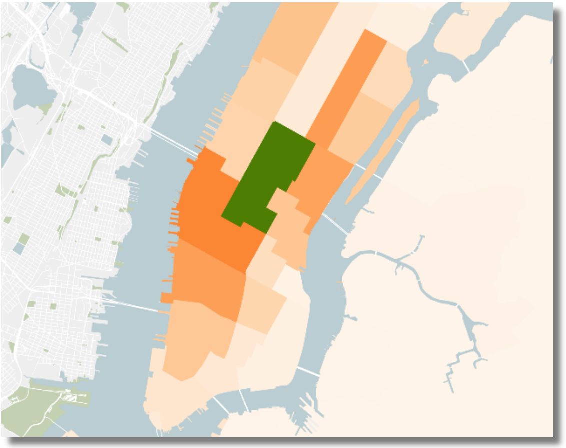
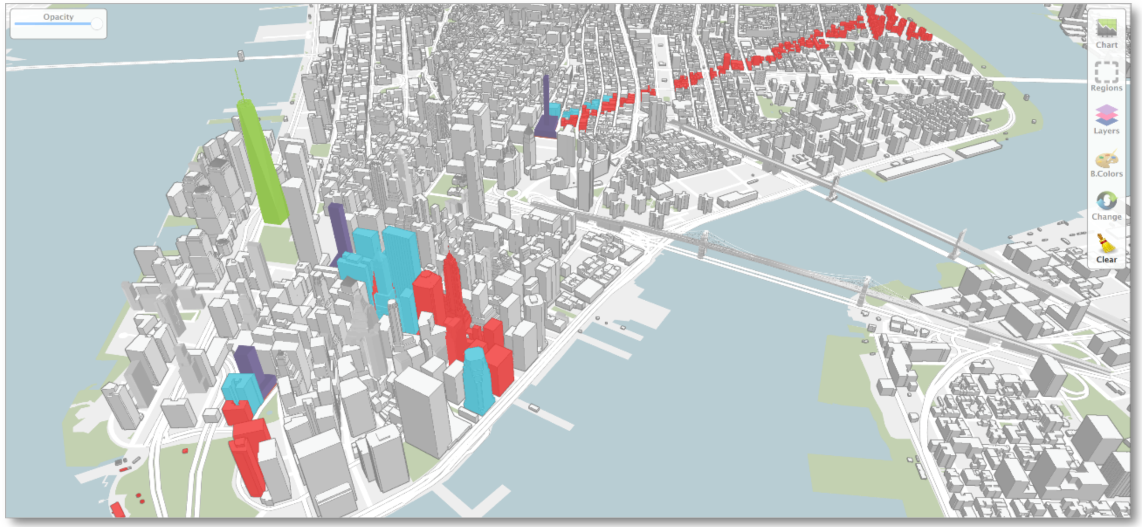



# Efficient Query Processing for Spatial and Temporal Data Exploration

**Eleni Tzirita Zacharatou**

Advisor: Anastasia Ailamaki

Public Thesis Defense

09.08.2019

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

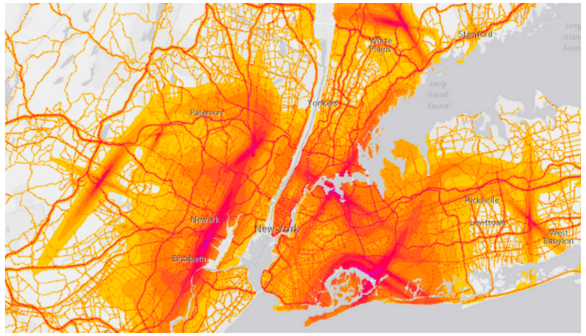
EPFL

# Urbanization and City Planning

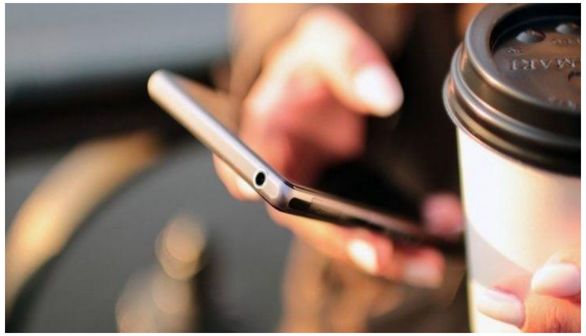
# Data Exhaust from Cities

*Infrastructure*


*Environment*



*People*



# Understanding Cities through Data

NYC OpenData

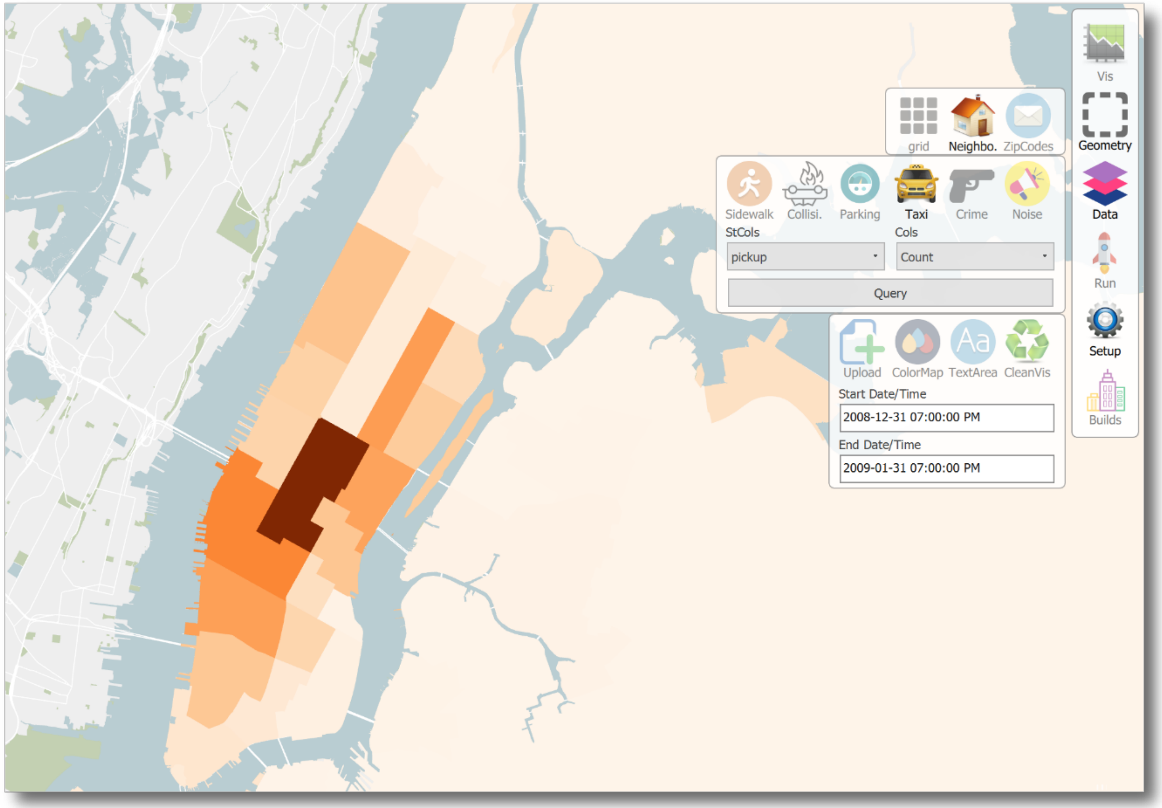
GOVDATA
Das Datenportal für Deutschland

data.gouv.fr

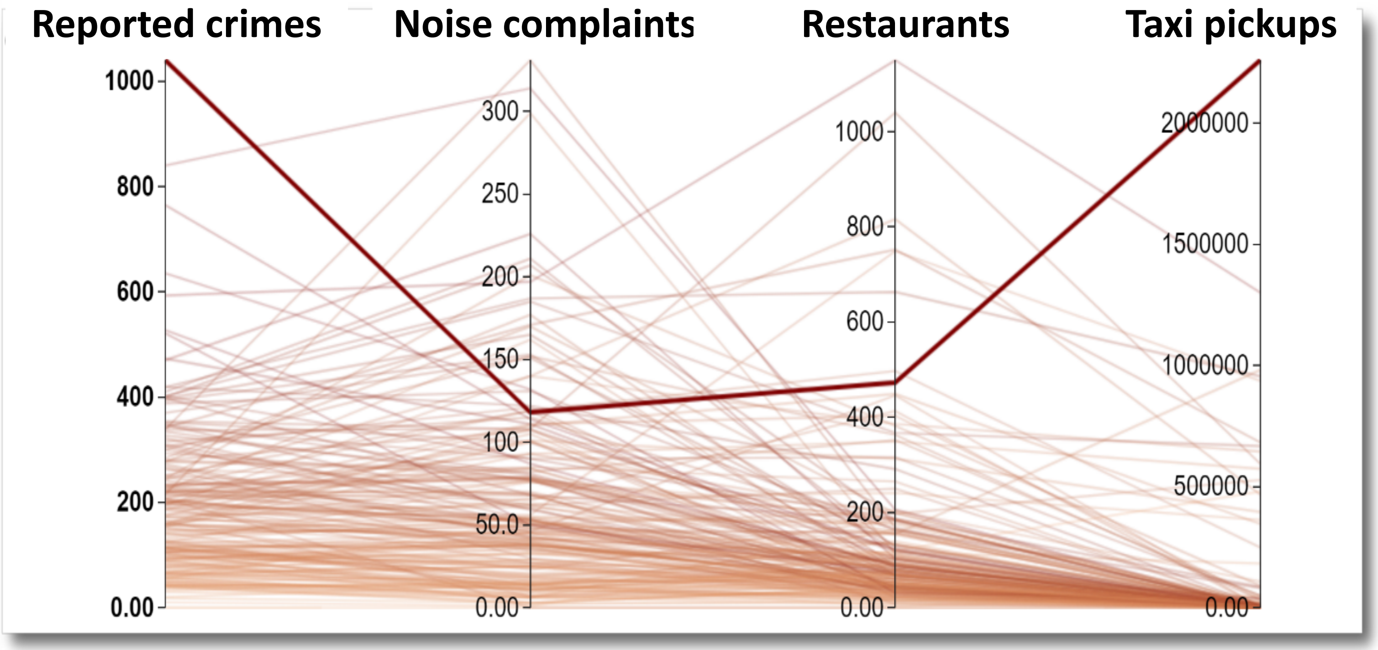
opendata.swiss

**Opportunity: Data-driven urban planning**

# Visual Spatial Data Exploration




Distribution of taxi pickups per neighborhood in Manhattan

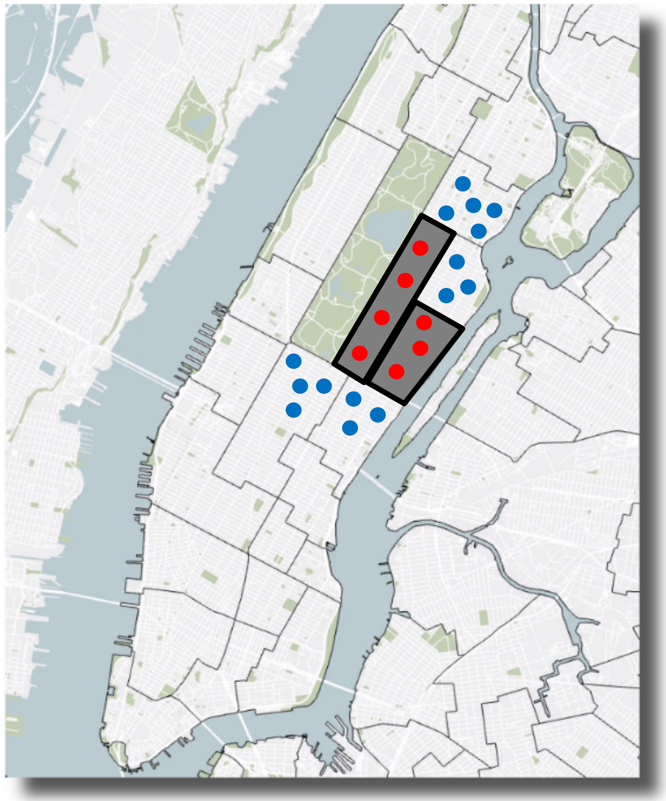



Comparison of different urban datasets

**Need: Interactive response times**

# Spatial Aggregation Queries

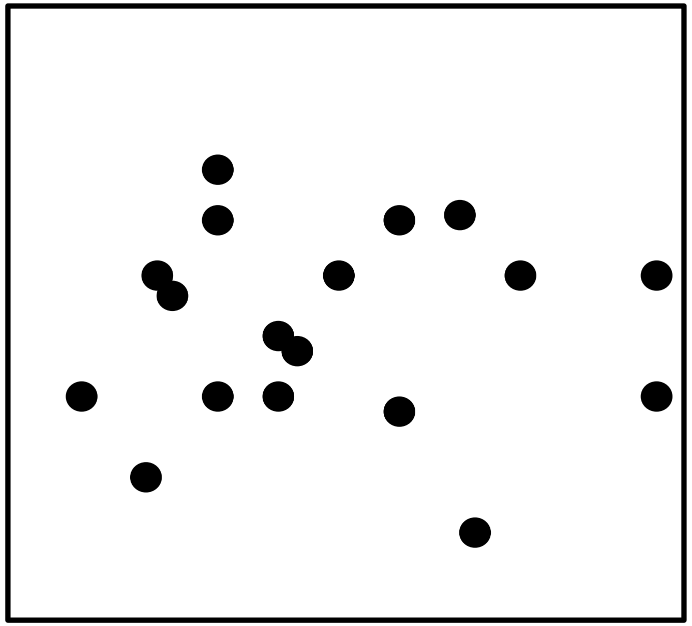
| | |
|---|---|
| Aggregation | SELECT COUNT(*) |
| Input | FROM taxi ride *T*, neighborhoods *N* |
| Spatial Join | WHERE *T*.pickup INSIDE *N*.geometry |
| Selection | AND *T*.picktime in January 2009 |
| Grouping | GROUP BY *N*.id |



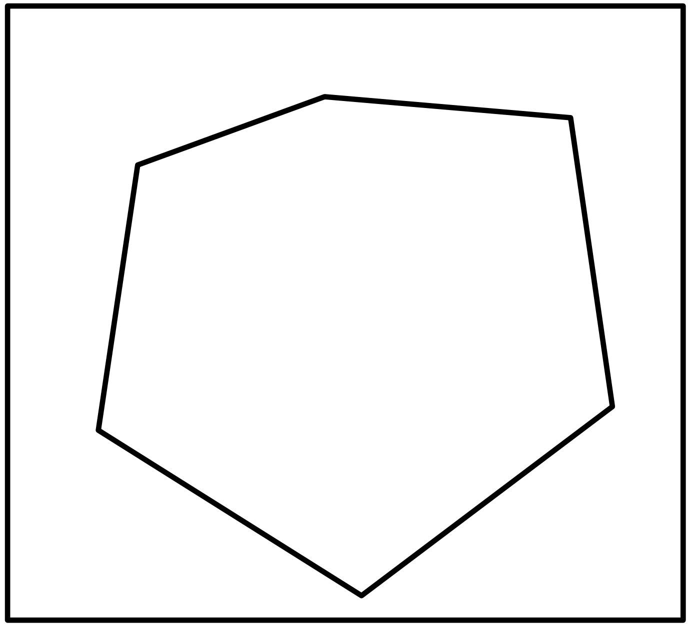
Point-in-Polygon tests

**Expensive Point-in-Polygon tests → High latency (minutes)**

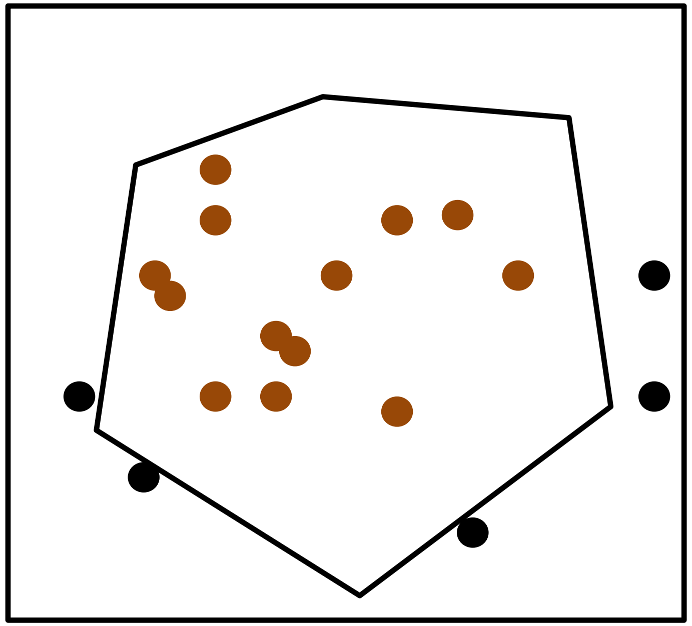
# Spatial Aggregation: a Geometric Perspective



Input points



Input polygon

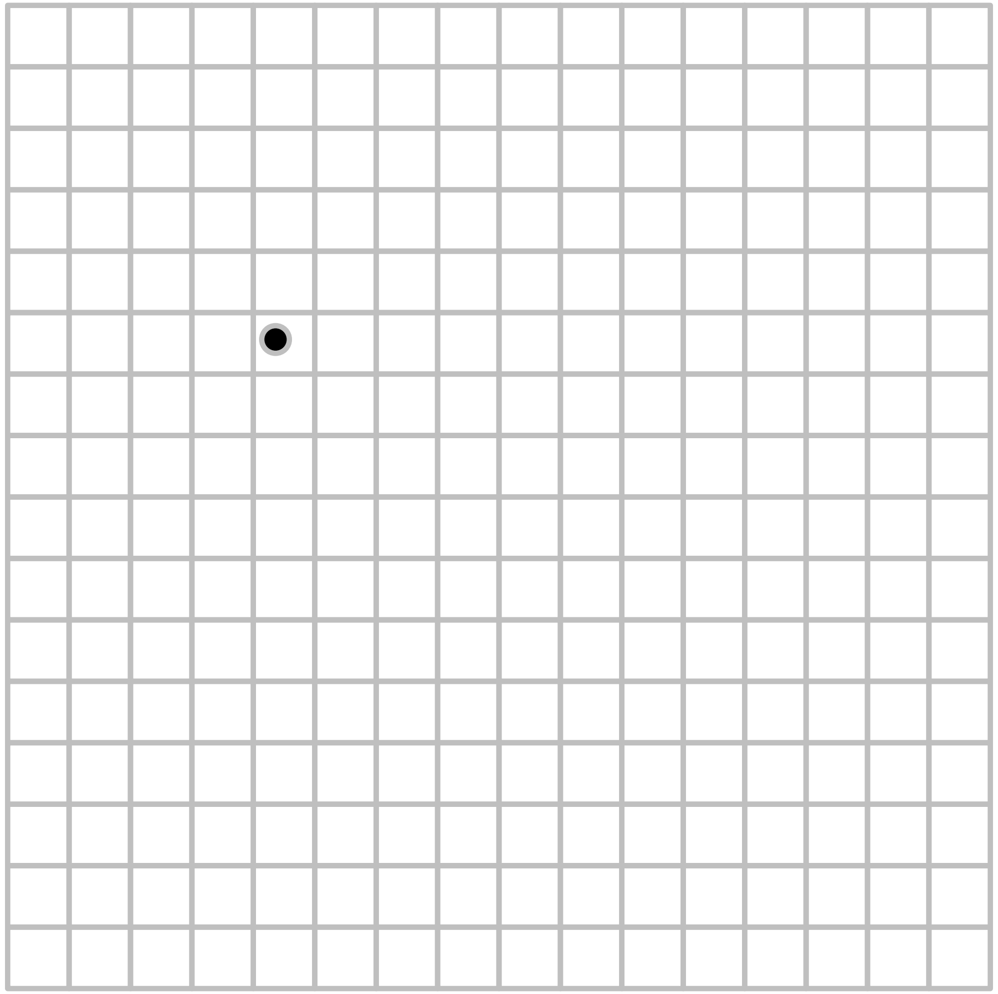


Spatial join

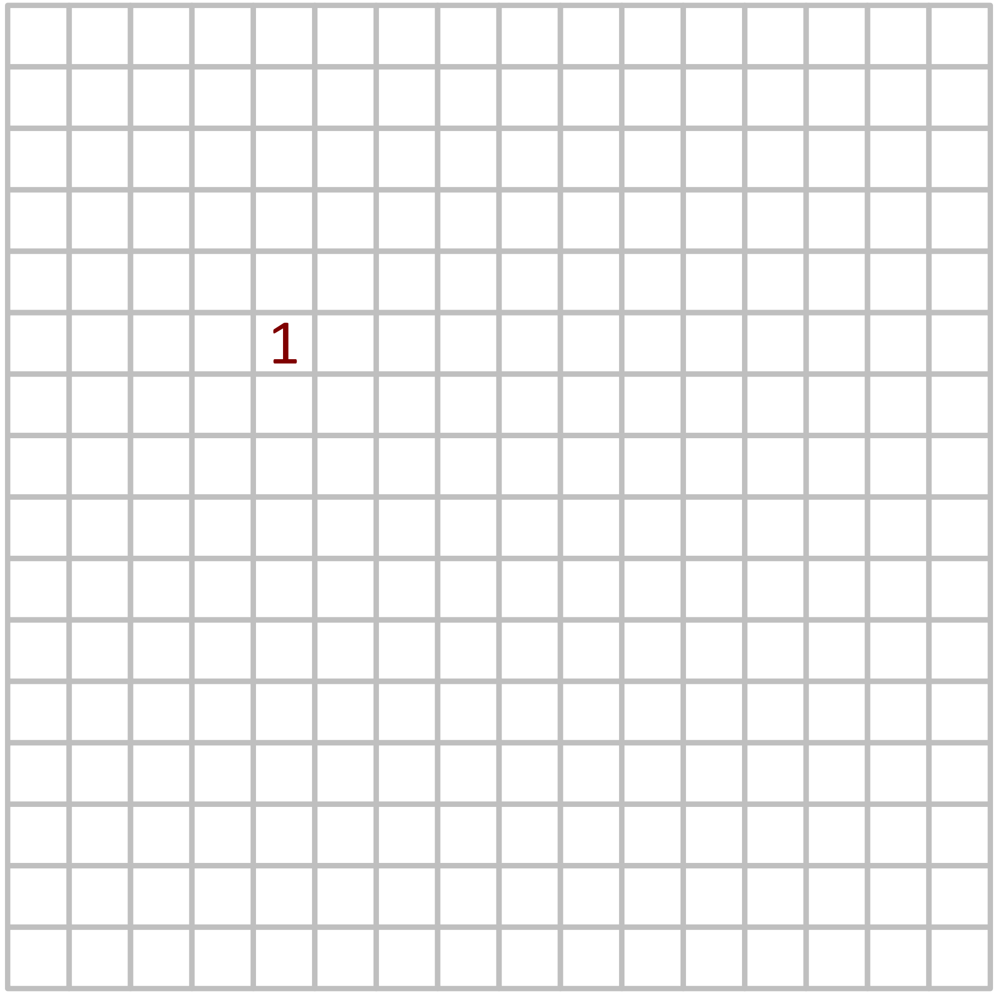
**"Drawing" on the same canvas**

**→ Leverage the graphics pipeline of the GPU**

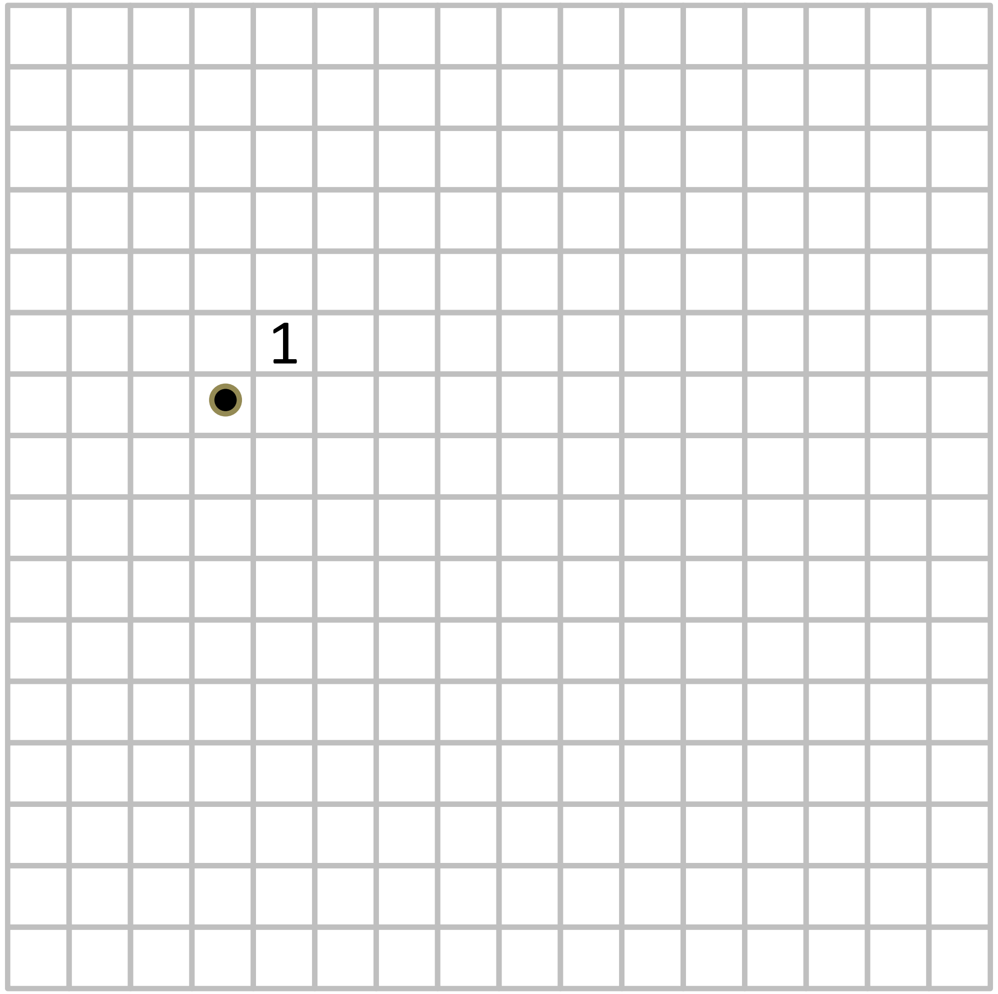
# Raster Join - Step 1: Draw the Points



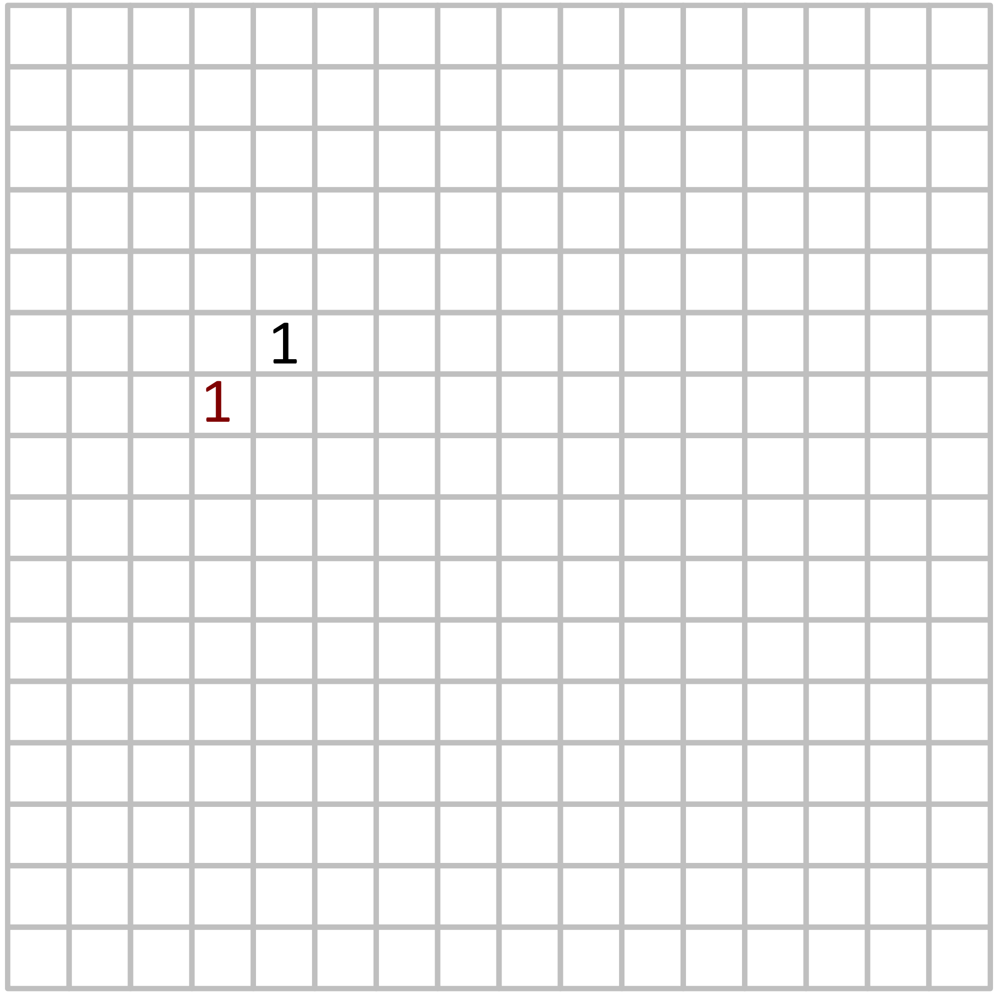
# Raster Join - Step 1: Draw the Points



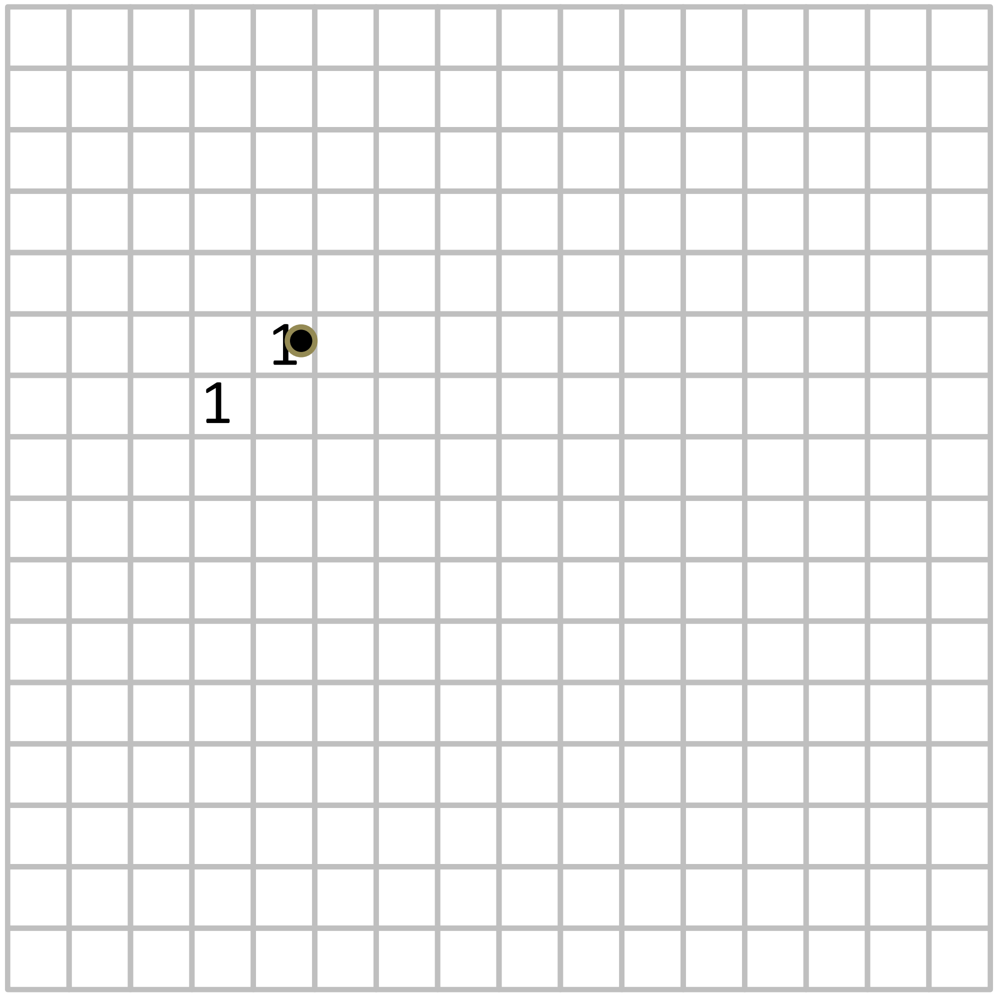
# Raster Join - Step 1: Draw the Points



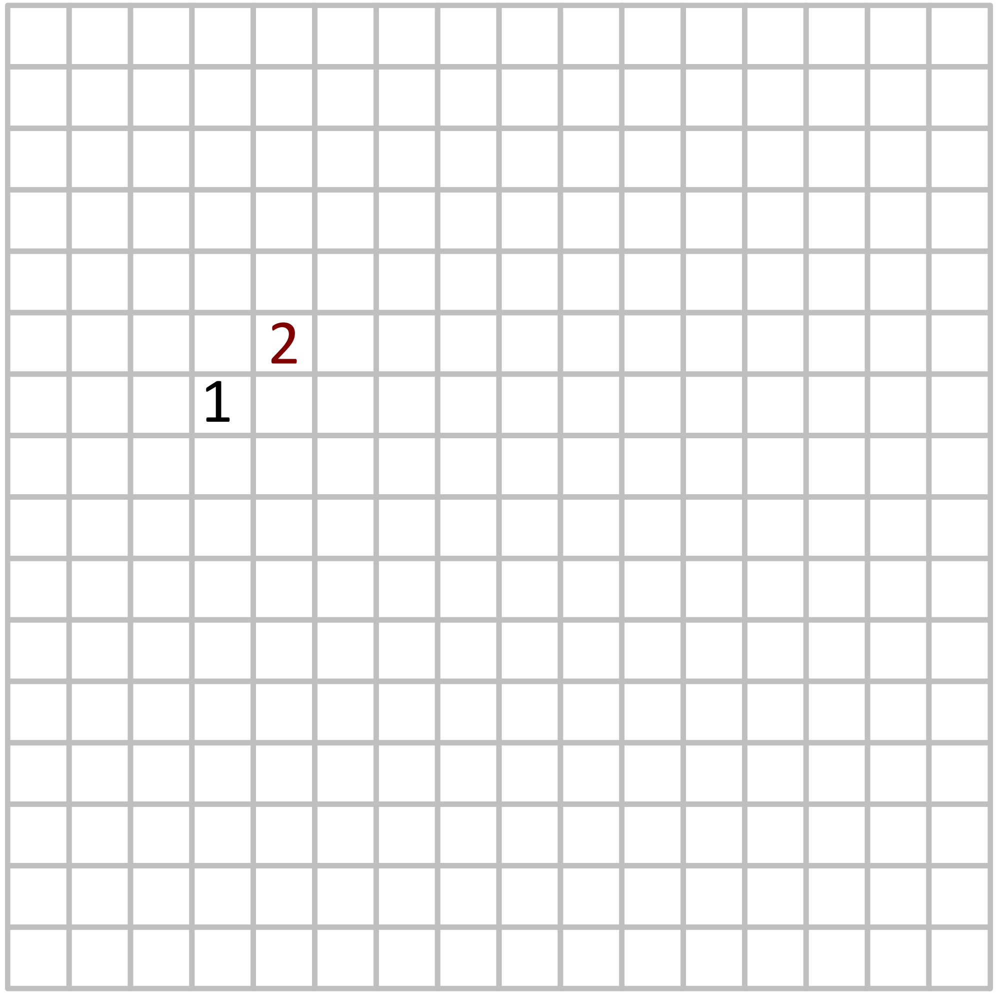
# Raster Join - Step 1: Draw the Points



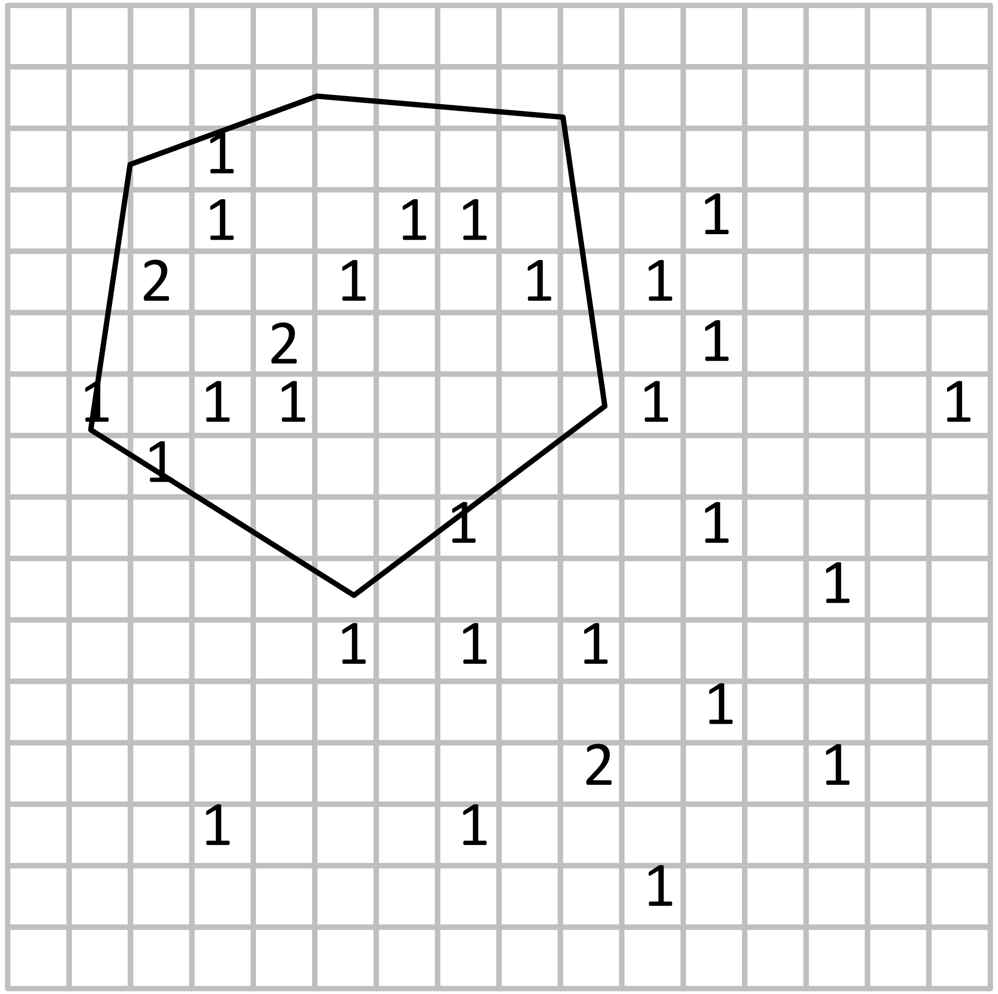
# Raster Join - Step 1: Draw the Points



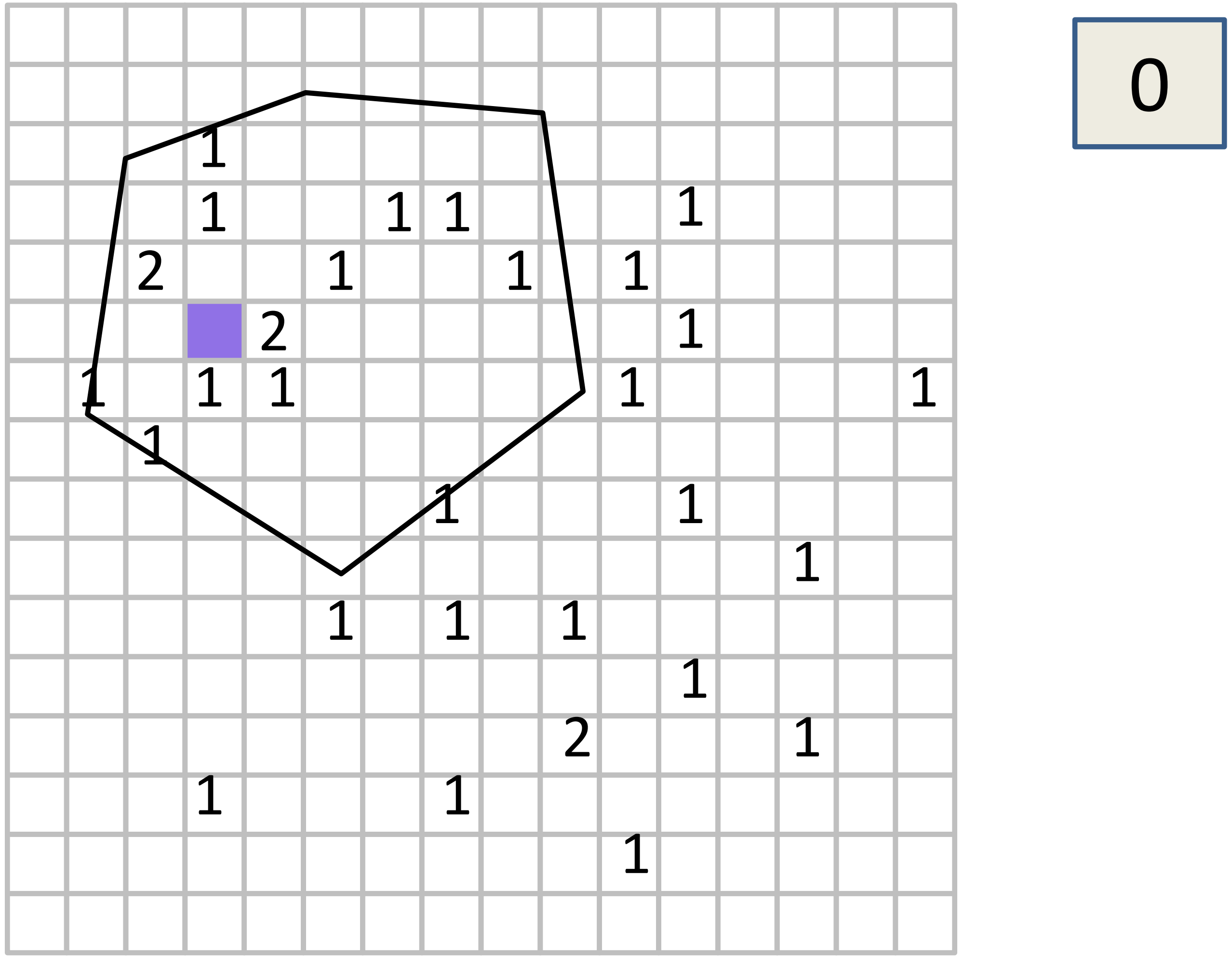
# Raster Join - Step 1: Draw the Points



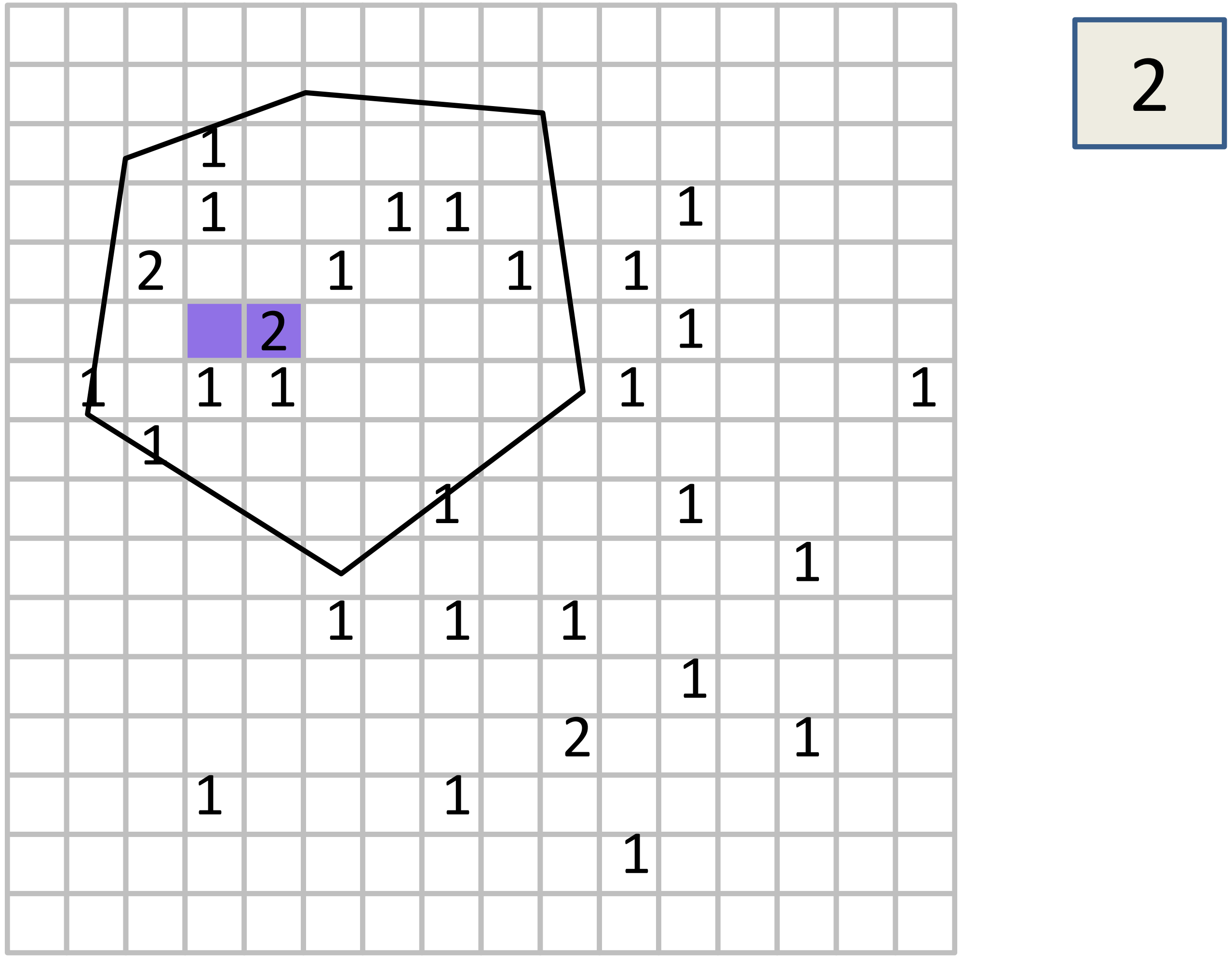
# Raster Join - Step 1: Draw the Points



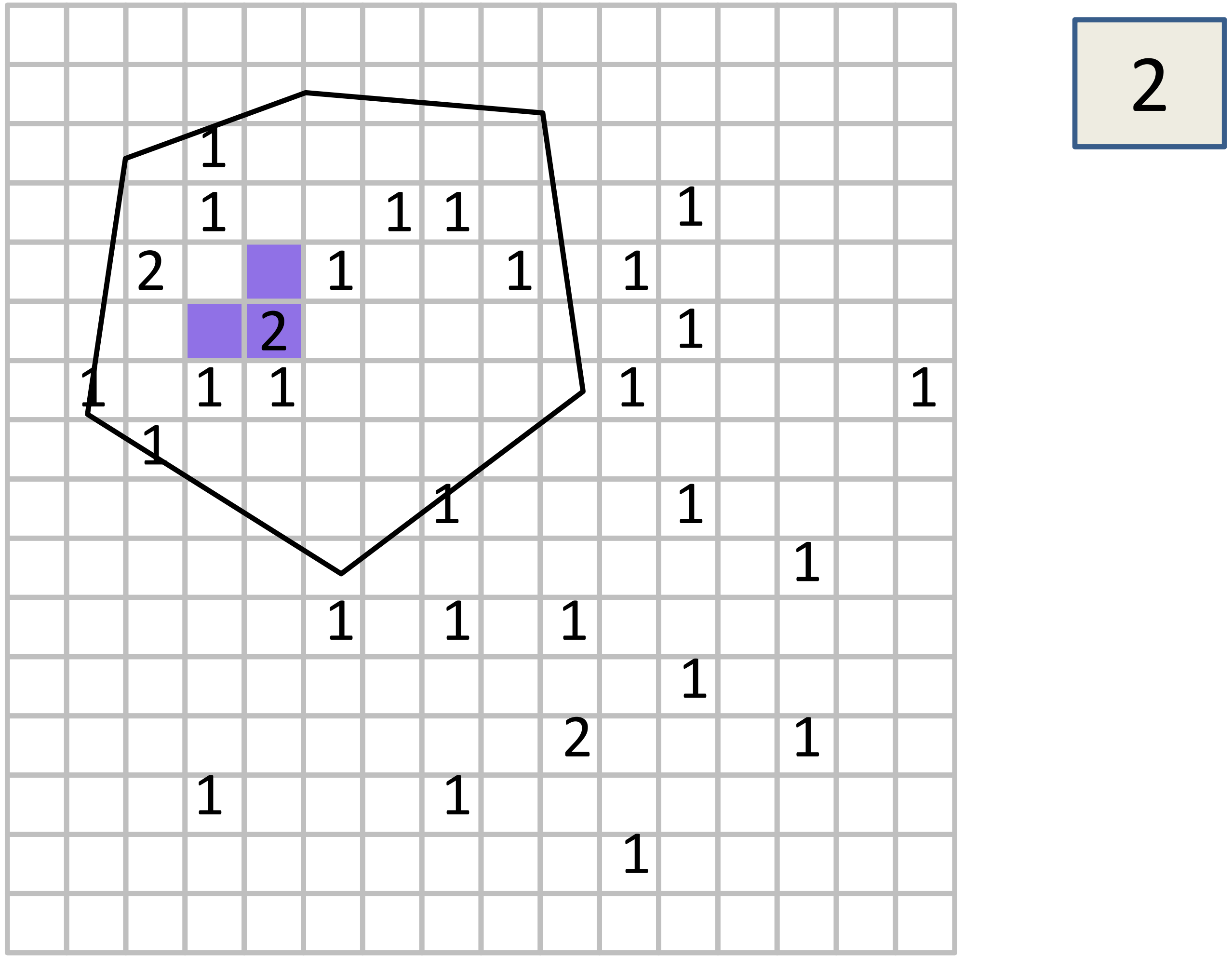
# Raster Join - Step 2: Draw the Polygons



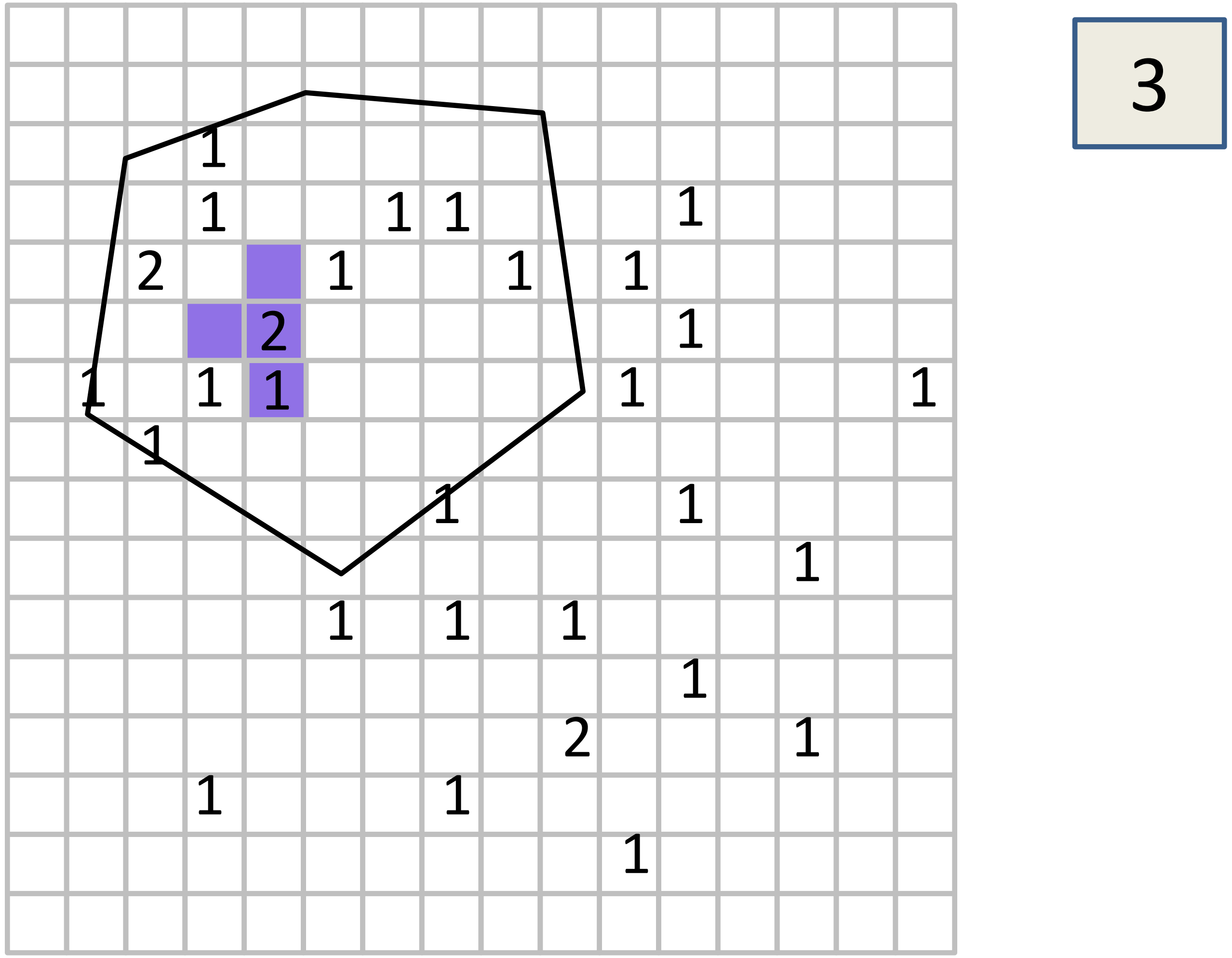
# Raster Join - Step 2: Draw the Polygons



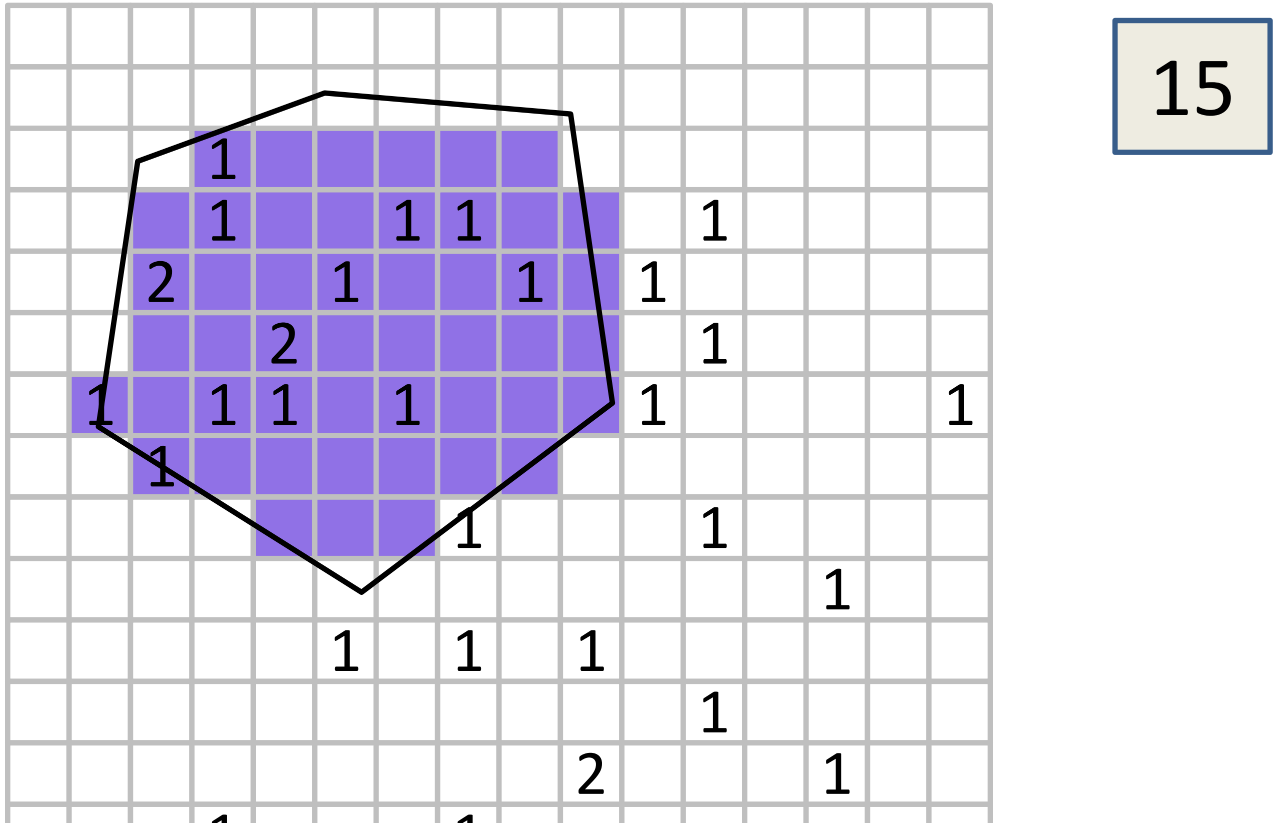
# Raster Join - Step 2: Draw the Polygons



# Raster Join - Step 2: Draw the Polygons



# Raster Join - Step 2: Draw the Polygons



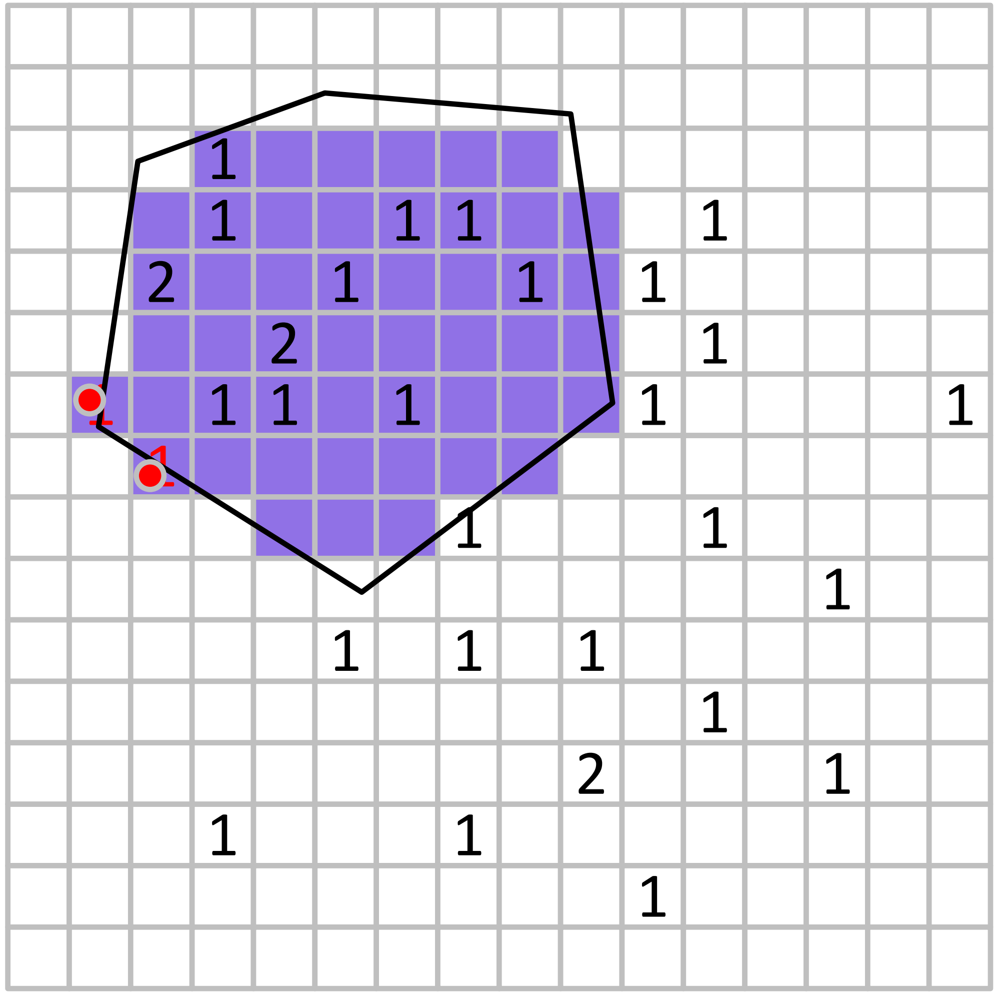

**Uses native support for drawing in GPUs**
**Combines the aggregation with the join operation**
**No Point-in-Polygon tests**

# Bounding the Approximation Error

- Bound the Hausdorff distance between the approximate (purple) and the original polygon.

$$H(P_a, P) \leq \varepsilon$$

- Smaller pixel size → higher accuracy.

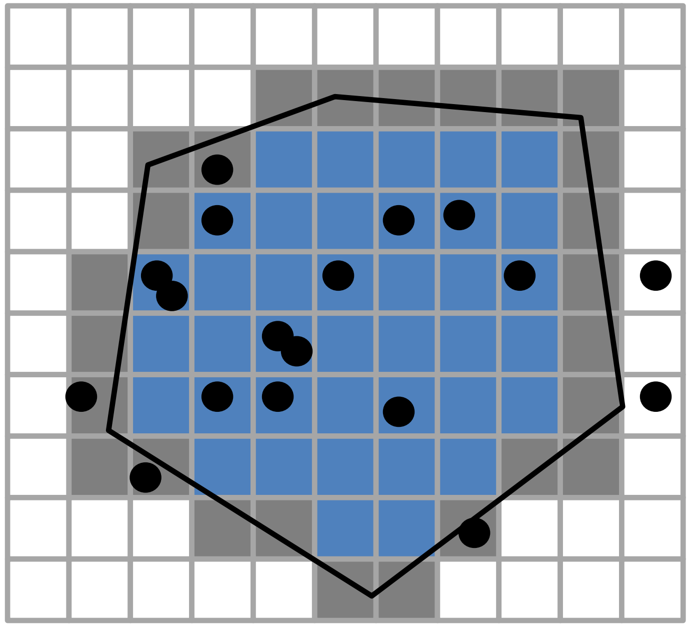


**Trade accuracy for response time**

# Hybrid Raster Join: an Accurate Technique

Blue pixels - completely inside the polygon: store count

Grey pixels - polygon boundary: Point-in-Polygon (PiP) tests



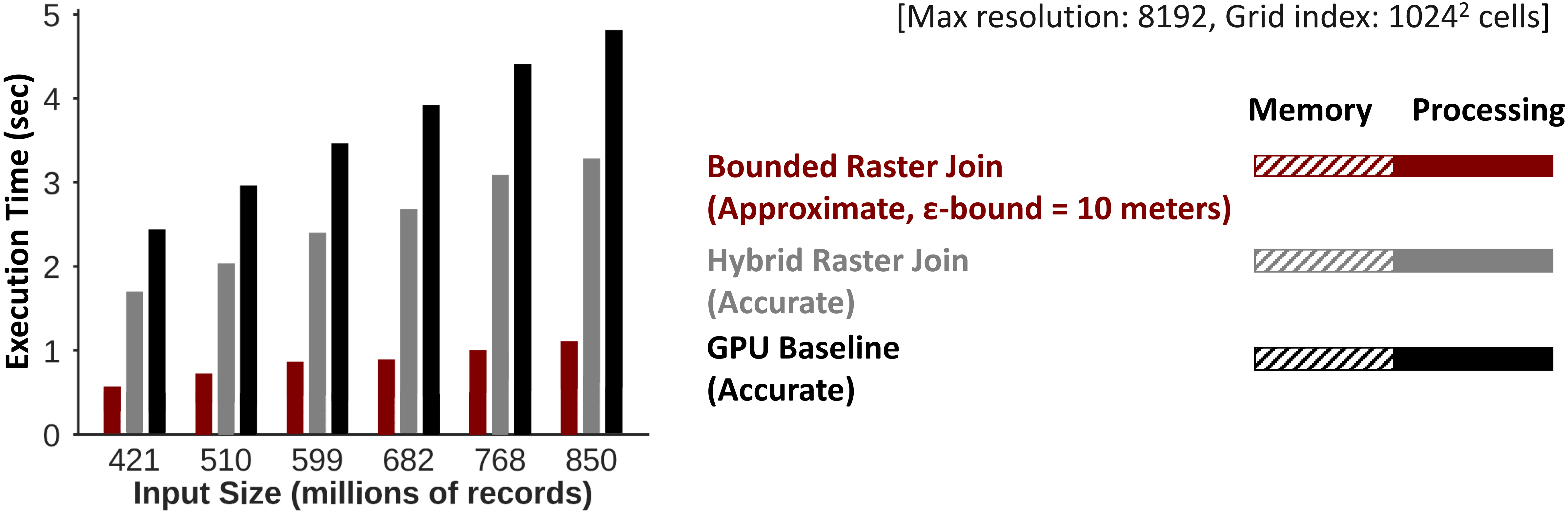
**Extra computation: identifying the boundary & performing PiP tests**

# Scaling with Increasing Data Sizes

**COUNT Taxi rides (points) GROUP BY NYC Neighborhoods (260 polygons)**

[Intel Core i7 Quad-Core CPU @ 2.80GHz, 16GB RAM, NVIDIA GTX 1060 GPU, 6GB memory (using only 3GB)]

[Max resolution: 8192, Grid index: $1024^2$ cells]




**Bounded Raster Join is 4X faster than GPU Baseline**

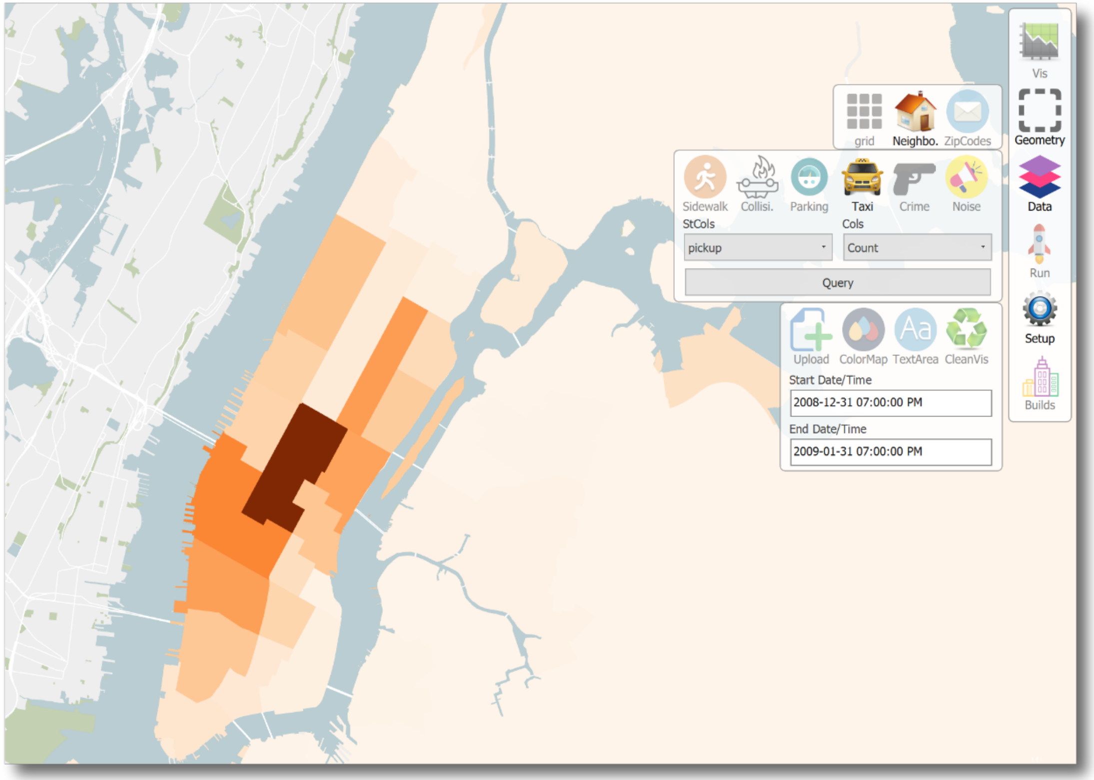
**CPU-GPU data transfer takes a significant amount of time**

# GPU Rasterization enables Interactive Spatial Queries

[VLDB18, SIGMOD18]

- Express queries as graphics primitives and use modern GPUs

- Aggregating 850M taxi records over NYC neighborhoods in ~1 second

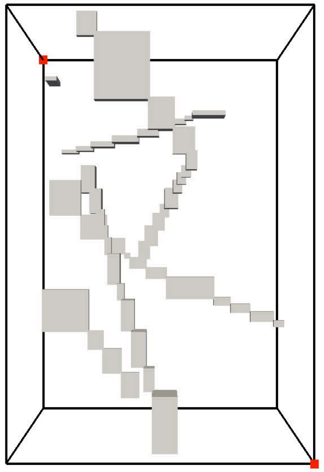


# Clipped Minimum Bounding Boxes for Efficient Spatial Indexing

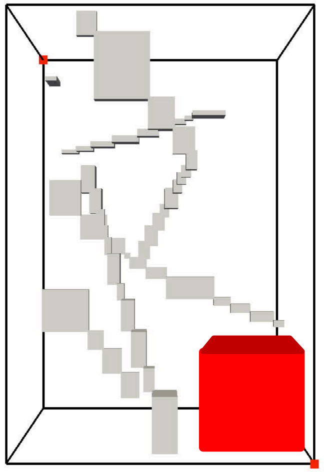
[ICDE18]

Improve precision by subtracting out empty bounded areas

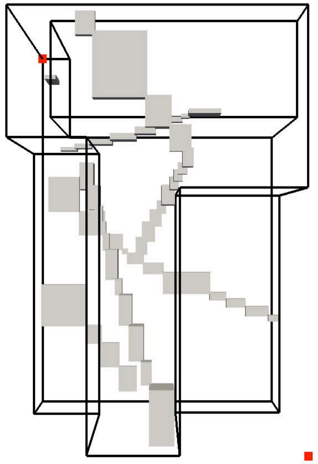
➔ Answering a spatial range query on 1B objects in less than 200ms



**Minimum Bounding Box**



**Empty space ➔ Ineffective filtering**



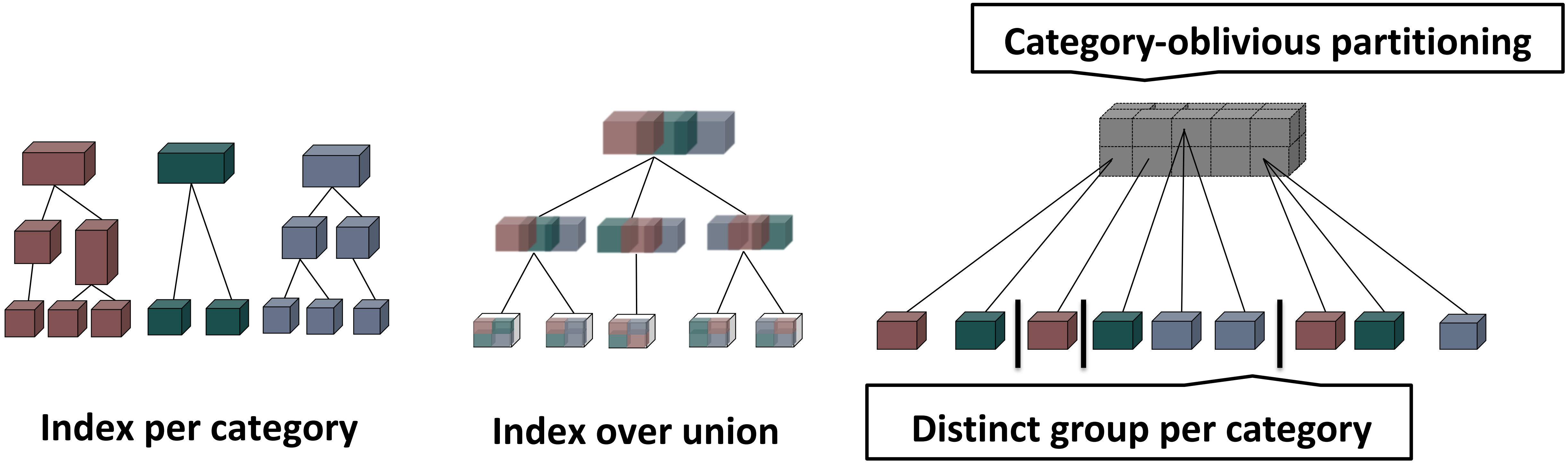
**Clipped Bounding Box**

# Workload-Aware Indexing enables Ad-hoc Spatial Queries

[ExploreDB16]

Category-aware spatial data organization

➔ Up to 12.3X faster queries on 10 different neuron categories

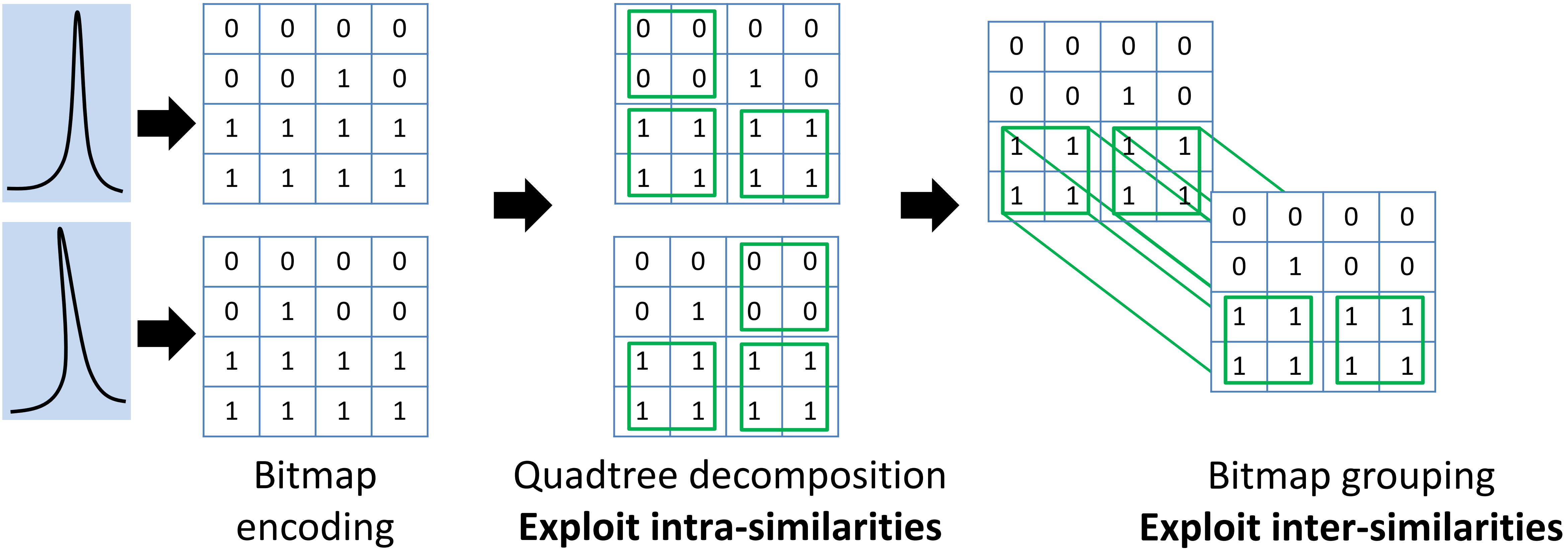


# Quadtree Bitmap Decomposition for Scalable Time Series Indexing

[SSDBM15]

Value-time searches exploiting space-time similarity

➔ From 9X to 23X faster queries on neuroscience data



# Thesis Statement

Modern applications need to explore large amounts of spatial and temporal data at interactive speeds, challenging traditional query processing techniques that rely on time-consuming computations and inefficient access methods.

*Query operators* that exploit **specialized hardware** and **workload-aware** *access methods* enable scalable and interactive exploration of spatial and temporal data.

# Looking Ahead

- **Approximation-based spatial data processing**
  - Fine-grained approximations and omission of exact geometric tests
  - Distance-based error bound
  - Trade precision / storage space for performance

- **Utility of graphics techniques for spatial data processing**
  - GPU rasterization for real-time approximation
  - 3D Join → Collision Detection

**Thank you!**