Setting Healthy Boundaries
Generating Geofences at Scale with Machine Learning

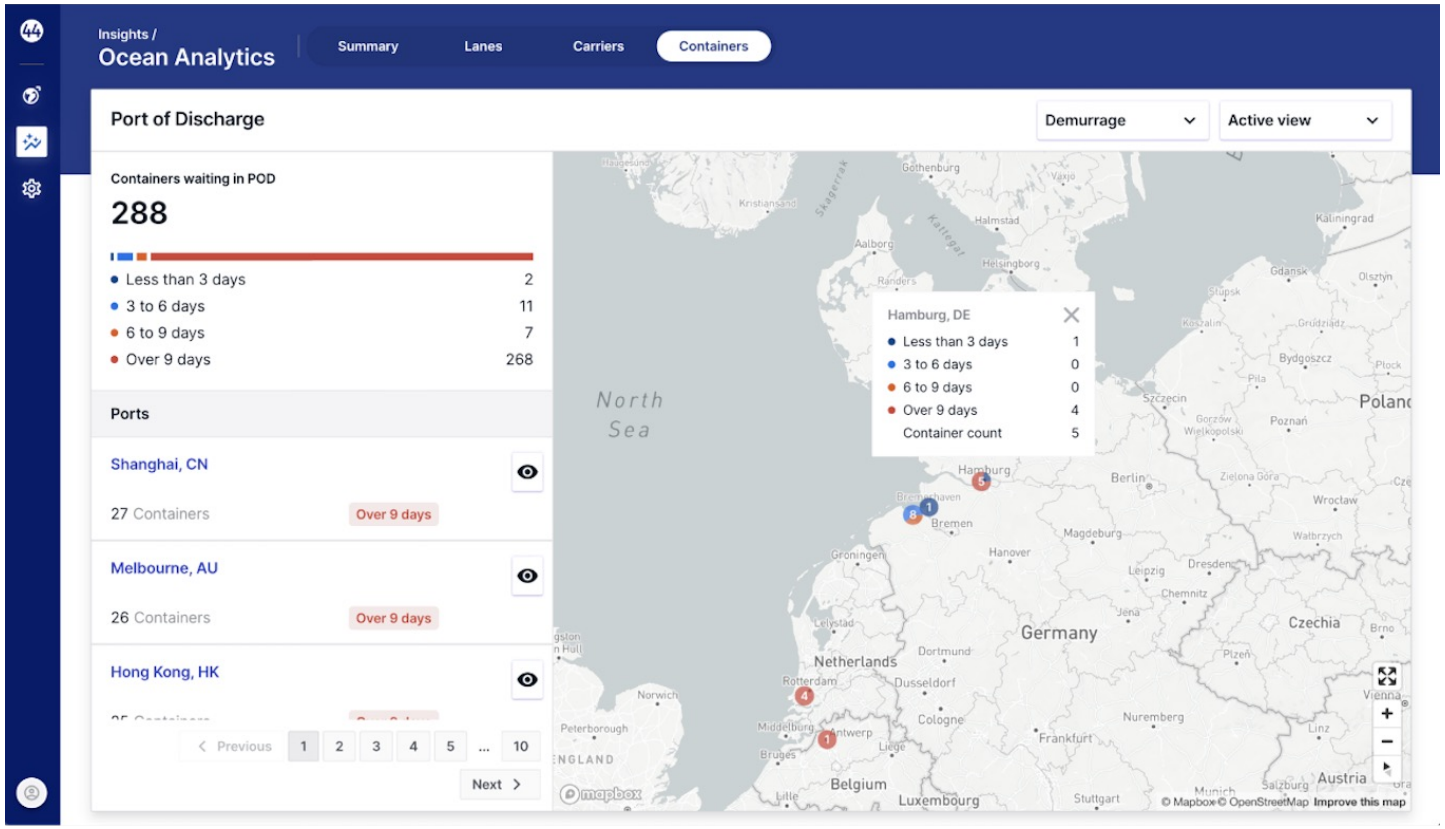
Stephanie Kirmer
Staff Data Scientist
www.project44.com

About project44

Our platform, Movement
Visit us to learn more!
www.project44.com

Logistics visibility and analytics

- Where's my stuff?
- When's my stuff gonna get here?



We work across the globe and across different modes, including rail, ocean, trucking, and air.

Our customers are shippers, recipients, carriers, and/or other participants in the movement of goods – we can help everybody involved.

Note: the geofence technique I'm discussing is patent pending

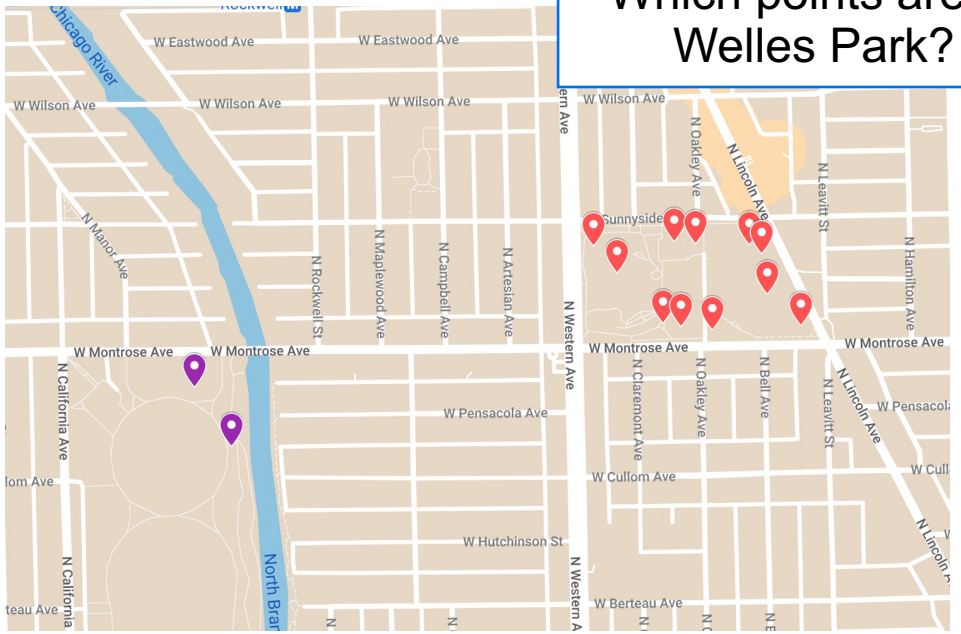
Introduction to Geofences

Geofence: a virtual boundary drawn around a physical location.

For this talk, we'll be using Welles Park in Lincoln Square as our example location. We have some historical data on visits to the park, as well as visits to nearby Horner Park.

How do we tell which is which, and where Welles Park actually is?

Which points are at Welles Park?

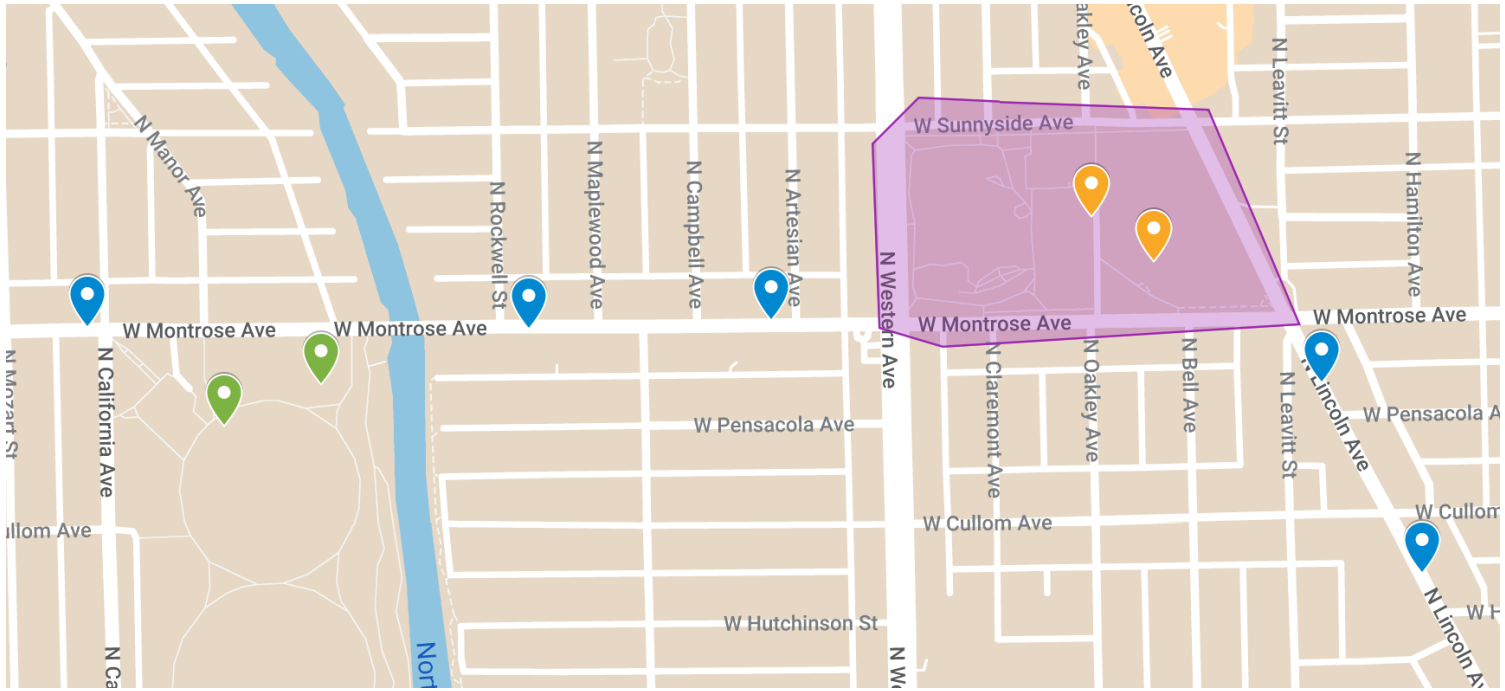


How We Use Geofences

telemetry = GPS pings indicating a vehicle's physical position and timestamp

At p44 we define arrival at shipping stops by comparing **telemetry** with **geofences**.

Where is the shipment + where is the location = Is the shipment at that location?



Open Source Tools Used

Python		https://www.python.org
Pandas		https://pandas.pydata.org
GeoPandas	*spatial analysis*	https://geopandas.org/en/stable/
Folium	*visualization*	http://python-visualization.github.io/folium/
Dask	*parallelization*	https://www.dask.org
DBSCAN in scikit-learn	*clustering*	https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
Airflow	*deployment*	https://airflow.apache.org

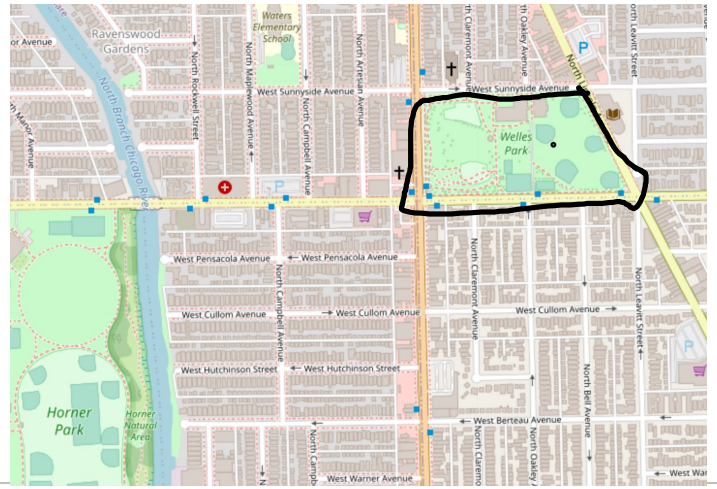
Ways to Create Geofences

Circular radius

- Imprecise
- Easy
- Fast
- Inaccurate

Draw by Hand

- Precise
- Difficult
- Slow
- Maybe accurate?



Computational

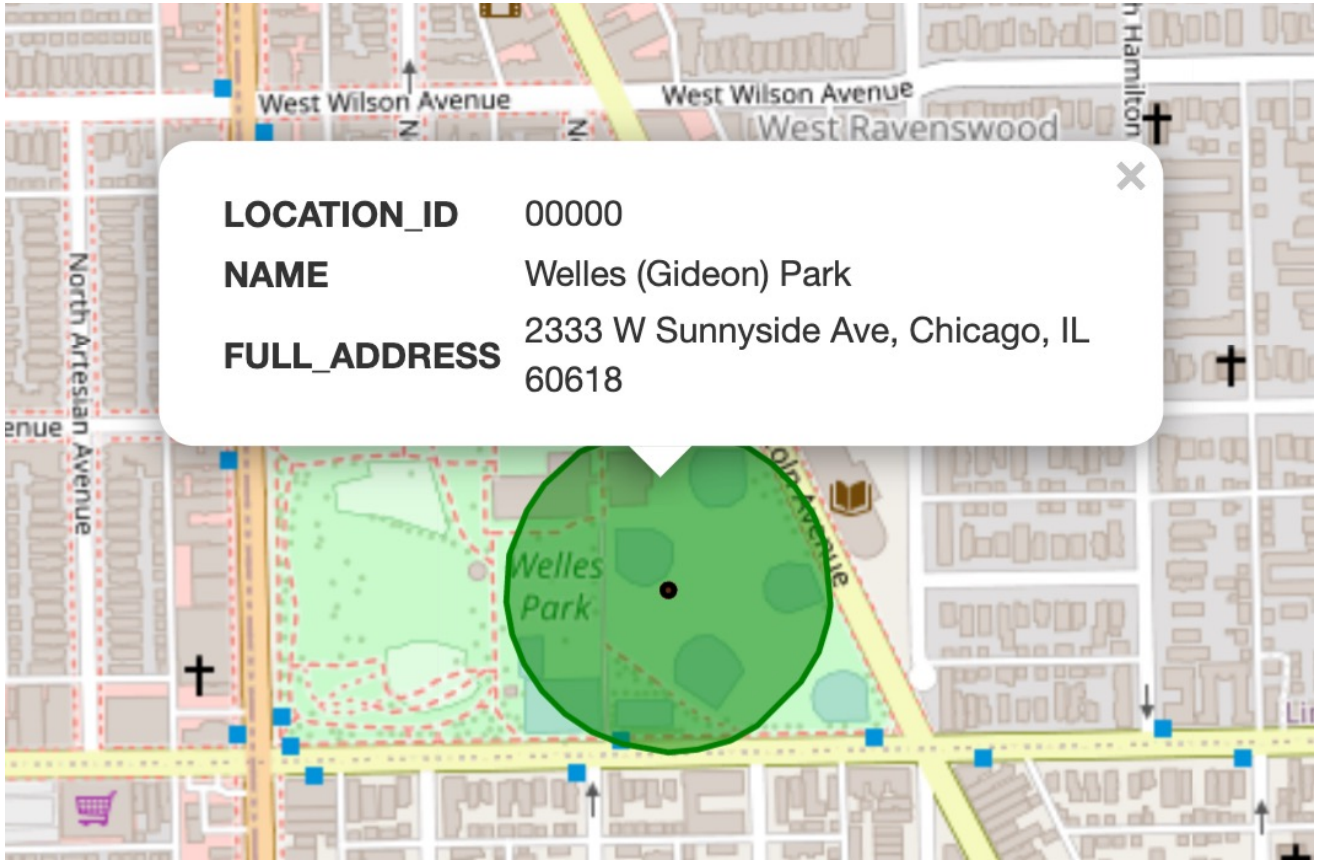
- Precise
- Easy
- Fast
- Data-driven accuracy

Evaluating Results – Too Small

Too-small geofence means we miss arrivals that really happened

Effects:

- Drivers don’t get credit for drop-offs they make
- Estimates of their next arrival time are messed up
- Stores don’t know where their inventory is

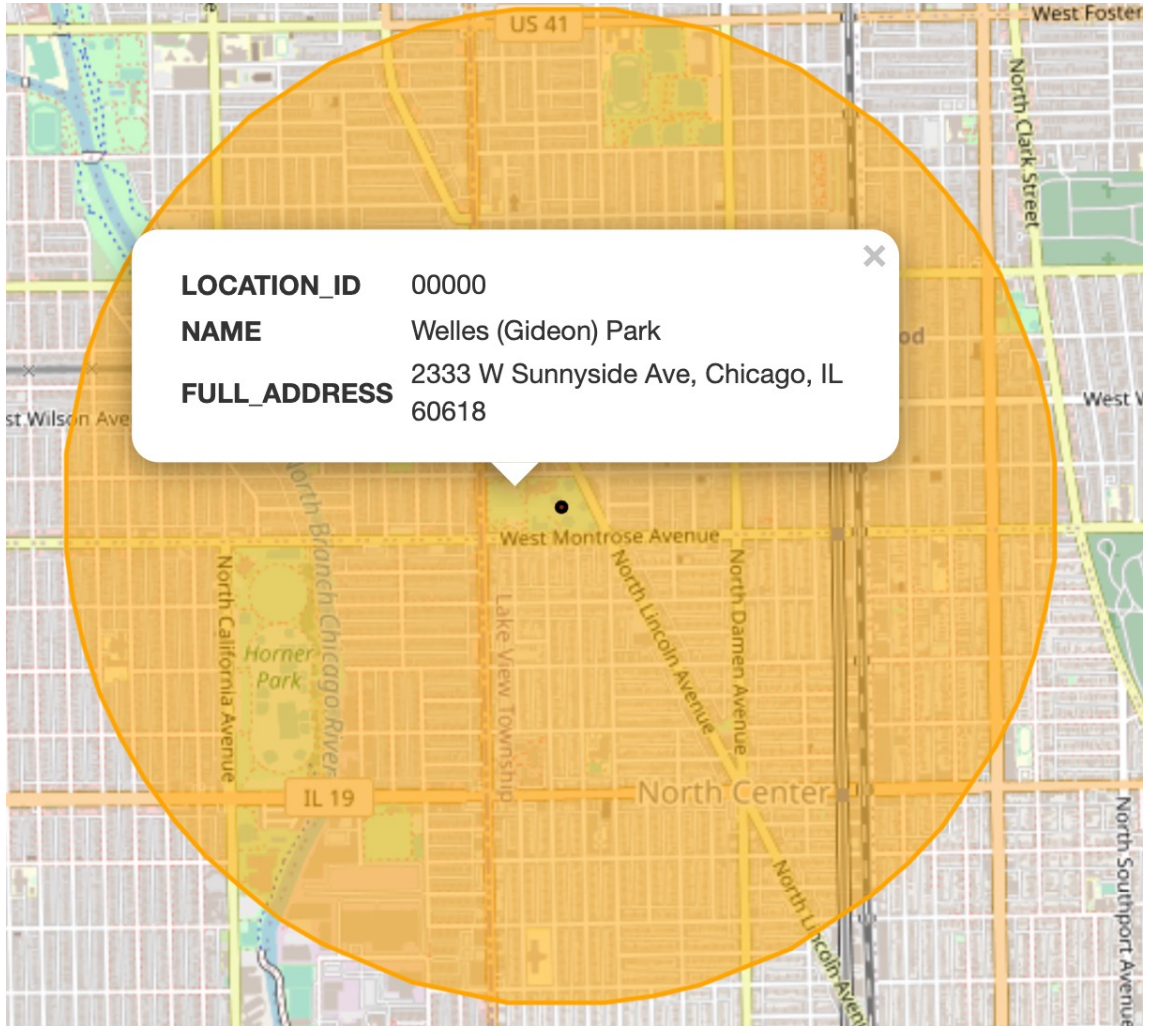


Evaluating Results – Too Large

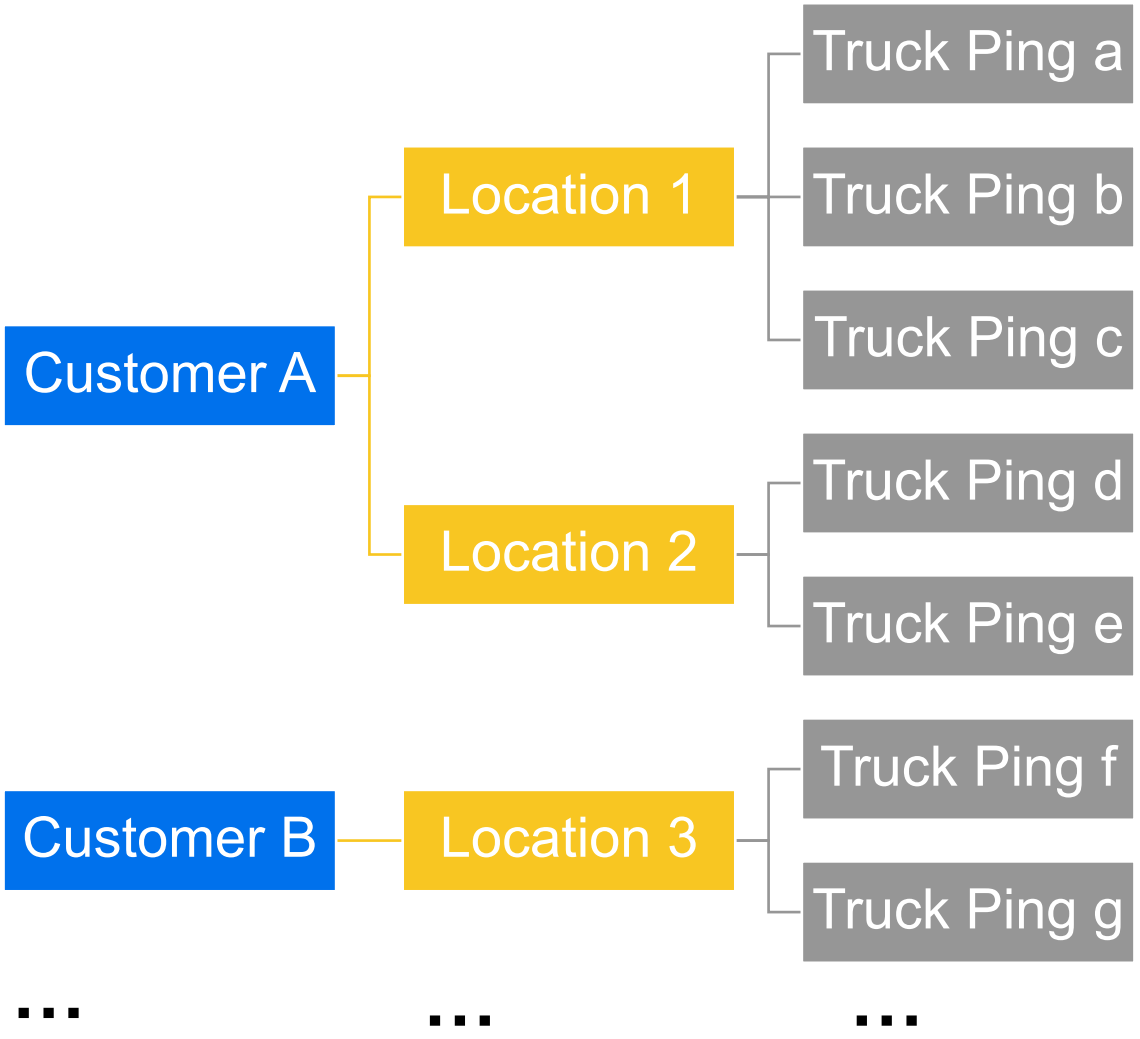
Too-large geofence means travels that are not linked to our location get captured by mistake

Effects:

- Arrivals get checked off that haven't happened yet
- Estimates of their next arrival time are STILL messed up
- We incorrectly mark shipments out of order / not following the order the schedule lists



Size of the Challenge



Could we draw geofences by hand?

What if we have thousands of customers, and millions of locations?

How many different truck GPS points are we going to have to handle?

How It Works

Applying Data Science Strategy

Concept

For this project, geofences are predictions of where vehicles will go when they are visiting the location in the future.

Approach

Find patterns in **historical data** and use them to predict **future events.**

Historical data: past traffic from customers' vehicles and shipments

- This tells us when they traveled near these locations and made a visit

Future Events: a visit to the location is happening

- We believe this scenario will have some commonality with past visits

Making a Geofence: Steps to Follow

1. Retrieve the center of the location (or as close as we can get)
2. Are there other stops nearby? If so, tighten up the perimeter we'll look at.
3. Extract all the telemetry we've got for this customer in the perimeter
4. Run DBSCAN clustering on the telemetry to produce clusters, and select the clusters that are relevant
5. Draw convex hull around the best clusters
6. Add buffer zone if desired

Examples of Cluster Filtering Rules

- Cluster must be not too far away from the center, or from some other feature
- Cluster must have a minimum number of points
- Range of cluster sizes must not exceed some value
- … any other criteria that's relevant to your use case

About DBSCAN

"Density-based spatial clustering of applications with noise"

TL;DR: looks at the points, finds regions with highest volume/most dense points, and groups these together. Identifies as outliers if points are not groupable with clusters.

Parameters

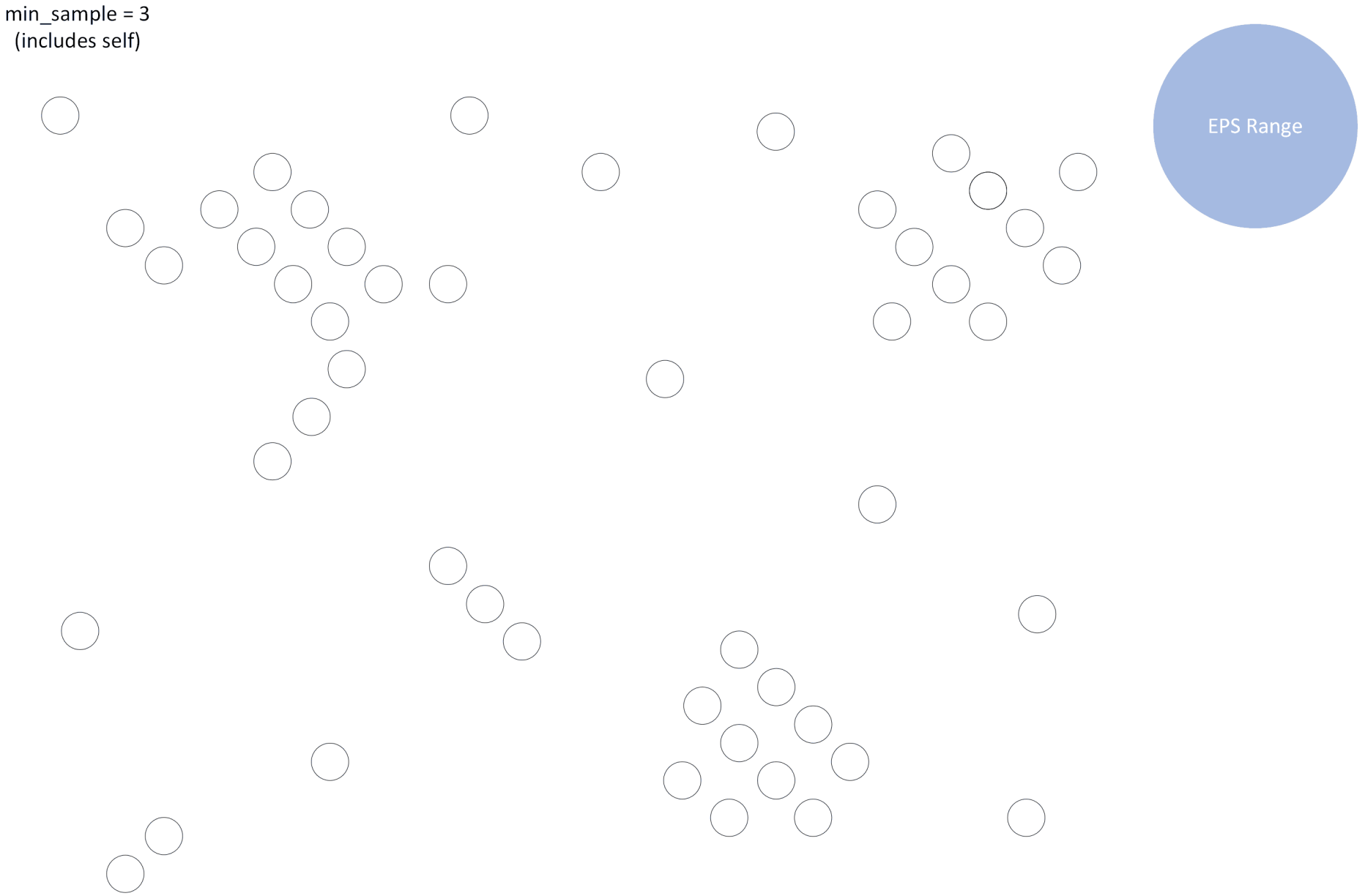
- ε : (epsilon) Defines the "neighborhood" size in which the algo will look
- *min_samples*: the number of points required for a point to be "core" potentially (default 5)

Steps

1. Examine each point, and count the other points falling in neighborhood (ε)
2. If a point has > *min_samples* neighbors, it's a "core" point.
3. Determine core connected components, which become building blocks of clusters
4. Non-core points are assigned to these clusters if in a neighborhood (ε); otherwise labeled "outliers".

DBSCAN Process

1. Identify core points
2. Group into connected components
3. Add qualifying noncore points
4. Remainder are outliers



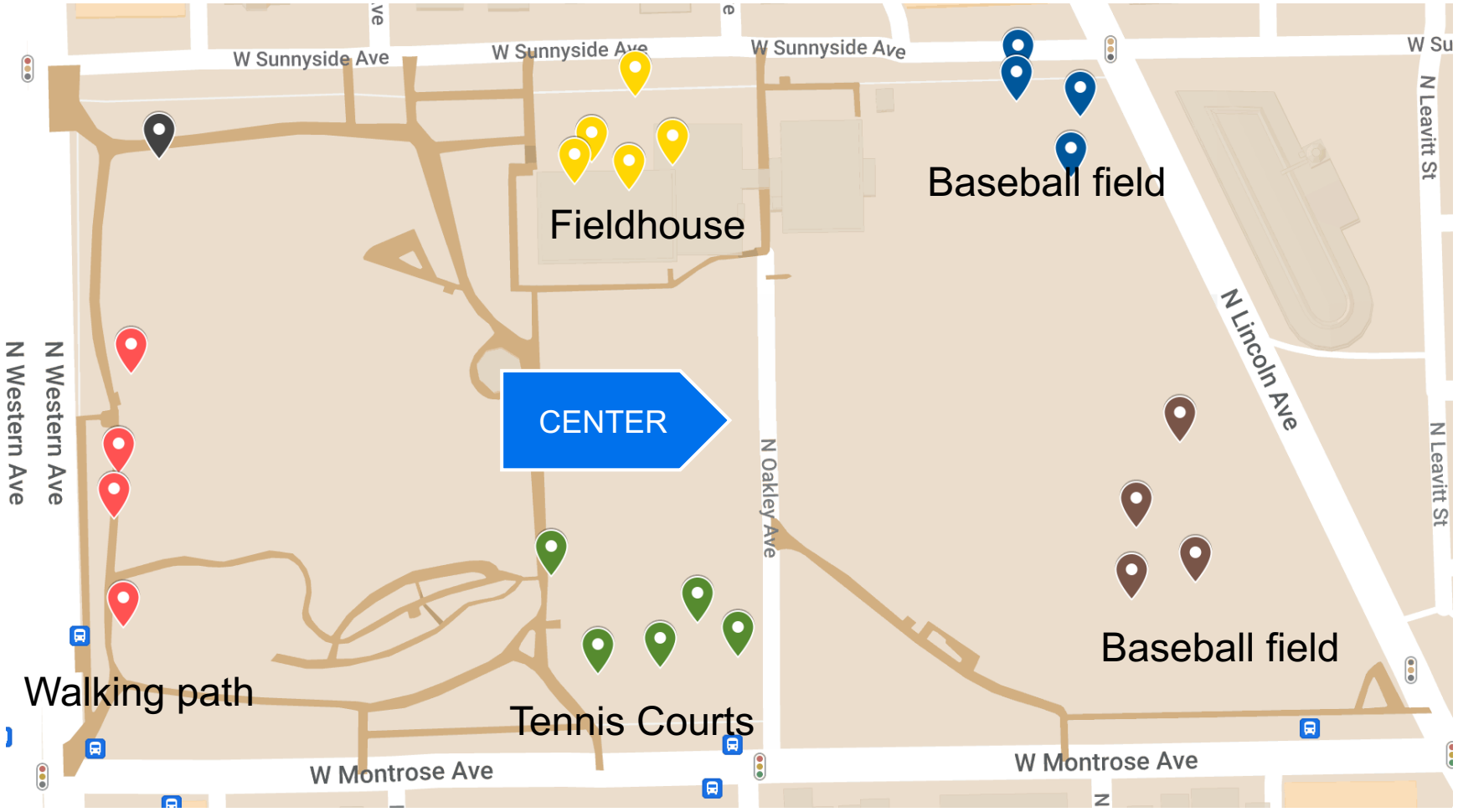
Points are spread in 2-dimensional space.

Find Clusters

Look around from the center and collect all our points.

DBScan groups clusters together for us.

This represents where people tend to gather, and thus may be where people will gather in the future.

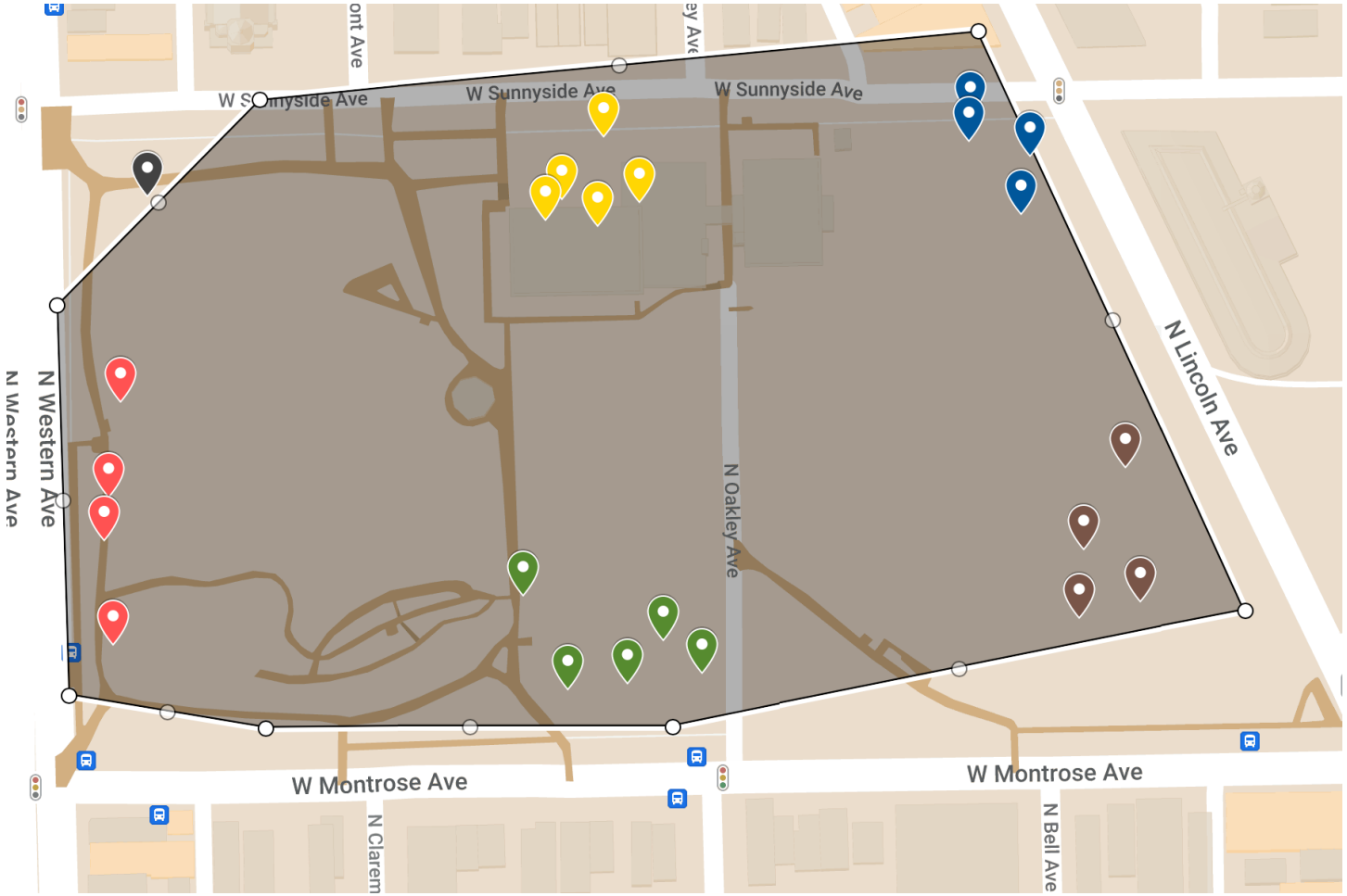


(These simplified examples are hand drawn, and not actually generated with algorithm, because the data is all made up)

Convex Hull

Draw the full perimeter, and don't include the outlier, just suitable clusters.

It's a little tight, and misses some of the edges, but it captures most of the park, and definitely captures the most high-traffic sections.

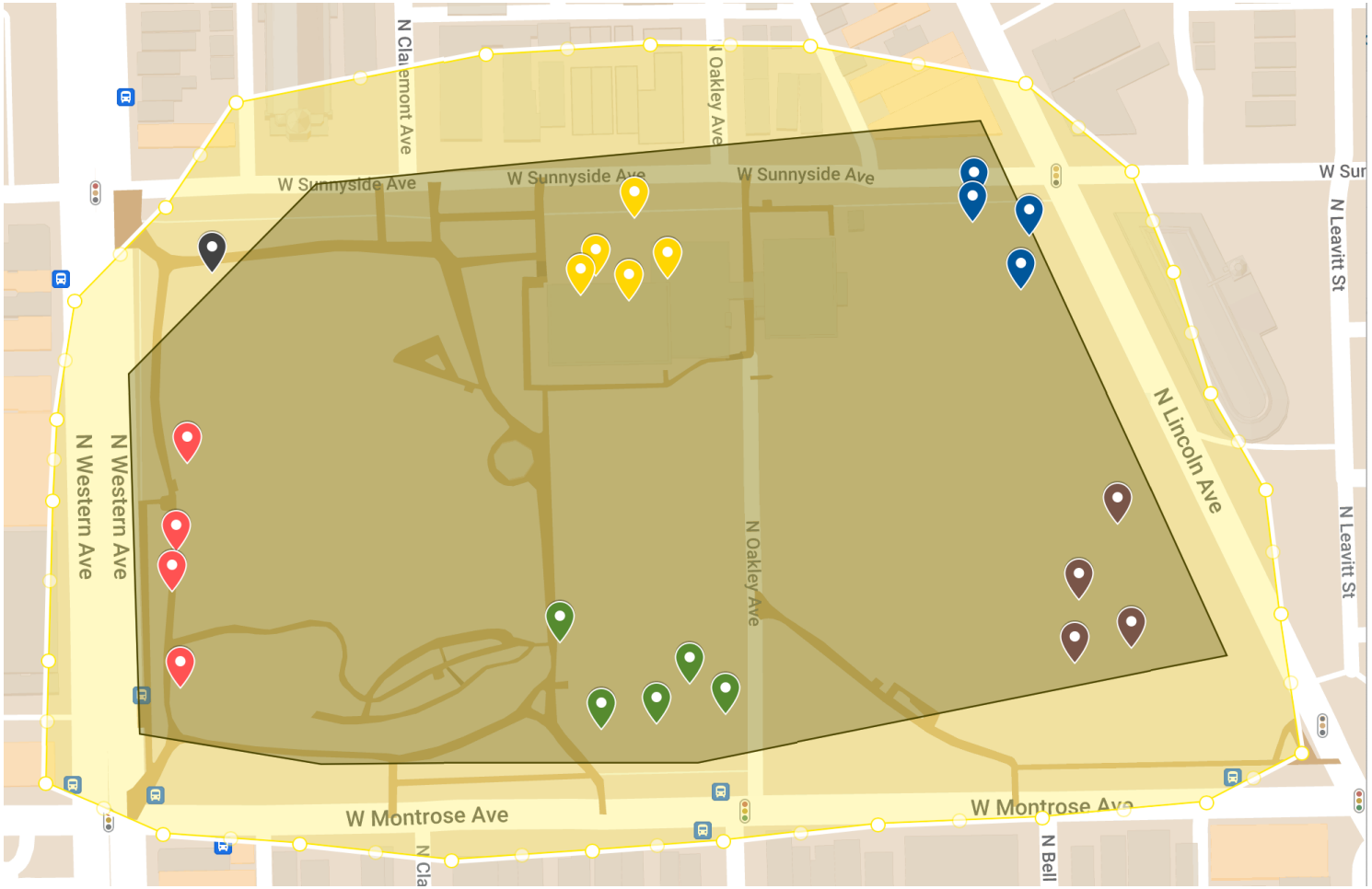


(These simplified examples are hand drawn, and not actually generated with algorithm, because the data is all made up)

Add Buffer

To accommodate any uncertainty, add a bit of space all the way around that perimeter.

This lets us cover any missed spots on the borders, where traffic is less busy but visits may still happen.



(These simplified examples are hand drawn, and not actually generated with algorithm, because the data is all made up)

Results

Our geofence shape ends up sort of lumpy, but it captures much of the real nuance of the park shape.

Most importantly, it is reflective of the spaces where visitors actually go.

This can work for any location where enough GPS telemetry represents real human movements.

(These simplified examples are hand drawn, and not actually generated with algorithm, because the data is all made up)

Deployment

Deployment Considerations

Selecting Cases

- Our business case means we can skip very underutilized places
- Saves money and compute resources

Database Resources

- We need to be careful about how much and how often we hit the Snowflake database
- Worked with DE team to estimate costs

Parallelization

- Dask is great for parallelism, but we have to limit our use due to the I/O
- May need to run millions of locations through

Scheduling

- Use Airflow to schedule and slack hooks keep us informed at all times
- Chain tasks to run many customers in 1 dag

Underutilized Locations

Most of our customers have a long-tail distribution of locations they use- some are regularly seen (like warehouses and distribution centers) and some are rarely visited (grandma doesn't order stuff that much).

By skipping very rarely visited locations, we prioritize the locations our customers use most.

- ~58% of our recorded customer locations have been visited at least once in 2022
- This means we can skip 42% of locations for now, and save a ton of compute
- If those locations are visited in the future, we'll generate new geofences for them then

Deployment Considerations

Selecting Cases	Database Resources	Parallelization	Scheduling
• Our business case means we can skip very underutilized places • Saves money and compute resources	• We need to be careful about how much and how often we hit the Snowflake database • Worked with DE team to estimate costs	• Dask is great for parallelism, but we have to limit our use due to the I/O • May need to run millions of locations through	• Use Airflow to schedule and slack hooks keep us informed at all times • Chain tasks to run many customers in 1 dag

Snowflake I/O

When this job runs, it makes a query for each location we're analyzing – this means many very small queries, but the volume (hundreds of thousands or millions of locations) can create backlog and queuing of Snowflake jobs.

- Our DE team made a warehouse for data science scheduled jobs like this one so we aren't causing everyone else pain
- We also limit how many queries we send to the Snowflake warehouse through our parallelization settings

Deployment Considerations

Selecting Cases

- Our business case means we can skip very underutilized places
- Saves money and compute resources

Database Resources

- We need to be careful about how much and how often we hit the Snowflake database
- Worked with DE team to estimate costs

Parallelization

- Dask is great for parallelism, but we have to limit our use due to the I/O
- May need to run millions of locations through

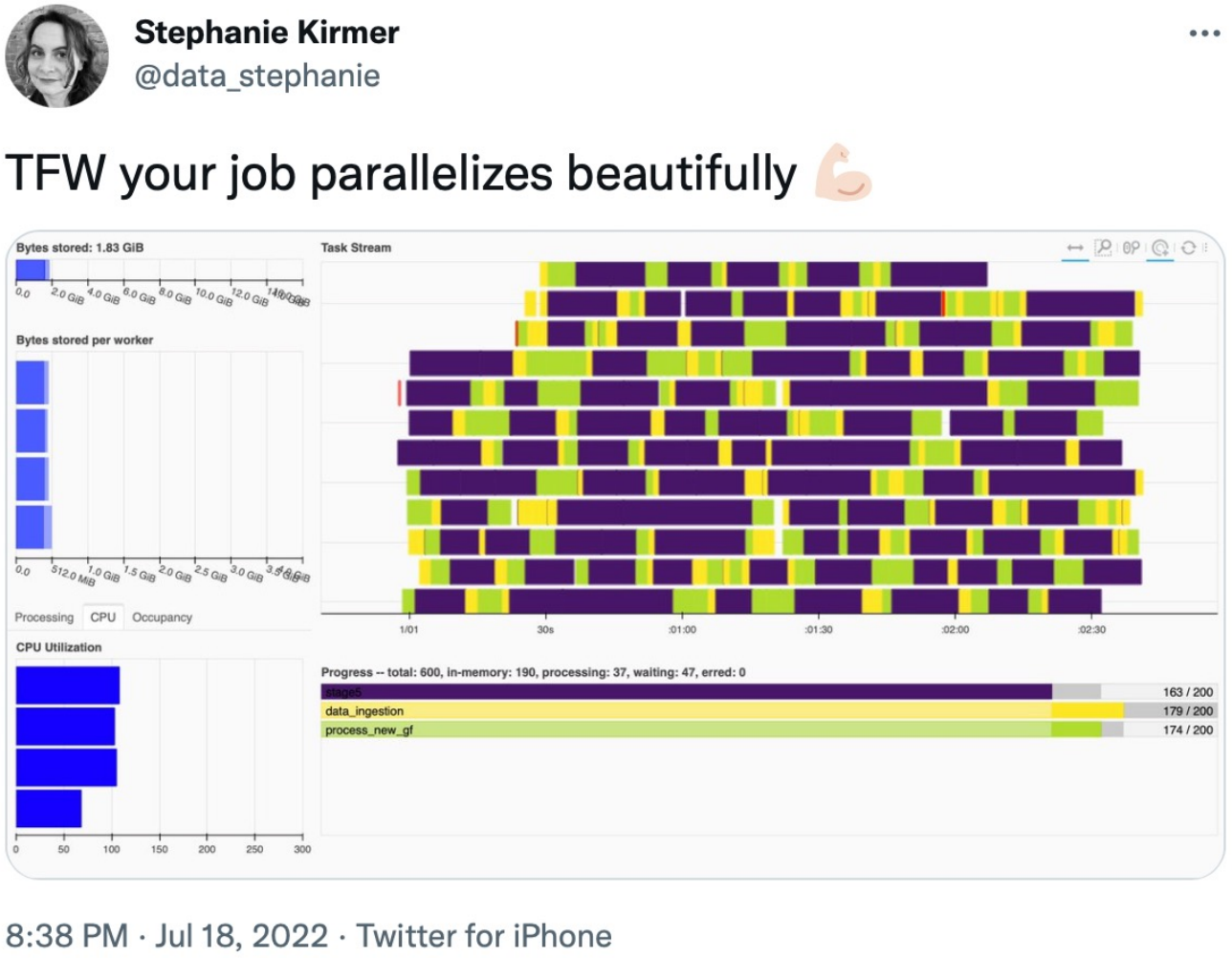
Scheduling

- Use Airflow to schedule and slack hooks keep us informed at all times
- Chain tasks to run many customers in 1 dag

Dask Parallelization

To maximize our use of the CPU available, I applied Dask to parallelize the different location analyses.

I originally used 16 threads (as shown) but the Snowflake I/O limitations meant I had to downgrade to just 4 threads. Even so, that still provides significant efficiency gains.



Deployment Considerations

Selecting Cases

- Our business case means we can skip very underutilized places
- Saves money and compute resources

Database Resources

- We need to be careful about how much and how often we hit the Snowflake database
- Worked with DE team to estimate costs

Parallelization

- Dask is great for parallelism, but we have to limit our use due to the I/O
- May need to run millions of locations through

Scheduling

- Use Airflow to schedule and slack hooks keep us informed at all times
- Chain tasks to run many customers in 1 dag

Airflow Task Chaining

We don’t want the customer jobs to run in parallel because of I/O considerations already discussed – so we use chaining and programmatically generate jobs while still making them run serially.

- Using k8s Operator lets us manage dependencies completely with Docker and not have to install a bunch of junk on Airflow box
- Callbacks at task level let us be notified in Slack whenever one is successful or fails.

(Some boring default args omitted for space)

```
with models.DAG(dag_id="generate_geofence_series") as
   dag:

   tasklist = []
   for tenant_name, tenant_id in tenant_dict.items():

      run_pod = KubernetesPodOperator(
        task_id=f"generate_geofences_{tenant_name}",
        name=f"generate_geofences_{tenant_name}",
        cmds=["poetry"],
        arguments=[
        "run",
        "python3",
        "geofence_generator_dask.py",
        "--tenant_id", f"{tenant_id}",
        ],
       startup_timeout_seconds=120,
       resources={"limit_memory": ”40G"},
       image="smartgeofences:latest",
       on_failure_callback= am.on_failure_callback,
       on_success_callback= am.on_success_callback
      )
   tasklist.append(run_pod)

chain(*tasklist)
tasklist[-1]
```

Final Process

Select customer location	• If it hasn't been scheduled in a year, we skip
Pull all relevant GPS into python	• Anywhere from ~100 to 70,000 points per location
Cluster with DBSCAN	• Drop unwanted clusters based on parameter settings
Draw convex hull and add buffer with GeoPandas	• Customizable buffer size, usually ~200 m
Run QA	• Soft delete malformed shapes if we detect
Write geofence to postgres prod database	• When new GPS is received, serve geofence as comparison

How It’s Going

Our customers really love the results they have been seeing- we are deploying to more customers all the time.

We have measured the impact on shipments in terms of capturing arrival milestones and other metrics, and results have been very good!

We are continuing to monitor results, and also exploring additional applications for the technique.

Thanks!

Questions?

Contact Me:

stephanie@stephaniekirmer.com

www.stephaniekirmer.com

@data_stephanie

Learn more about p44: www.project44.com