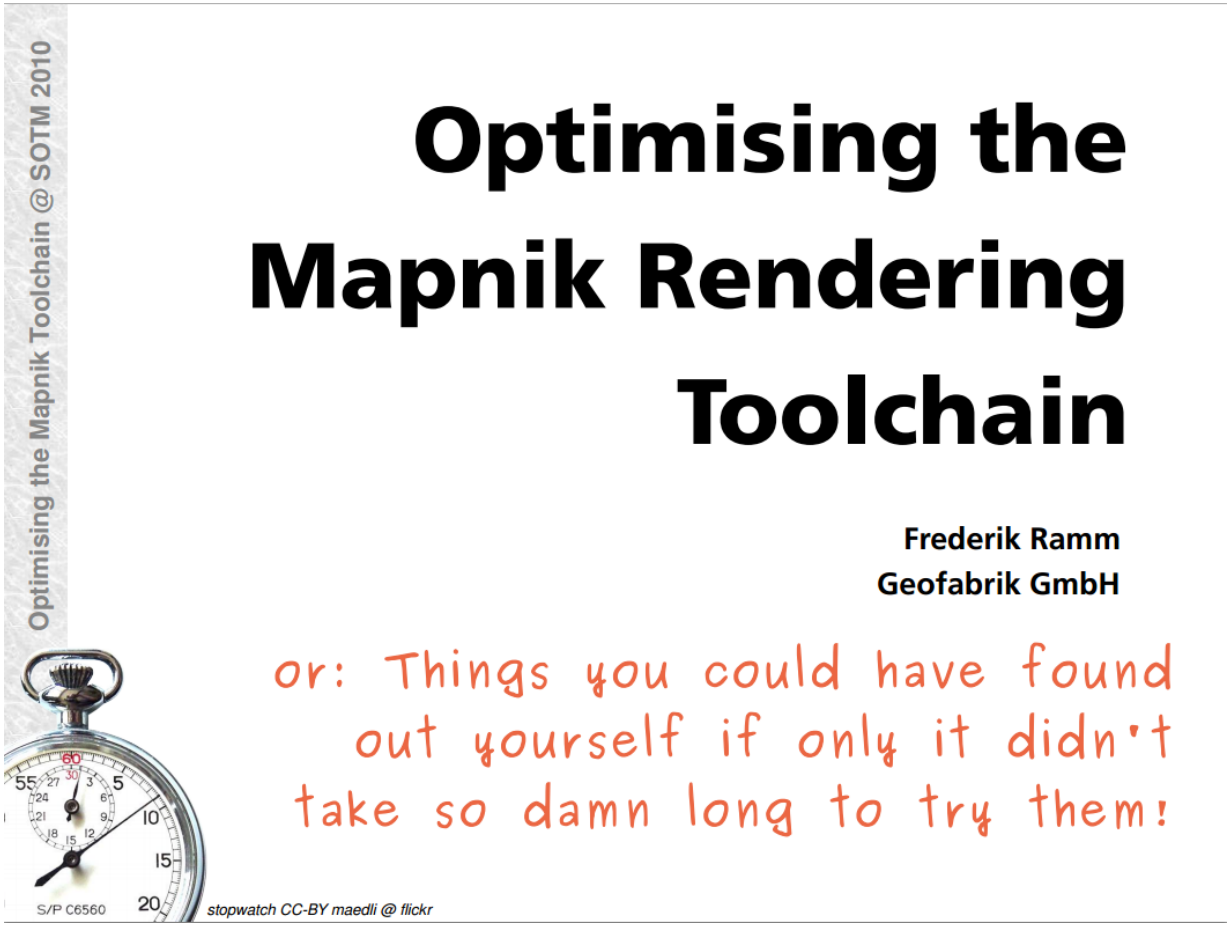
What you'd test if it didn't take so damn long,
and non-obvious improvements

Past work



http://www.geofabrik.de/media/2010-07-10-rendering-toolchain-performance.pdf

Past work

Optimising the Mapnik Toolchain @ SOTM 2012

Optimising the Mapnik Rendering Toolchain 2.0

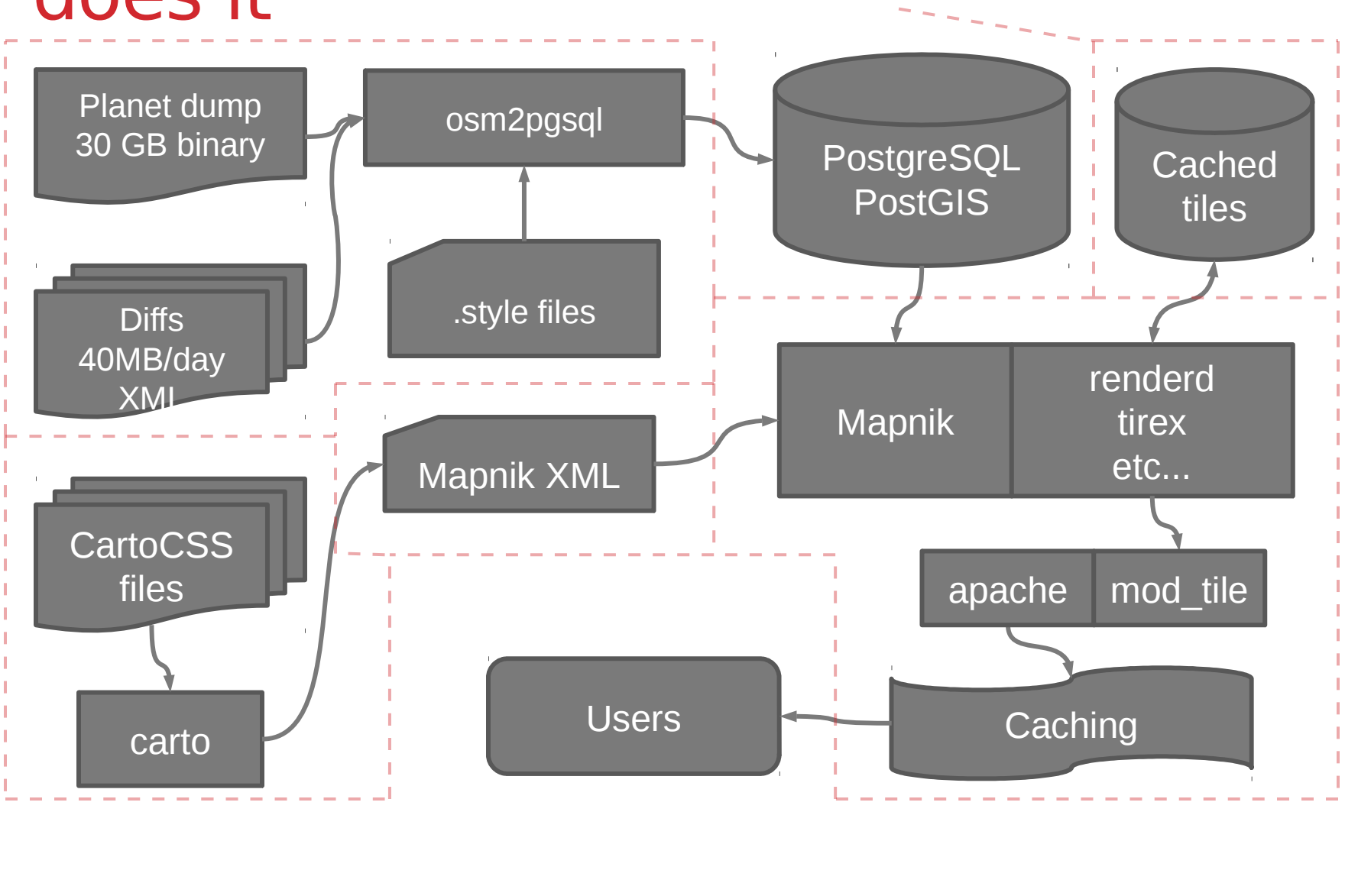
SOTM TOKYO 2012

Frederik Ramm
frederik@remote.org

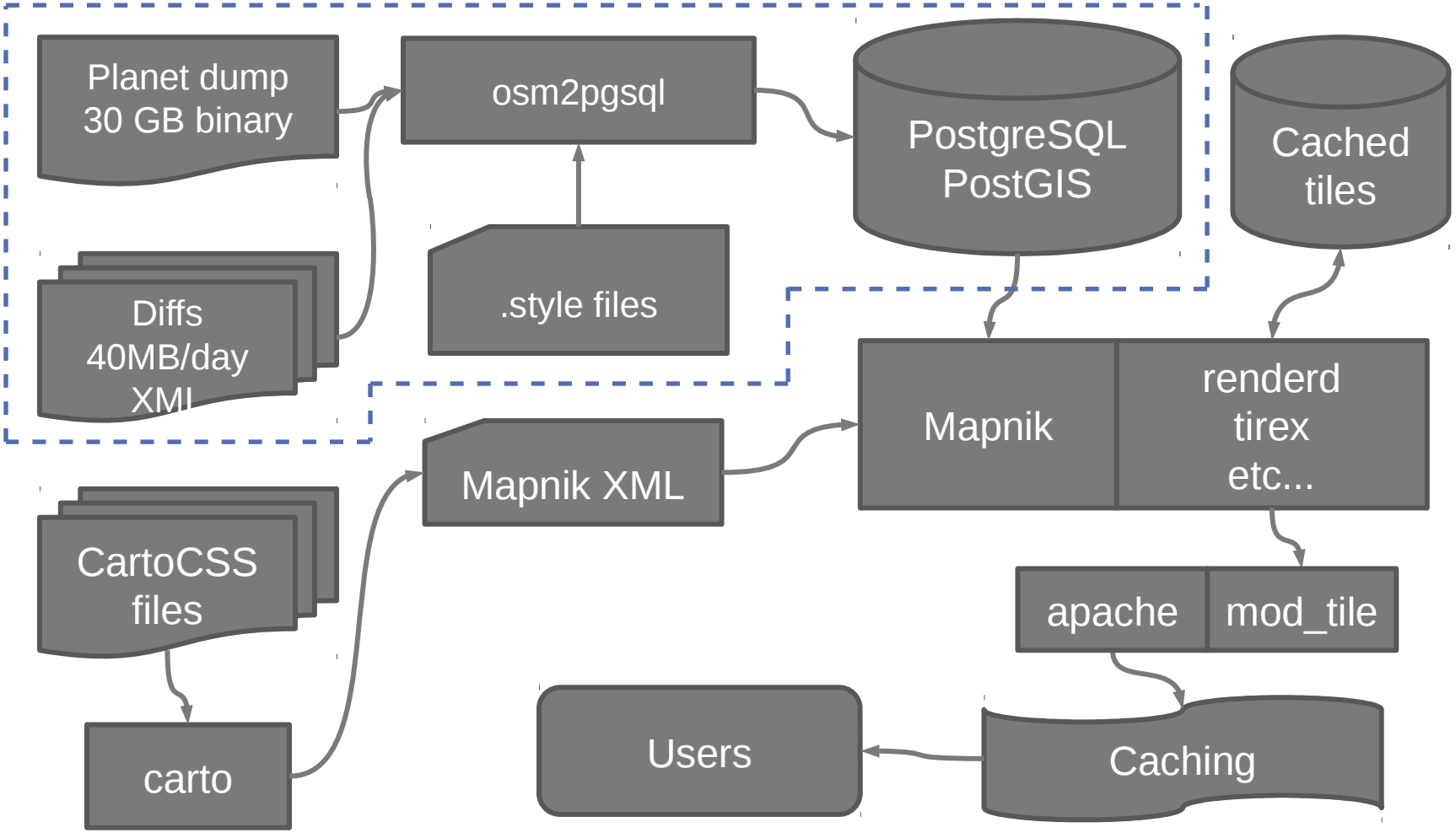
stopwatch CC-BY maedli @ flickr

http://www.remote.org/frederik/tmp/ramm-osm2pgsql-sotm-2012.pdf

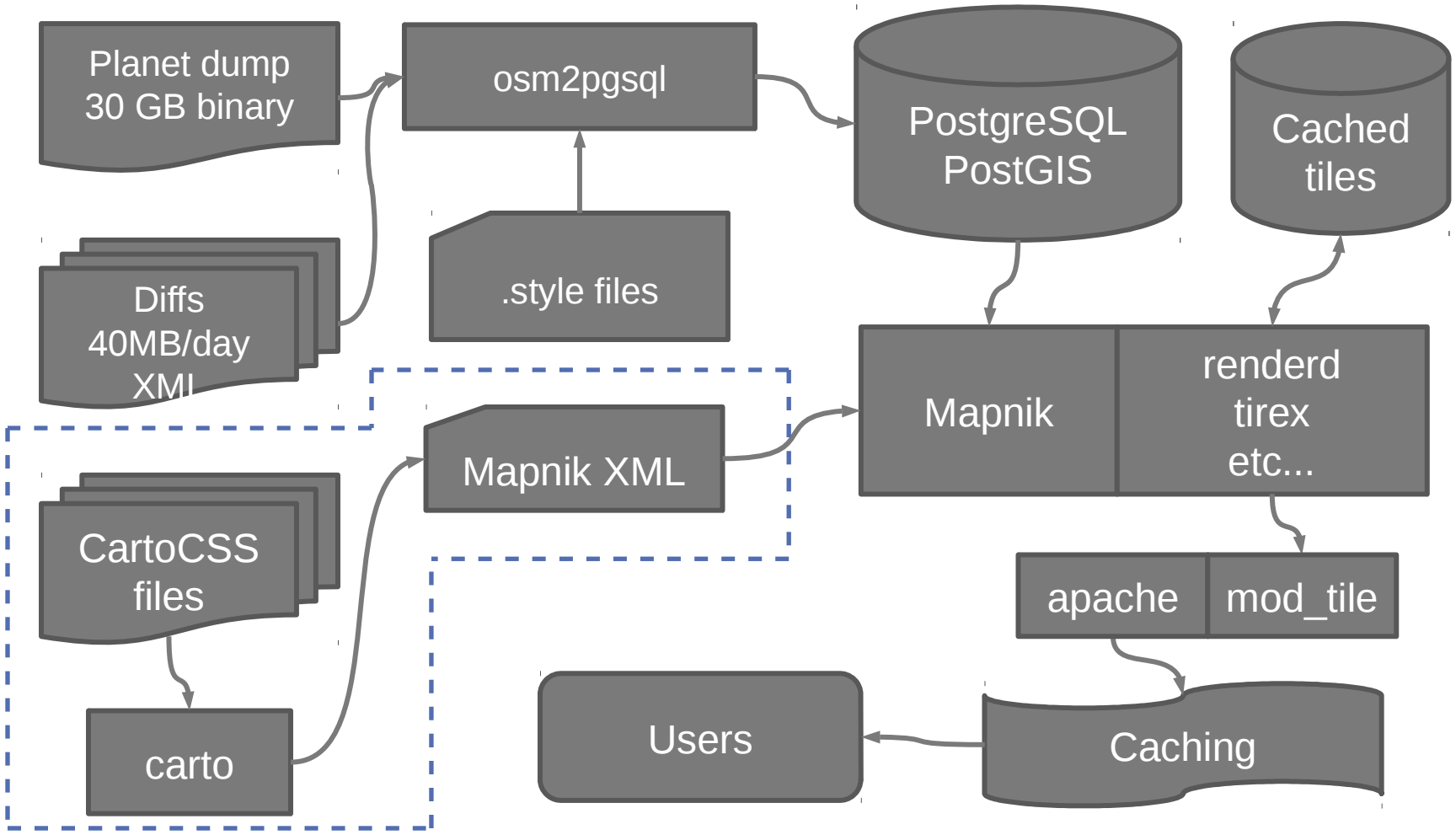
How almost everyone does it



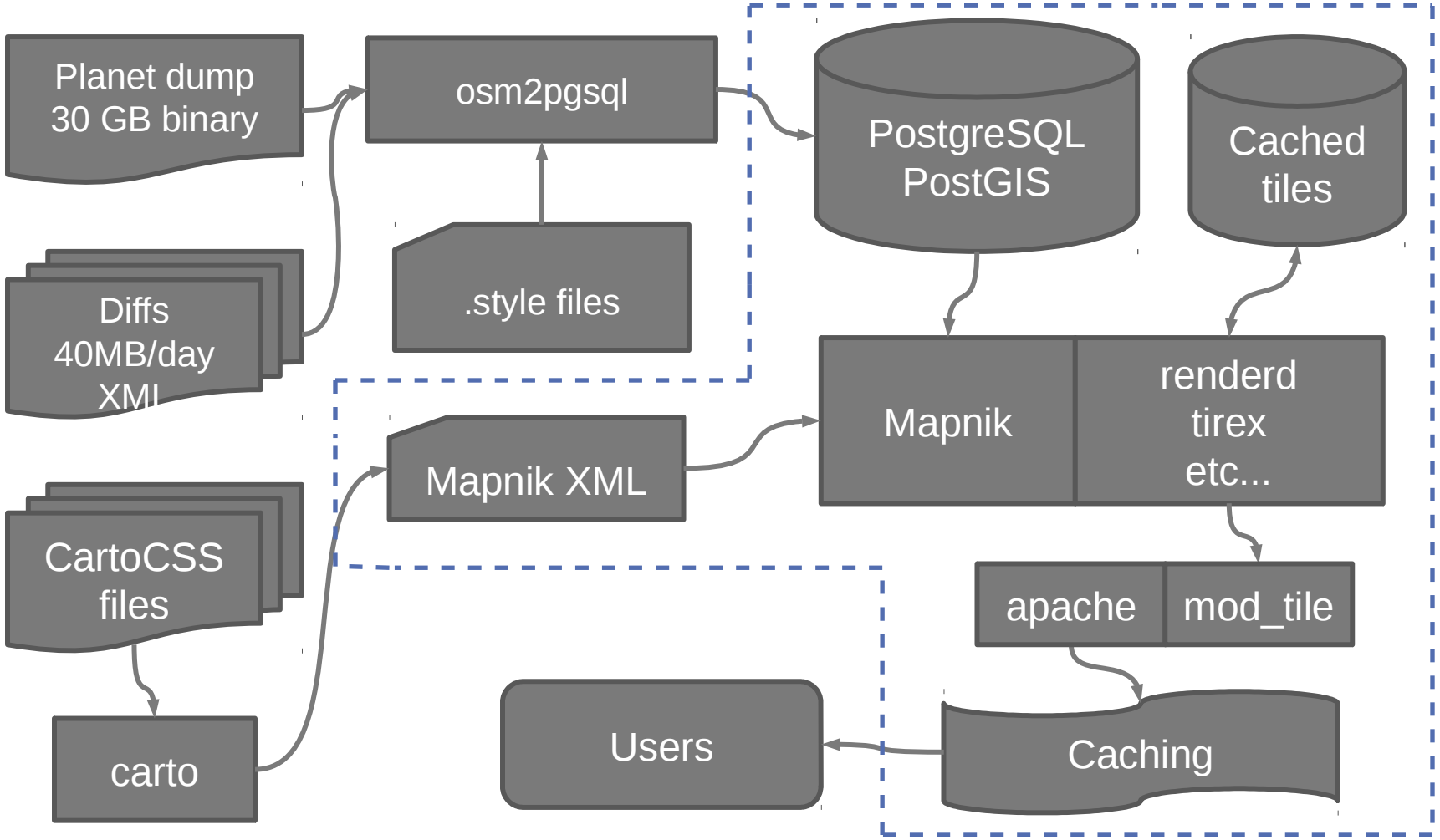
osm2pgsql


Planet dump
30 GB binary
osm2pgsql
PostgreSQL
PostGIS
Cached
tiles
Diffs
40MB/day
XML
.style files
Mapnik
renderd
tirex
etc...
Mapnik XML
CartoCSS
files
apache
mod_tile
carto
Users
Caching

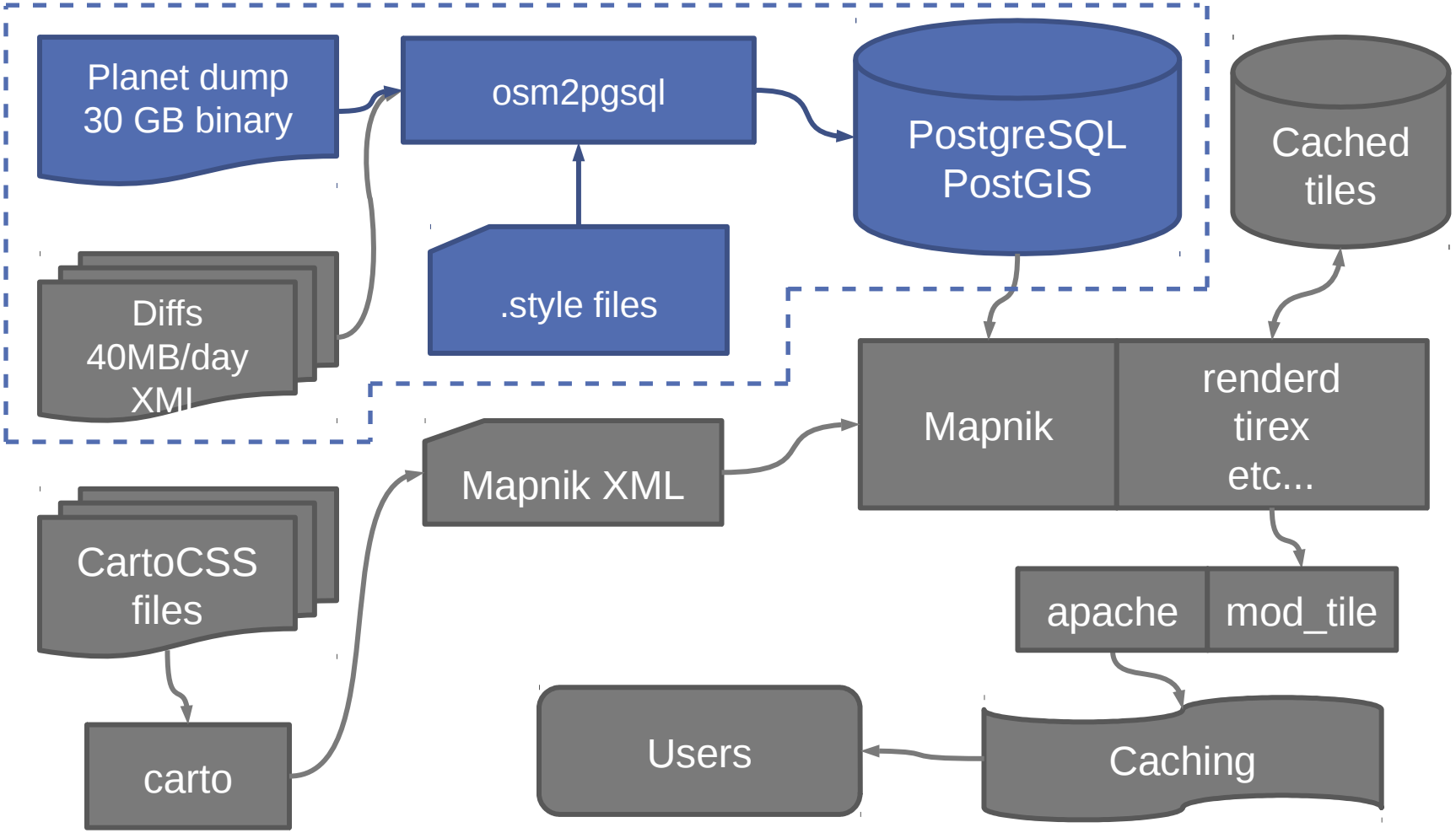
stylesheets



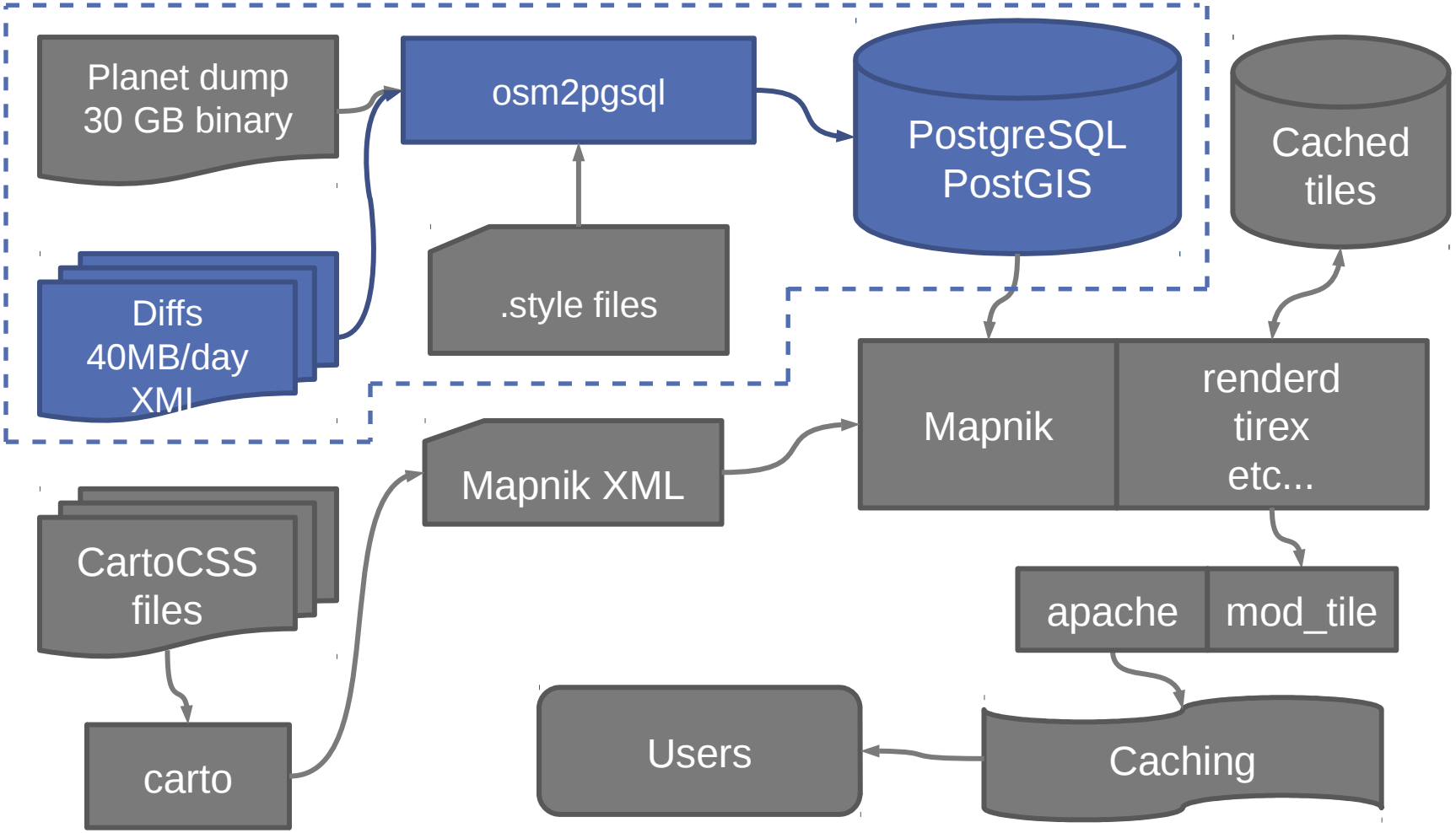
Rendering server



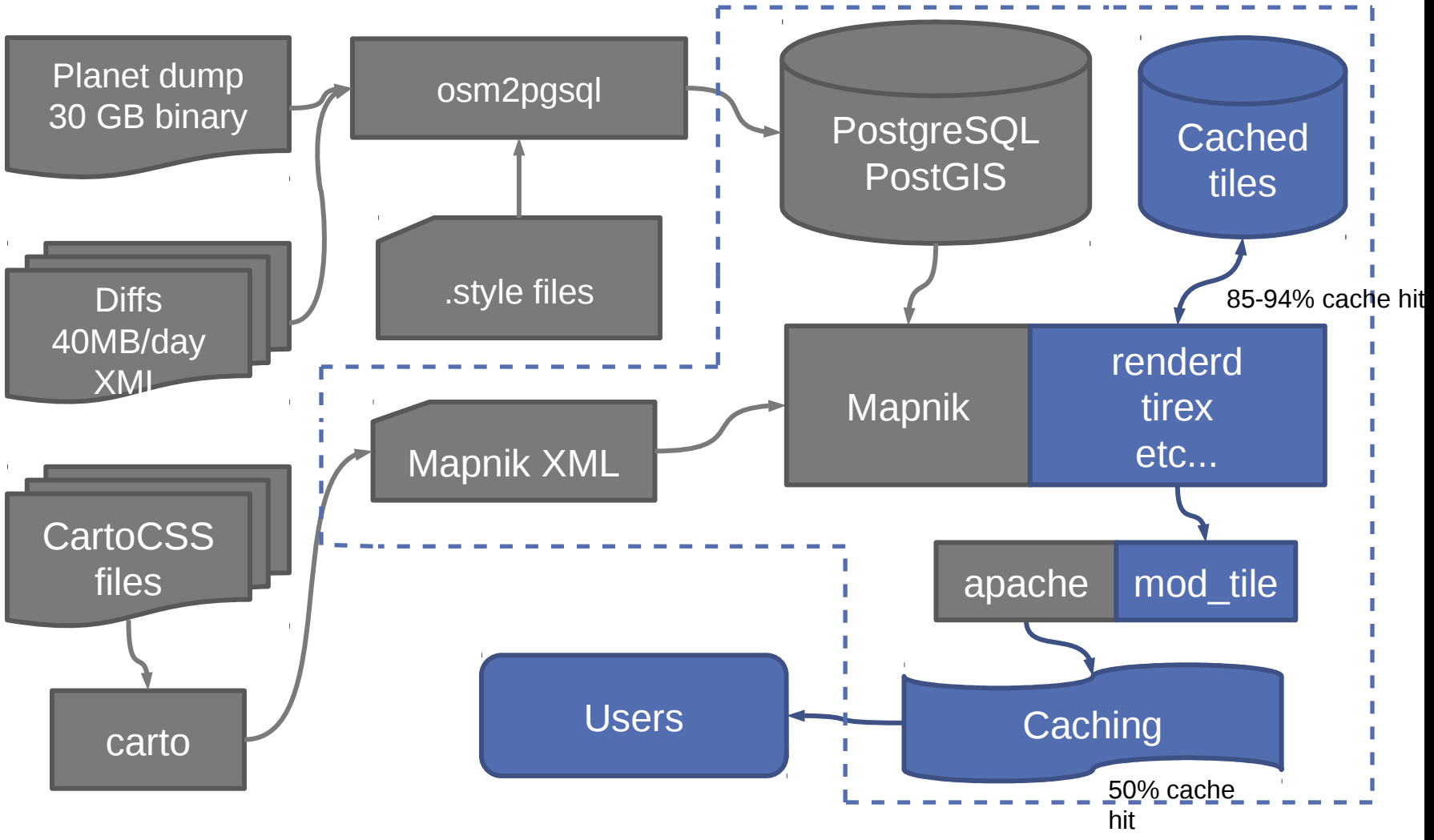
Initial import



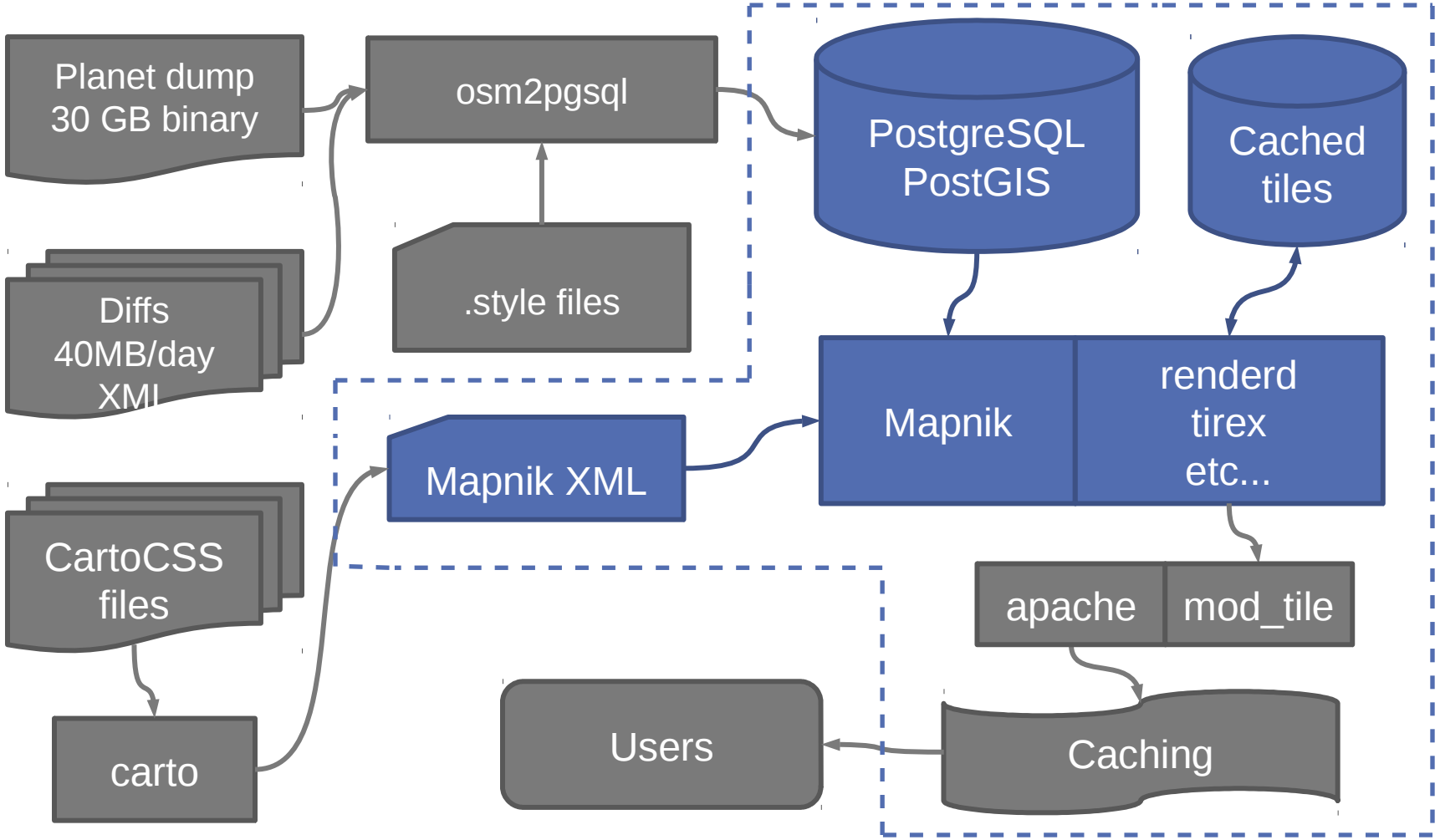
Updates


Planet dump
30 GB binary
osm2pgsql
PostgreSQL
PostGIS
Cached
tiles
Diffs
40MB/day
XML
.style files
Mapnik
renderd
tirex
etc...
Mapnik XML
CartoCSS
files
apache
mod_tile
carto
Users
Caching

Tiles: Cache hit



Tiles: cache miss



First, define requirements

- **If you don’t know your requirements, then how will you make sure your tile server meets them?**
- **Adjusting some of your requirements can speed up your server drastically**

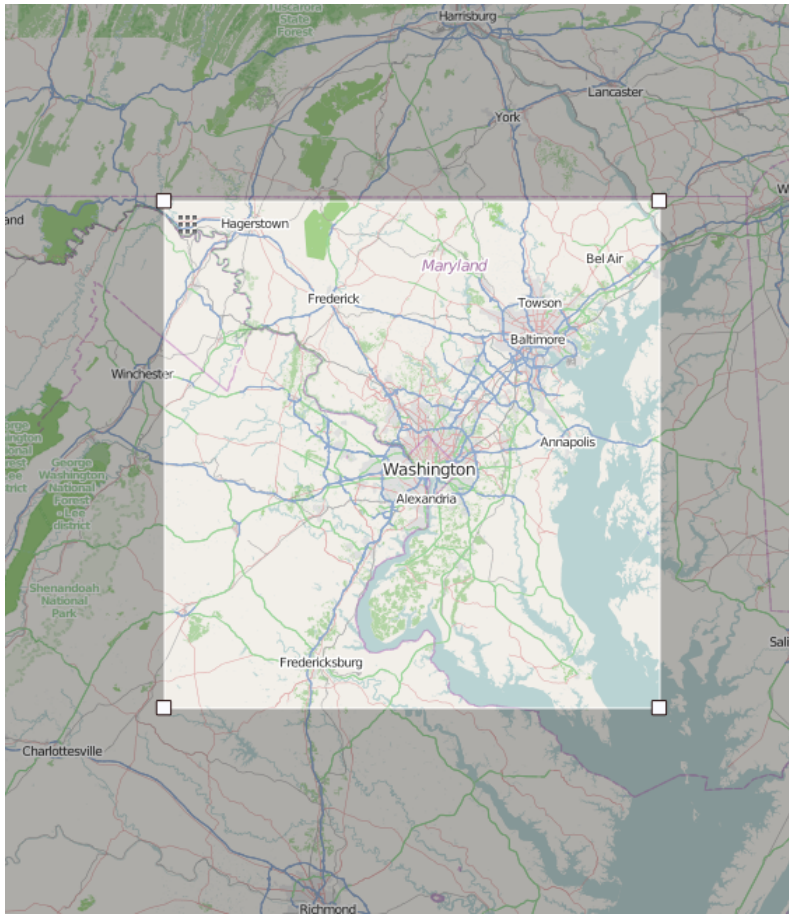
Area

Figure out what area you want to render

- Smaller areas need less drive space, better ram caching, quicker imports

Pick the smallest geofabrik extract that covers that area

- Allows their daily diffs for updates
- Possible to extract your own data, but more work



Updates

- **OpenStreetMap provides “minutely diffs” which let you keep your server in sync up to the minute**
- **You can update every minute, but should you?**
- **Minutely is less efficient than batching updates into chunks**
- **Suggested times**
  - 5 minutes
  - 1 hour
  - 1 day
  - 1 week

How old maps can you serve?

- **More aggressive caching means older maps but much less load**
- **Tiles in the browser cache don’t need to be requested from the server at all**

Load

- **Measured in tiles/second or similar**
- **tile.openstreetmap.org has 5000 peak and 3250 average tiles/second, double traffic a year ago**
- **Relates to site hits, but how depends how big the maps on your page are and how much panning they do**
- **One openstreetmap.org screen has about 30 tiles**
  - Your users will pan around looking at the map less than ours!
- **Check tile., render., orm., and yevaud. on http://munin.openstreetmap.org/**

Cache hit rates

- **Depends on *how* the users view the map**
- **If all users are viewing the same areas, higher cache hit rate**
- **If users view random places, lower cache hit rate**
- **How do you predict? Don’t know**

Hardware

- **Less expensive than you might think**
- **Requires a balance**
- **Requires drive speed, not capacity**
- **Don’t buy/rent a server with fast CPU and big slow drives and nothing else!**
- **For a full planet high performance server, ideally the database on a SSD, 24GB or more RAM, lots of CPU cores. Tile cache can be on HDDs**
  - wiki.osm.org/Servers has our specifications
- **Easy to run multiple servers in parallel, but generally you won’t have the load to need it**
  - tile.openstreetmap.org uses only two rendering servers

Optimization

Optimizing rendering

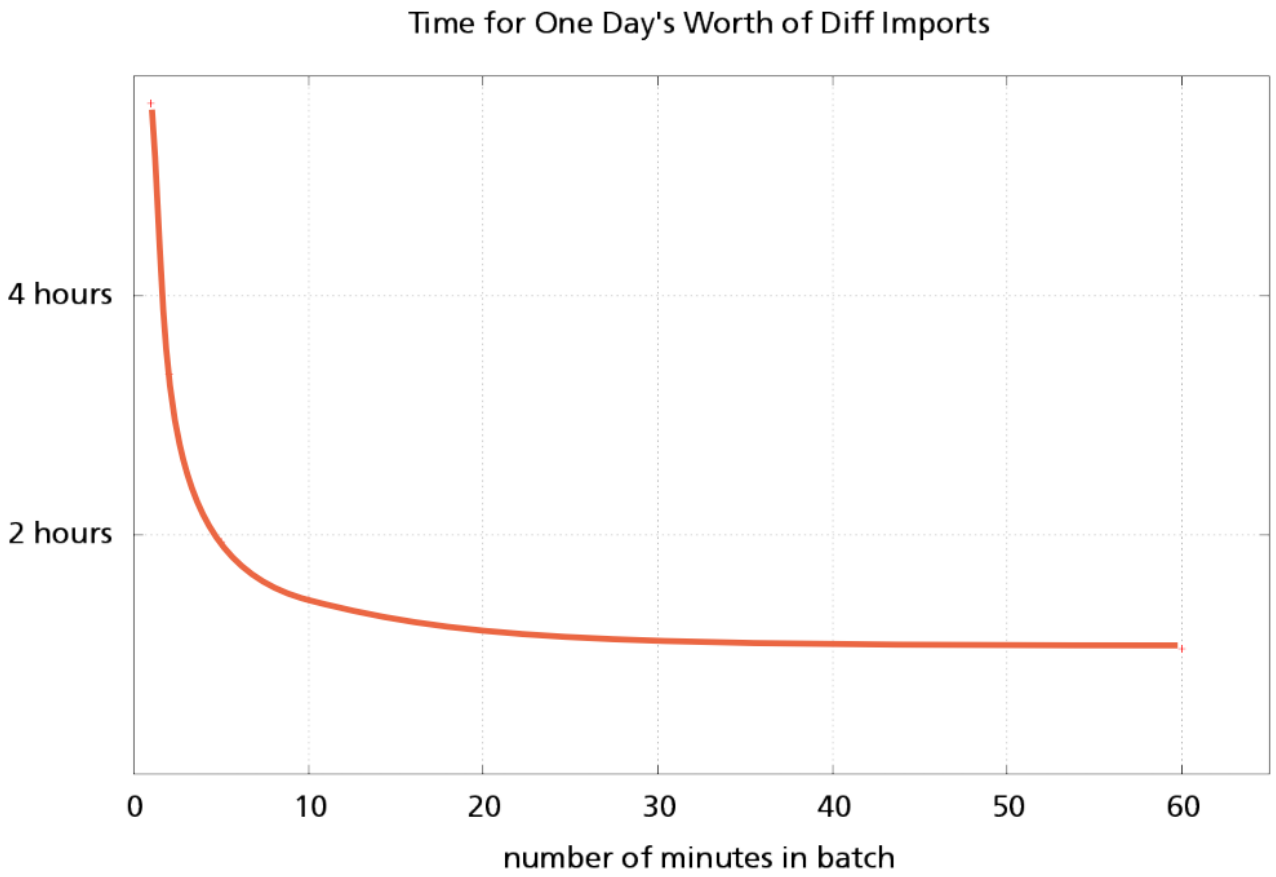
≈

Optimizing PostgreSQL

Updates

- **Updates come from planet.osm.org in minutely, hourly or daily increments**
- **Use osmosis to fetch updates, group them together and give them to osm2pgsql to apply to the database**
- **Update less often for less total time spent updating**
- **Update in low-traffic hours for less impact**
  - Might not work if you're worldwide
- **Geofabrik makes daily update files for their extracts**
  - Look for raw directory index then look for updates

Less updates



From 2012 presentation

Do you need to update?

- **Instead of updates, you can re-import the database**
- **Imports for small areas are fast**
- **Imports where you don’t need the update tables are faster**
- **Takes less disk space without update tables**
- **No need to re-CLUSTER (more on this later)**
- **Not updating only worth it for small areas or if you’re only updating weekly or monthly**

PostgreSQL queries

- **Lots of queries like this**

```
SELECT *
  FROM
    (SELECT way, highway FROM planet_osm_line
      WHERE highway IN ('motorway', 'trunk')
      ORDER BY z_order
    AS major_roads
  WHERE way && !bbox!;
```

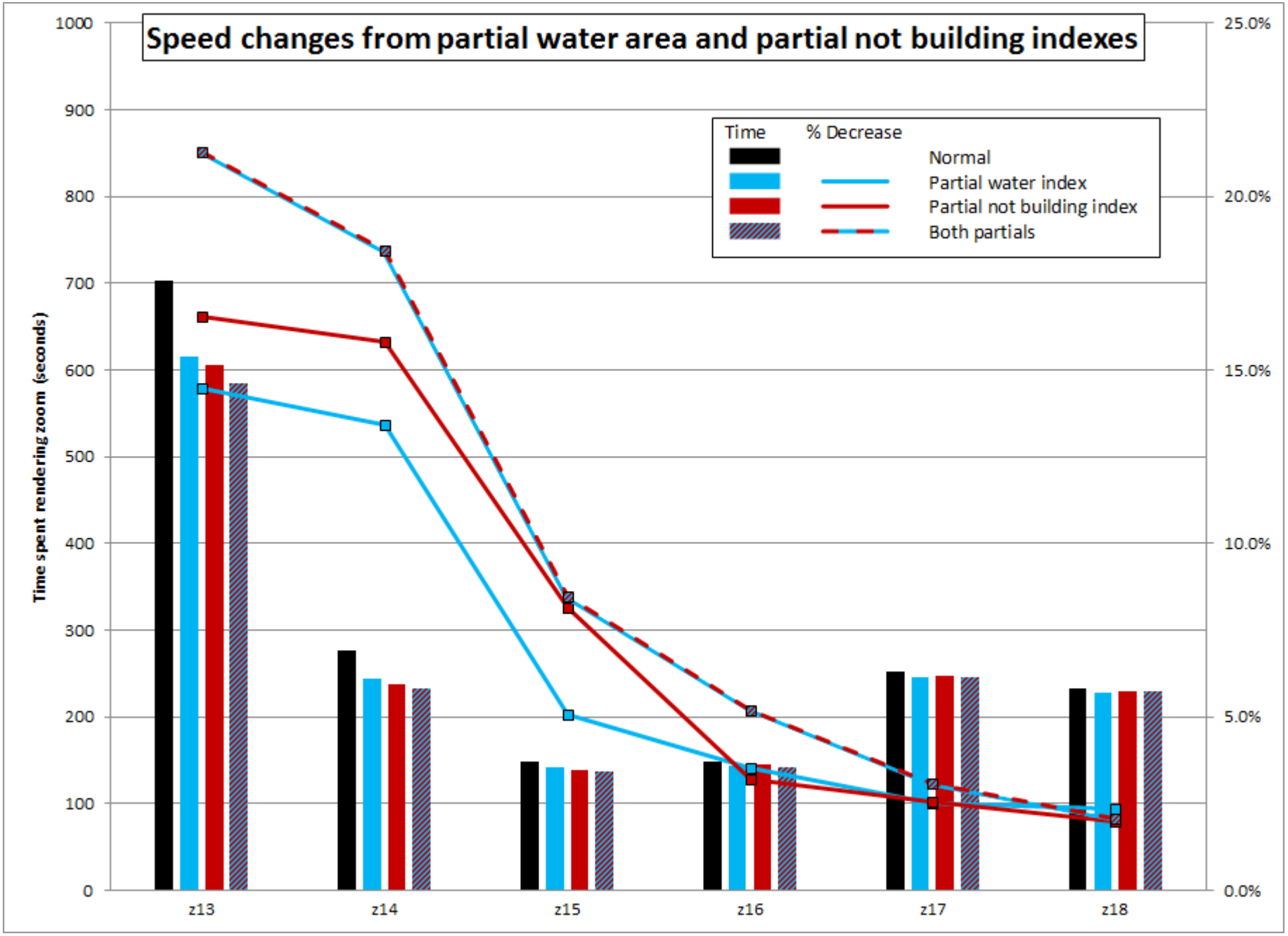
- **PostgreSQL will**
  - Index scan `planet_osm_line` based on `!bbox!`
  - Filter out non-roads
  - Sort the result

speed it up: Index scanning

- **Index scanning planet_osm_line for rows that match `way && !bbox!`**
- **Default index is `"planet_osm_line_index" gist (way)`**
  - This index covers *all* lines, not just roads
- **Wouldn't it be handy to have an index that just covers roads**
  - Smaller index
  - Only returns rows so less to filter out
- **Partial indexes!**

```
CREATE INDEX ON planet_osm_line USING gist (way)
  WHERE highway IS NOT NULL;
```

- **This index only contains highways**


Speed changes from partial water area and partial not building indexes
Time
% Decrease
Normal
Partial water index
Partial not building index
Both partials
Time spent rendering zoom (seconds)
1000
900
800
700
600
500
400
300
200
100
0
25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
z13
z14
z15
z16
z17
z18

Speed it up: Filtering

- **Filtering out rows for `highway IN ('motorway', 'trunk')`**
- **Partial geometry indexes help hugely**
- **Btree partial indexes might help, but not as much as partial geometry indexes in my testing**

```
CREATE INDEX ON (highway)
  WHERE highway IS NOT NULL;
```

- **Make sure to match indexes to queries**
- **One other type of filtering… geometry && re-checks**

Bitmap index scans and filtering

- **When scanning an index, PostgreSQL builds a list of locations where data matching the query is**
- **At low zooms this list can get very large, so it has to use a bitmap index scan instead**
- **This technique uses less memory, but returns unnecessary rows**
- **Governed by `work_mem` setting**
- **Default PostgreSQL setting is far too low for rendering**
- **Try 32-64 MB**

Speed it up: sorting

- **Give PostgreSQL enough `work_mem` to sort in memory instead of slow disk sorts**
- **32-64 MB**
- **Total of 20-40% speed gain with appropriate `work_mem`**

Clustering

- **Clustering places geographically nearby points**
- **Don’t cluster with a gist index, use ST_GeoHash.**
- **Do not rely on osm2pgsql to cluster tables, released versions do not do this right**

```
CREATE INDEX tmp_line
  ON
(ST_GeoHash(ST_Transform(way,4326));
CLUSTER planet_osm_line USING tmp_line;
DROP INDEX tmp_line;
```

- **~14% faster with gist (way) clustering**
- **~25% faster with ST_GeoHash clustering**

DB maintenance

- **Database maintenance is an essential task**
- **Statistics**
  - Increase `default_statistics_target` to 1000 or 10000
  - Run `ANALYZE` periodically
- **Table bloat**
  - autovacuum will limit this
  - Make autovacuum more aggressive! Adjust `autovacuum_vacuum_scale_factor` to about 0.05, `autovacuum_analyze_scale_factor` to about half

DB maintenance

- **Index bloat**
  - autovacuum does not limit index bloat
  - MUST reindex or rewrite the table
  - Reindexing can be done concurrently, but faster if you can stop the server and reindex all tables in parallel
  - ~80 minutes in parallel on my benchmark server
  - Consider if you need to reindex just the rendering tables or also the update tables
    - Update tables take longer to reindex, but you can stop updates and reindex
- **CLUSTER degredation**
  - As the tables are updated, they're no longer in the nice clustered order
  - Recreate the ST_GeoHash index and re-CLUSTER
  - Rewrites the indexes too, so eliminates index bloat

When to reindex and cluster

- Postgres provides tools to examine table/index bloat
  - pgstattuple extension
- Strong correlation between table correlation and reduced rendering speed

```
SELECT CORR(page,geohash)

FROM (SELECT (ctid::text::point)[0] AS page,

rank() OVER (ORDER BY
st_geohash(st_transform(way,4326))) AS geohash

FROM planet_osm_roads) AS s;
```

Upcoming results

- **More detailed relationship between correlation and slowdown to better plan when clustering is required**
- **Bulk rendering strategies**
- **Pgbouncer?**
- **Hstore and slimmer styles**
- **Blog posts to paulnorman.ca**
- **Results cross-posted to tile-serving@ mailing list: https://list.osm.org/listinfo/tile-serving**

Upcoming osm2pgsql features

- **Threaded branch**
  - **Scaling of more parts of the import across multiple cores**
  - **Fixes clustering!**
- **Partitioning**
  - **Allows partitioned tables**
- **Better tools for dump and reload updates**