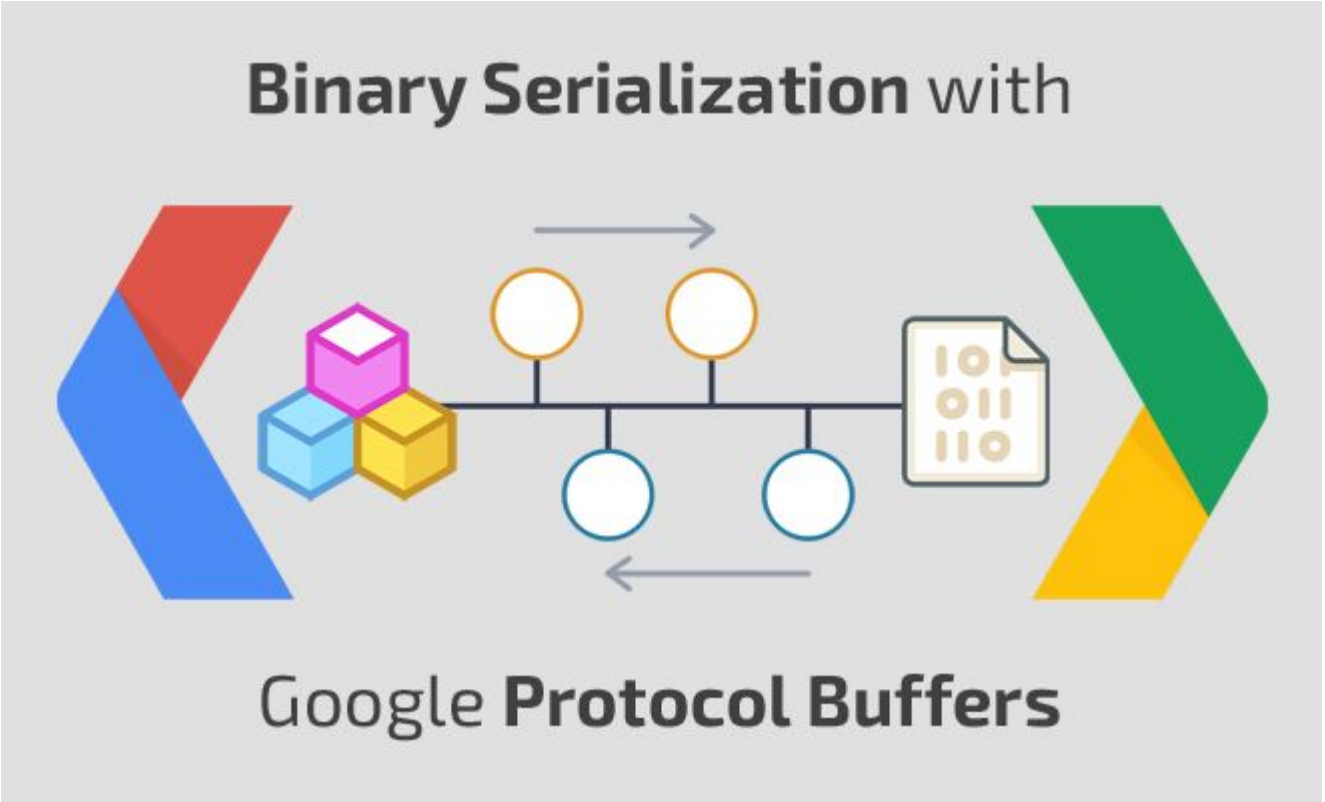
Produce MVT Attribute Dictionaries

```
{
 "type": "FeatureCollection",
 "features": [
  {
   "geometry": { ... },
   "type": "Feature",
   "properties": {
     "hello": "world",
     "h": "world",
     "count": 1.23
     }
  },
  {
   "geometry": { ... },
   "type": "Feature",
   "properties": {
     "hello": "again",
     "count": 2
     }
  }
  ]
}
```

```
layers {
  version: 2
  name: "points"
  features: {
    id: 1
    tags: [0, 0,
           1, 0,
           2, 1]
    type: Point
    geometry: ...
  }
  features {
    id: 2
    tags: [0, 2,
           2, 3]
    type: Point
    geometry: ...
  }
  keys: "hello" "h" "count"
  values: "world" 1.23 "again" 2
```

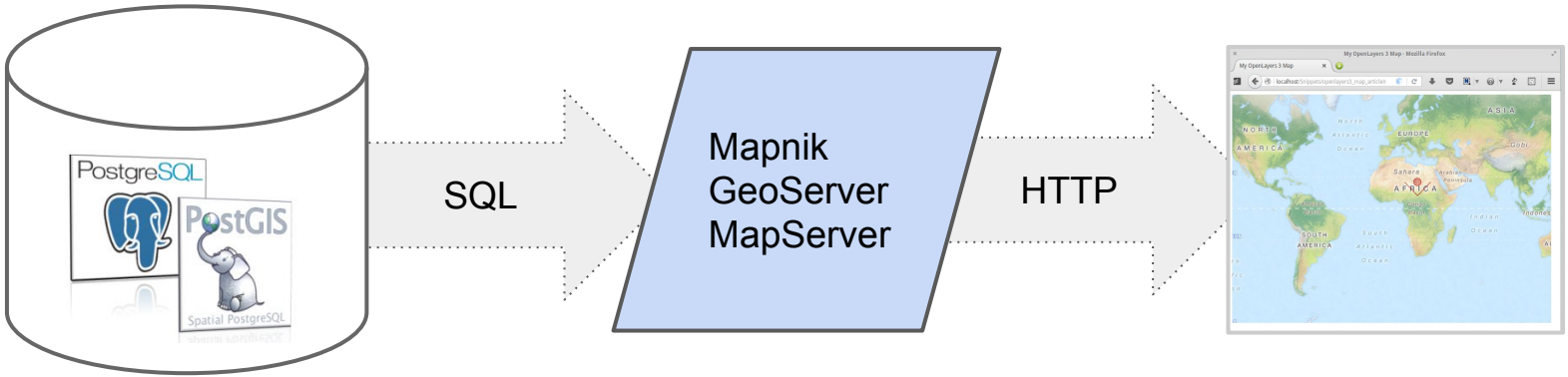
Google Protocol Buffers

- Language neutral (C, C++, Java, .Net, JS, etc)
- Platform neutral (Intel, ARM, RISC, etc)
- The dream of the 90's is still alive... (XDR format)



Dynamic Map Tile Serving

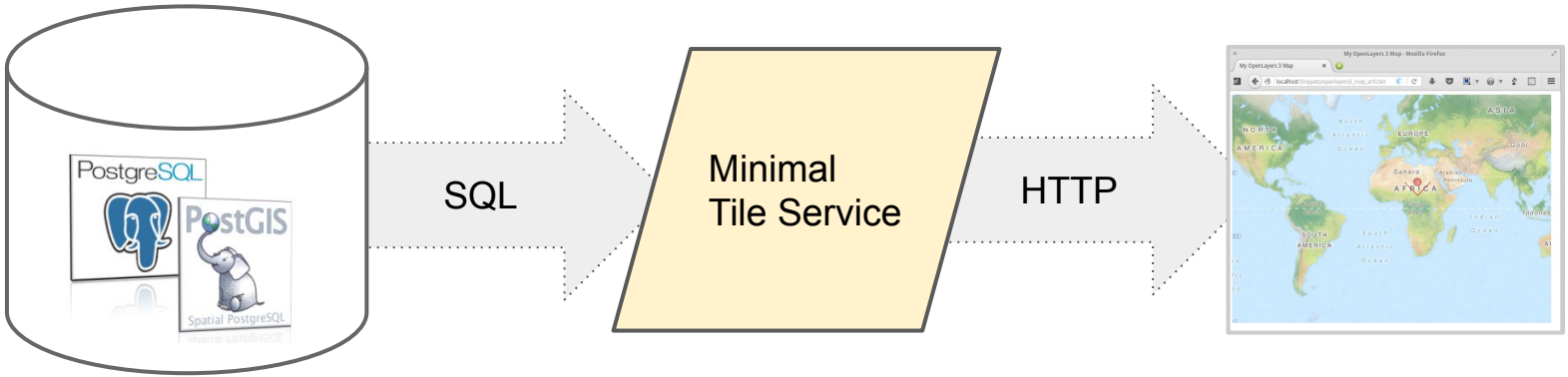
Just add (big, fat) middleware!



- Request features in tile (SQL)

- Req/Resp (HTTP)
- Simplify
- Quantize
- Clip
- Encode

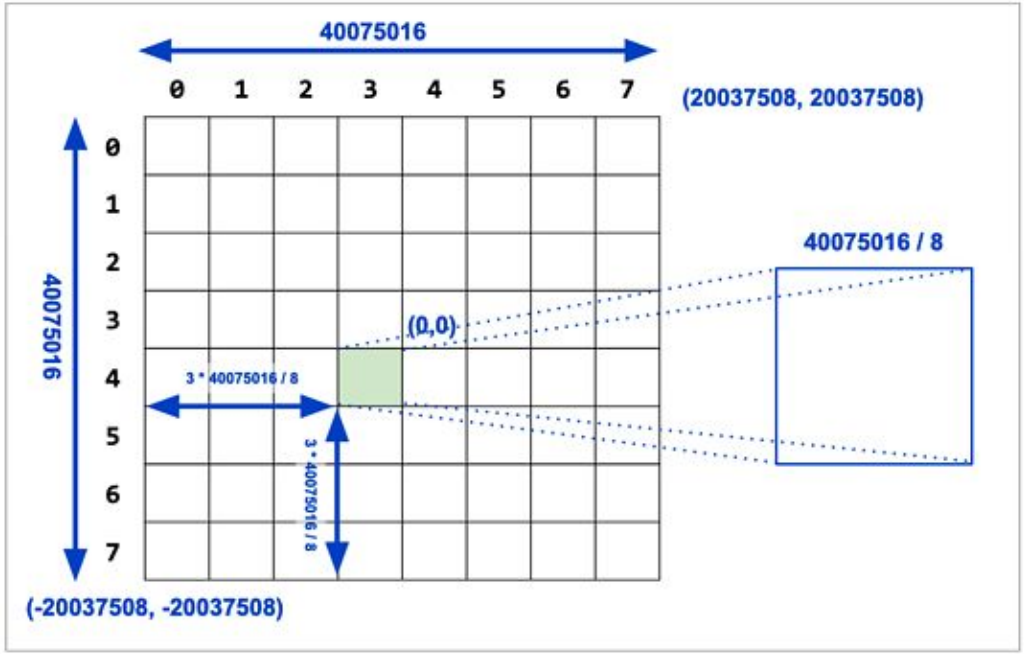
No, just add HTTP!



- (SQL)
  - Query
  - Simplify
  - Quantize
  - Clip
  - Encode
- Req/Resp (HTTP)

Query Construction

- Z/X/Y → Bounds
- eg, 3/3/4.pbf
- Zoom = 3
- X/Y = 3, 4

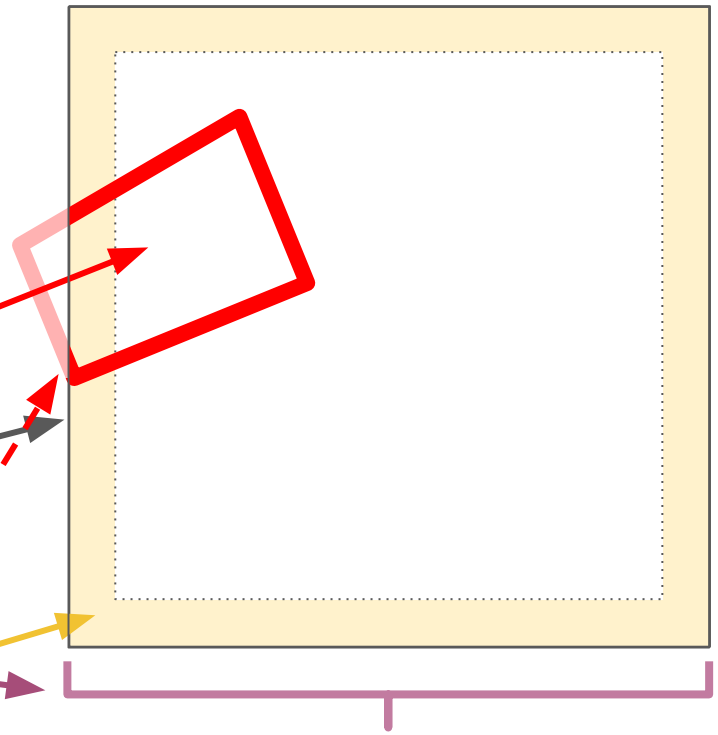



```
ST_TileEnvelope(
   integer tileZoom,
   integer tileX,
   integer tileY,
   geometry bounds=[mercator])
)
```

Geometry Preparation

- Simplify
- Quantize
- Clip

```
ST_AsMVTGeom(
   geometry geom,
   box2d bounds,
   integer extent=4096,
   integer buffer=256,
   boolean clip_geom=true
)
```



MVT Encoding

- Delta coordinates
- Attribute dictionaries
- Protobuf encoding

```
ST_AsMVT(
  row row,
  text layer_name="default",
  integer extent=4096,
  text geom_name="geom",
  text feature_id_name=NULL
)
```

All Together Now!

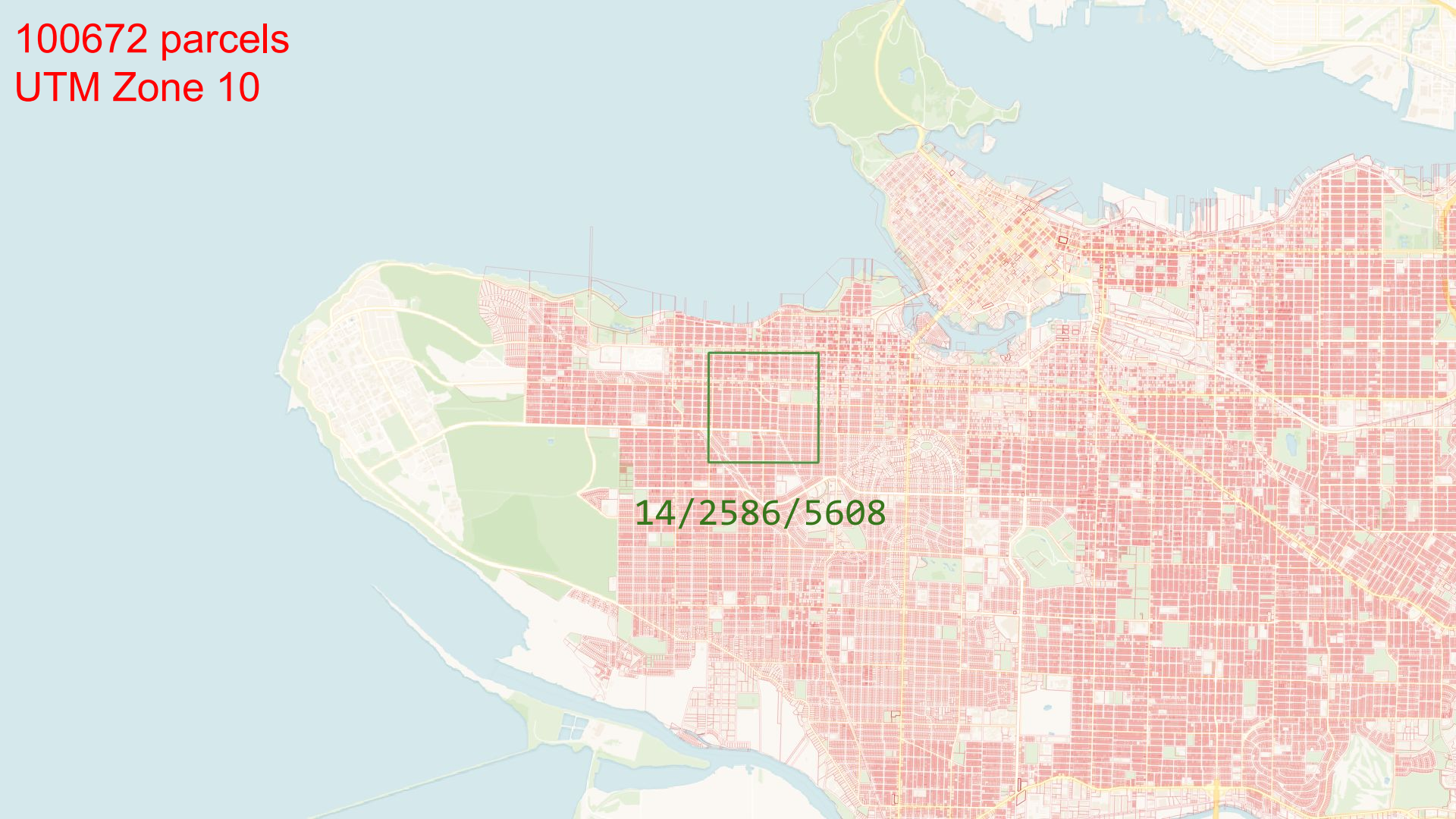
```
WITH mvtgeom AS (
  SELECT
    ST_AsMVTGeom(geom,
      ST_TileEnvelope(12,513,412)) AS geom,
    name,
    description
  FROM points_of_interest
  WHERE ST_Intersects(
    geom,
    ST_TileEnvelope(12,513,412))
  )
SELECT ST_AsMVT(mvtgeom.*) AS mvt
FROM mvtgeom;
```

All Together Now!

```
WITH mvtgeom AS (
  SELECT
    ST_AsMVTGeom(geom,
      ST_TileEnvelope(12,513,412)) AS geom,
    name,
    description
  FROM points_of_interest
  WHERE ST_Intersects(
    geom,
    ST_TileEnvelope(12,513,412))
  )
SELECT ST_AsMVT(mvtgeom.*) AS mvt
FROM mvtgeom;
```

EXPENSIVE

PARALLELIZABLE


100672 parcels
UTM Zone 10
14/2586/5608

Working in a Real Projection

```
WITH mvtgeom AS (
  SELECT
    ST_AsMVTGeom(
       ST_Transform(geom, 3857),
       ST_TileEnvelope(14,2586,5608)) AS geom,
    streetname, civic_no
  FROM property_parcel_polygons
  WHERE ST_Intersects(
    geom,
    ST_Transform(ST_TileEnvelope(14,2586,5608), 26910))
  )
SELECT ST_AsMVT(mvtgeom.*) AS mvt
FROM mvtgeom;
```

Working in Parallel

```
Finalize Aggregate
-> Gather
   Workers Planned: 2
   ->  Partial Aggregate
       ->  Parallel Bitmap Heap Scan on property_parcel_polygons
           Filter: st_intersects(geom, '01...41'::geometry)
           ->  Bitmap Index Scan on property_parcel_polygons_gix
               Index Cond: (geom && '01...41'::geometry)
```

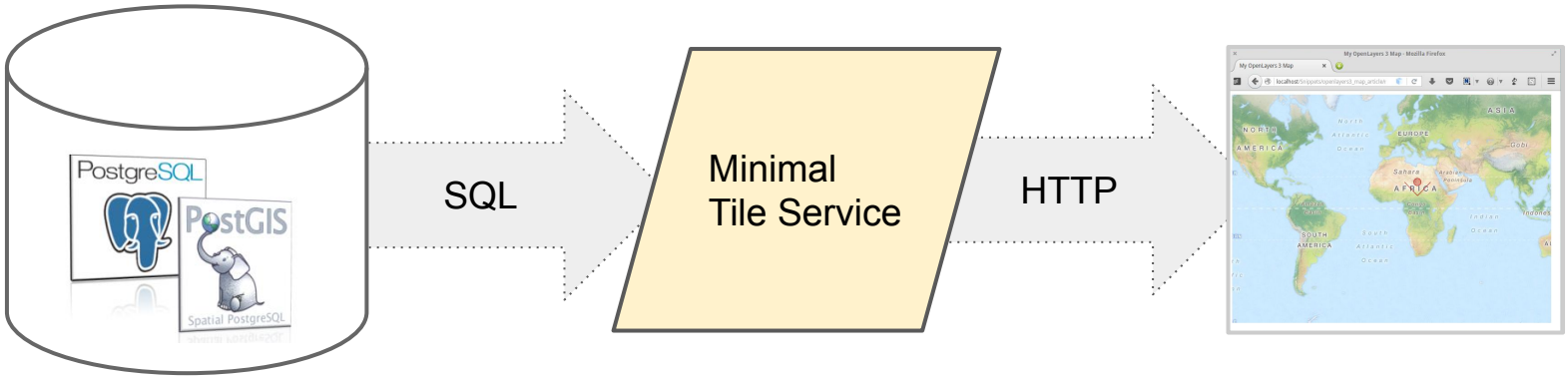
Planning Time: 83.956 ms
Execution Time: 166.764 ms
Results: 3324 / 100672

https://github.com/pramsey/minimal-mvt

Show me a DEMO!

crunchy data

Demonstration Architecture



- (SQL)
  - Query
  - Simplify
  - Quantize
  - Clip
  - Encode
- Req/Resp (HTTP)

Questions?

paul.ramsey@crunchydata.ca

https://is.gd/XIkcK6