FastAPI and Tile38: Ultra fast retrieval of geospatial data

Ganesh N. Sivalingam

Parkopedia

Overview

- Introduction to myself and Parkopedia
- Example problem
- Tile38
- FastAPI
- Demo!
- Benchmarking against PostGIS
- Conclusions

About me

- Started at Parkopedia 6 years ago as a Data Scientist, Big Data
- Head of Data Science, technical work predominantly focussed on Data Engineering

Parkopedia

- World's largest parking se
  providing parking inform
  from Apple to Lambor
- Products
  - ○ Database of static
  - ○ Predictions of park
  - ○ Parking search routes
  - ○ Many others
- Moving into EV data

Lots of i ospatial, data science and
es

cessing

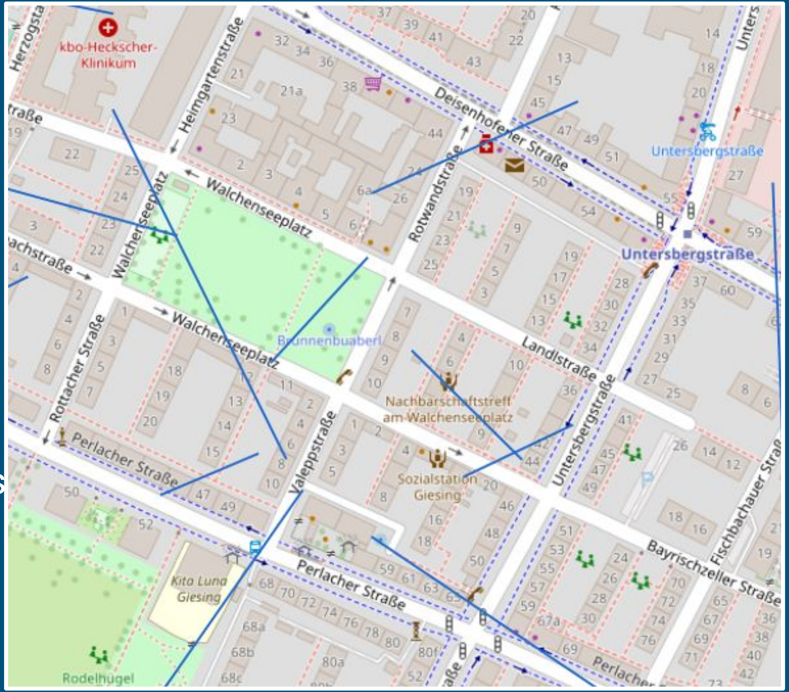
jobs

y of data serving millions of vehicles

- Machine learning modelling
- Routing and route optimisation

Example problem

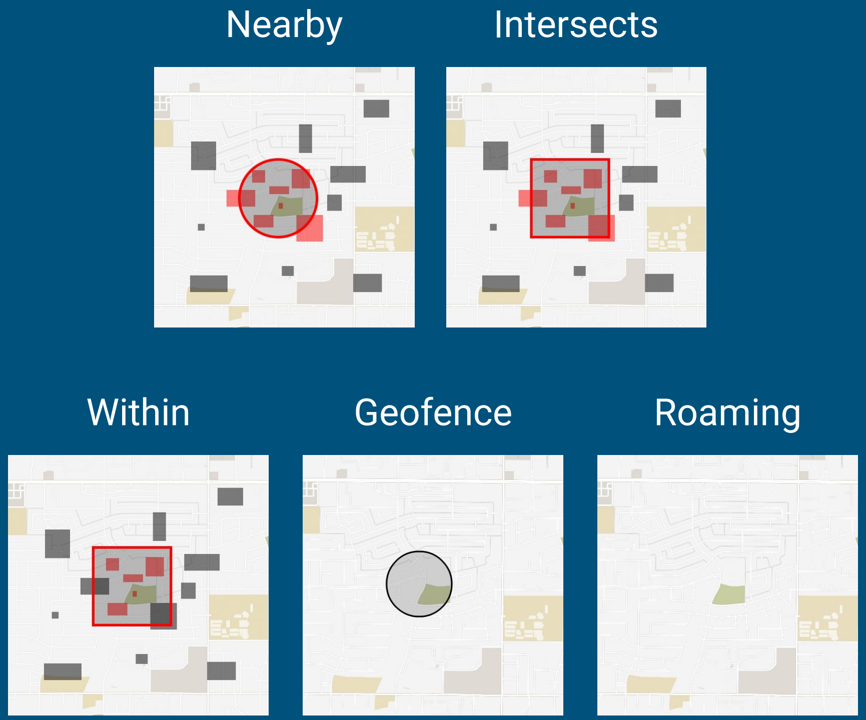
- Millions of parking locations to store
- Retrieve parking locations that are
  - ○ Within a certain distance of a car
  - ○ Are free or paid
  - ○ We need to be able to serve results as quickly as possible to service the large fleet of users
- Code is available at github.com/gnsiva/tile38-fastapi



Generated dummy parking locations

Tile38

- Open source
- Fast, in-memory, geospatial database and geofencing server
- Rapid release cycle, and very quick to fix bugs
- Thoroughly documented
- Supports several geospatial queries



Nearby

Intersects

Within

Geofence

Roaming

Credit: tile38.com

Tile38 is easy to use

- Compatible with most Redis clients
- Docker container and binaries available so very easy to get started
- Dedicated Python package `pyle38`, but it isn't very mature so we will use `redis-py`

Data is held in a structure similar to a Python dictionary within dictionary

Inserting data

```
SET <key> <field> OBJECT <geojson>
```

Retrieving data directly

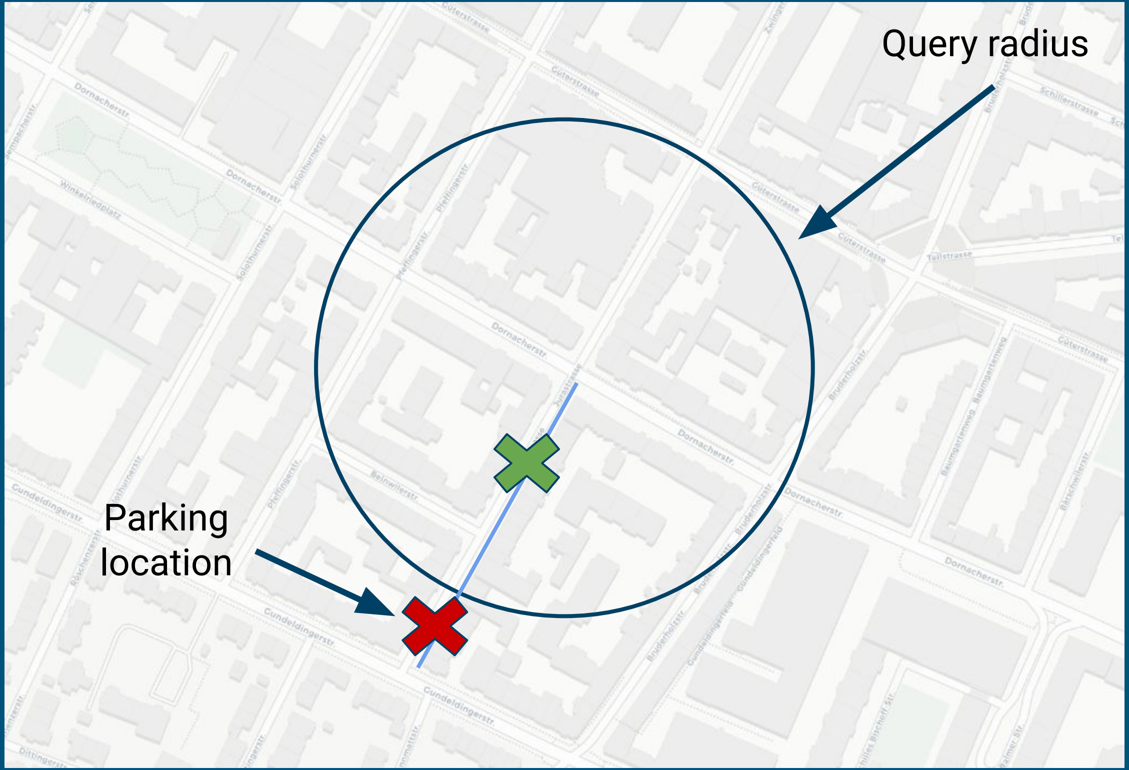
```
GET <key> <field>
```

Retrieving all objects near a location

```
NEARBY <key> POINT <lat> <lng> <meters>
```

Why not use Redis?

- Extremely fast BUT
- Can only store points, not polylines or polygons
- Limited querying
  - Radius
  - Bounding box



Query radius

Parking location

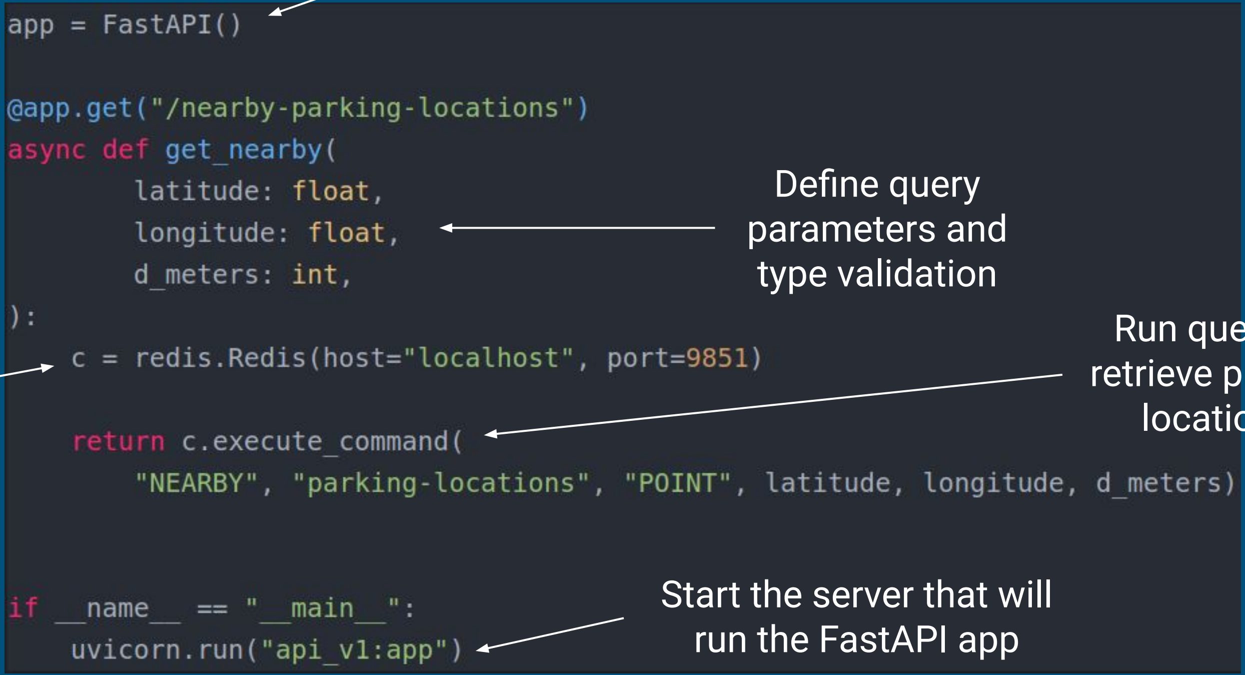
FastAPI

- An open-source “Fast”, “API” framework for Python
- Developer friendly and easy to work with
- Automatically generates OpenAPI schemas and launches documentation
- Built around Python’s asyncio module, which allows it to handle high volumes of I/O bound queries.
- Great for developing microservices

FastAPI basic app

Instantiate the FastAPI app

Decorator to add endpoints



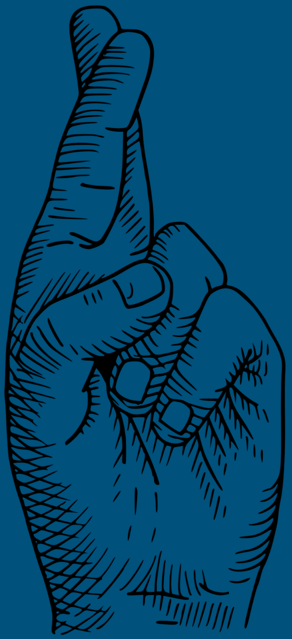
```python
app = FastAPI()

@app.get("/nearby-parking-locations")
async def get_nearby(
        latitude: float,
        longitude: float,
        d_meters: int,
):
    c = redis.Redis(host="localhost", port=9851)

    return c.execute_command(
        "NEARBY", "parking-locations", "POINT", latitude, longitude, d_meters)


if __name__ == "__main__":
    uvicorn.run("api_v1:app")
```

Define query parameters and type validation

Create client to connect to Tile38

Run query to retrieve parking locations

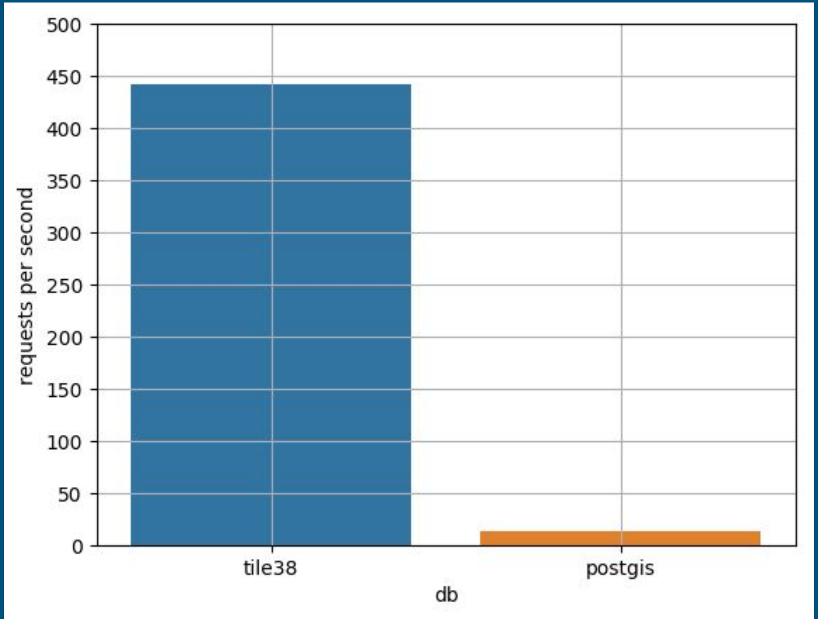
Start the server that will run the FastAPI app



Demo

Benchmark

- Tested against PostGIS
- Spatial index on geometry column in PostGIS
- Used Apache Bench to conduct benchmarks



>30x increase in requests per second

Conclusion

- We developed a “Fast” API for retrieving parking locations
- Parkopedia
- Where to learn more
  - tile38.com
  - fastapi.tiangolo.com

Demo code and slides
github.com/gnsiva/tile38-fastapi

We are hiring Data Engineers and
Computer Vision Research Engineers
parkopedia.com/jobs

Demo code and slides
github.com/gnsiva/tile38-fastapi

We are hiring Data Engineers and
Computer Vision Research Engineers
parkopedia.com/jobs

Thank you!

Questions?