CS 686: Special Topics in Big Data

Managing Geospatial Data

Lecture 19

Today's Agenda

- Q&A from previous class
- Introduction to spatiotemporal data
- P2 dataset specifics
- Geohash
- Spatial Indexes
- Spatial Queries

Today's Agenda

- **Q&A from previous class**
- Introduction to spatiotemporal data
- P2 dataset specifics
- Geohash
- Spatial Indexes
- Spatial Queries

Q&A

- Who invented MapReduce?
  - Functional programming languages, way back…
  - Google popularized it
  - Hadoop made it pervasive
- MapReduce 2.0?
  - Hadoop: YARN (Yet Another Resource Negotiatior)
  - Splits up resource management across the cluster
  - Still very much the same old paradigm

Another Question: Why?

- Google has stated that they no longer use MapReduce
  - **But** I am sure the paradigm itself (perhaps not the implementation) is alive and well at Google!
- Spark has recently become more popular for these types of analyses
  - Allows in-memory working sets
- Hadoop, like many "cool" technologies, was over-prescribed
  - It's still a great (the best?) option for processing gigantic amounts of data in a batch fashion

One More Thing

- Turn in:
  - Design doc
  - Project retro
- !!!

Today's Agenda

- Q&A from previous class
- **Introduction to spatiotemporal data**
- P2 dataset specifics
- Geohash
- Spatial Indexes
- Spatial Queries

Spatiotemporal Data

- One of the many sources of big data is ***spatiotemporal*** datasets
- These datasets are multidimensional:
  1. Space (geographic location, x-y coordinate, etc.)
  2. Time (could be years, days, even microseconds)
- Besides space and time, a spatiotemporal data point isn't very useful without additional ***features***:
  - Name, Age, ID
  - Speed, Weight, Direction

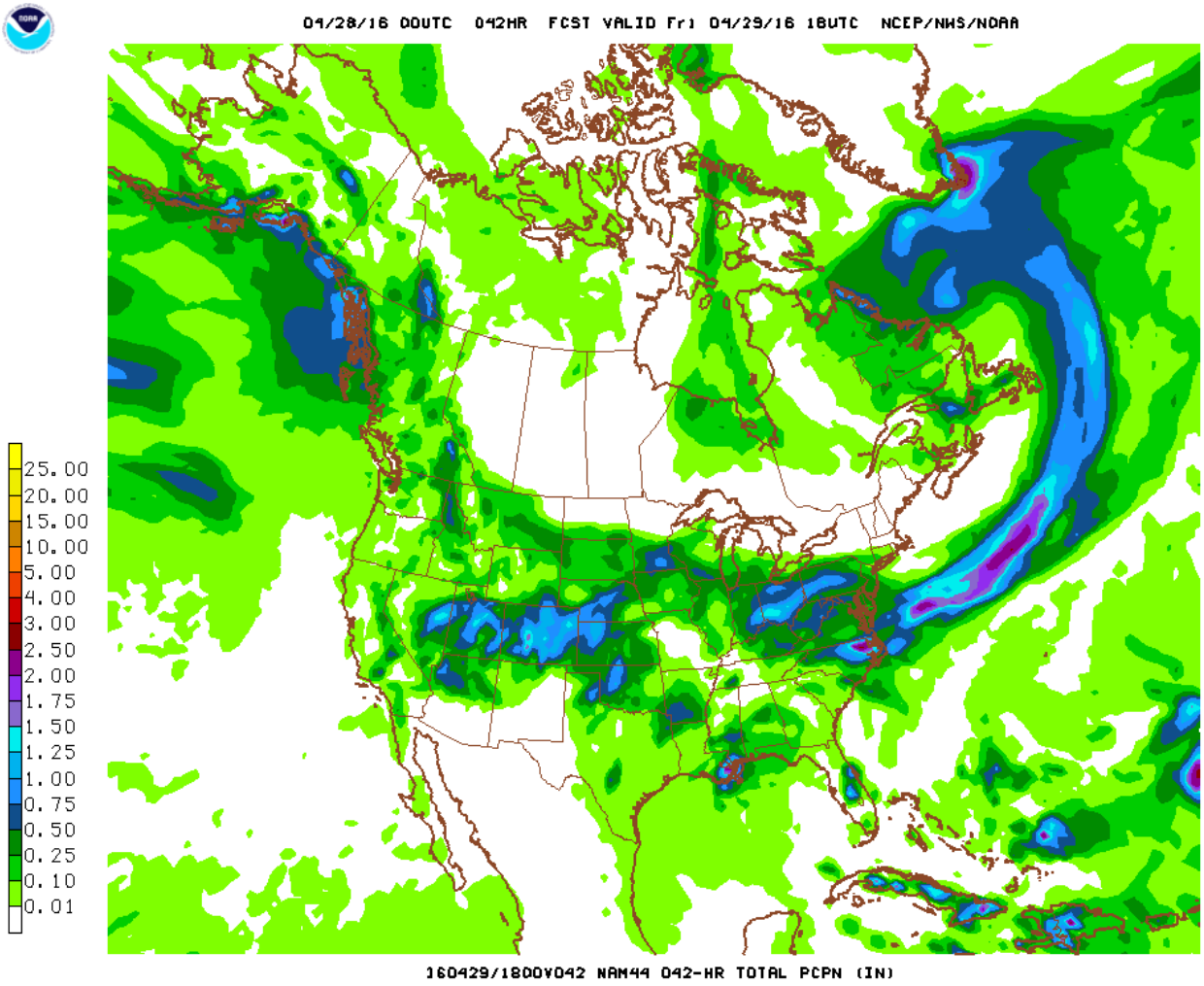
Spatiotemporal Data Sources

- Geographic information systems
  - Electric usage in a city over time
- Object tracking systems
  - GPS, atomic clocks, speed, direction
- Multiplayer games
  - Player location, attributes
- Networked sensors and radars
  - Temperature sensor with Wi-Fi connectivity
  - Cloud cover or reflectivity readings

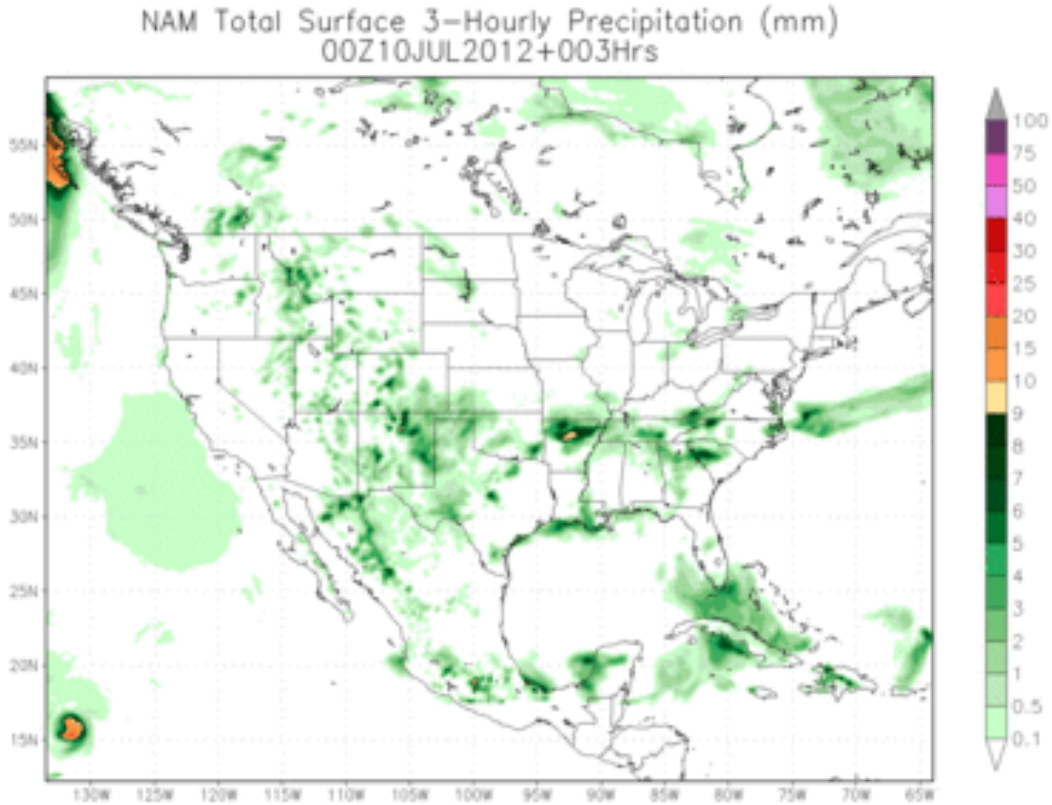
P2 Motivation: NOAA Dataset

- Sourced from NOAA NAM Project
- Some Dimensions/Features:
  - Geospatial: Latitude, Longitude
  - Time Series: Start Time, End Time
  - Temperature
  - Relative Humidity
  - Wind Speed
  - Snow Depth
- ~2 PB (about 20 billion files)

NAM Precipitation Snapshot



Animation


NAM Total Surface 3-Hourly Precipitation (mm)
00Z10JUL2012+003Hrs
55N
50N
45N
40N
35N
30N
25N
20N
15N
130W
125W
120W
115W
110W
105W
100W
95W
90W
85W
80W
75W
70W
65W
100
75
50
40
30
25
20
15
10
9
8
7
6
5
4
3
2
1
0.5
0.1

Learning More

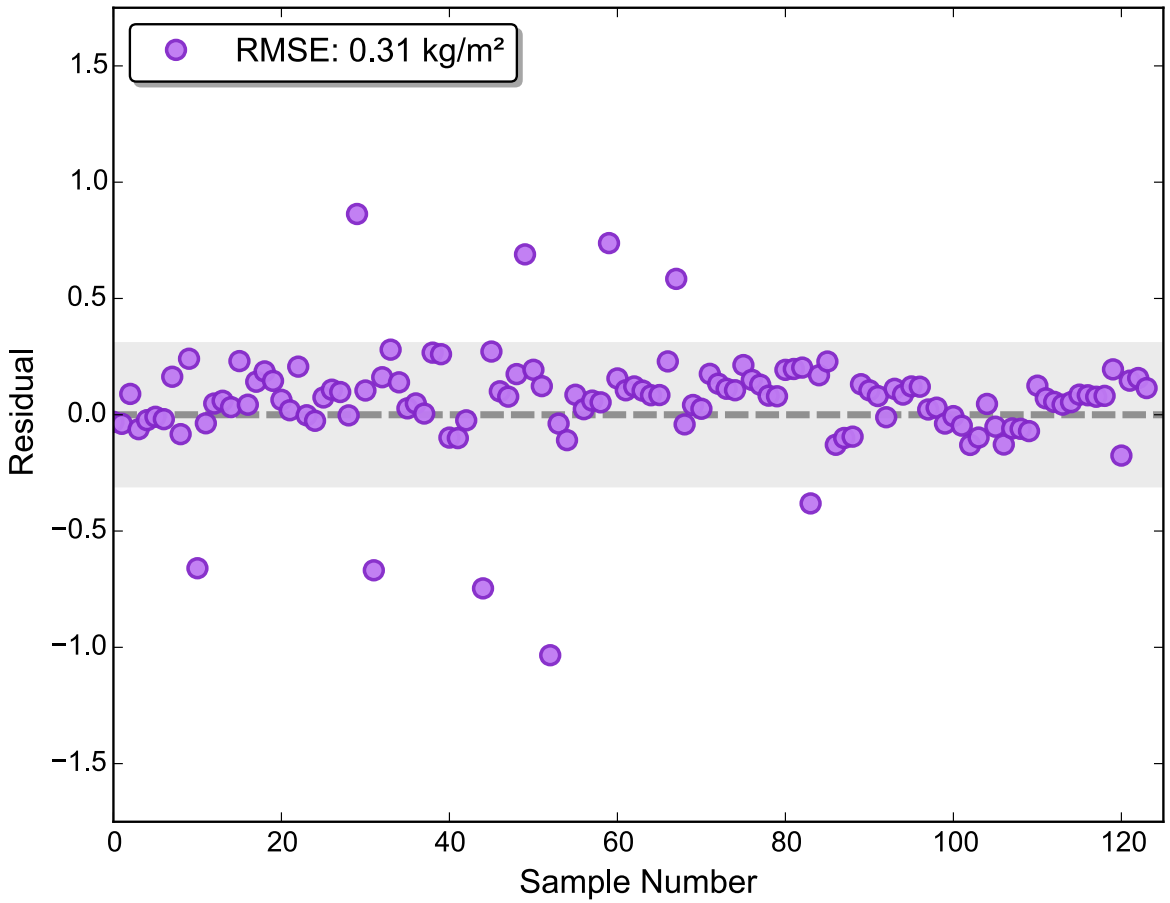
- See:

  https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/north-american-mesoscale-forecast-system-nam

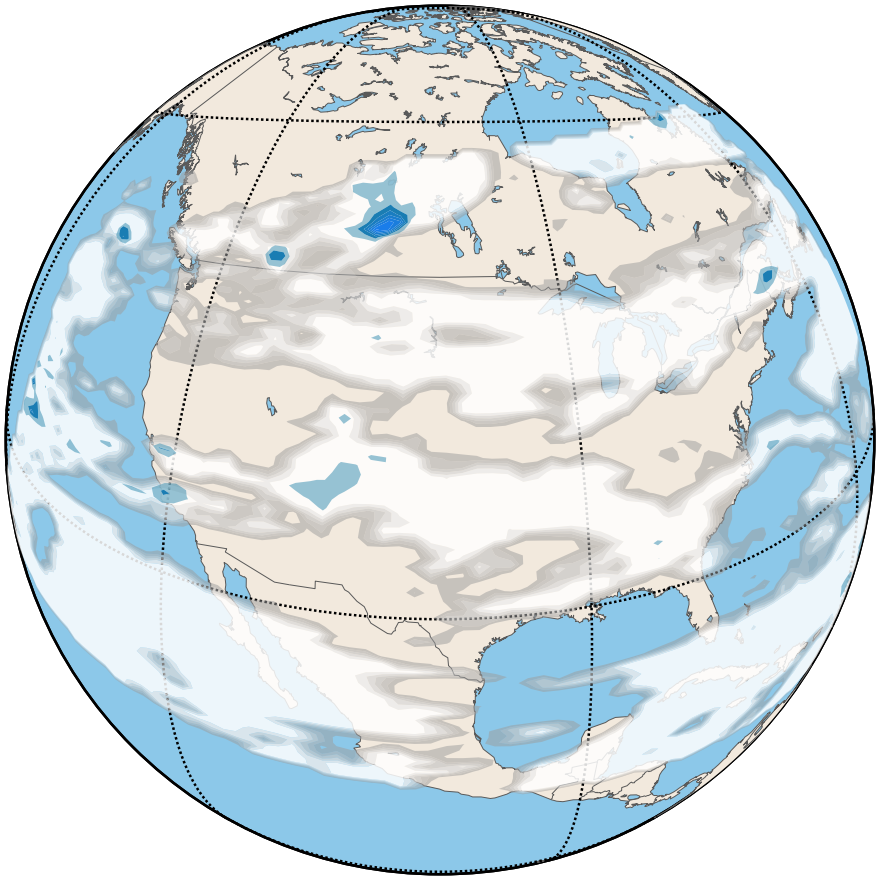
NAM Dataset Applications

- Predicting future weather events or patterns
  - Machine learning
  - Statistics
- Summarizing Information
  - Visualizations
  - Reports
- Exploring relationships between features
  - How does the temperature influence humidity?
  - How does the location influence precipitation?

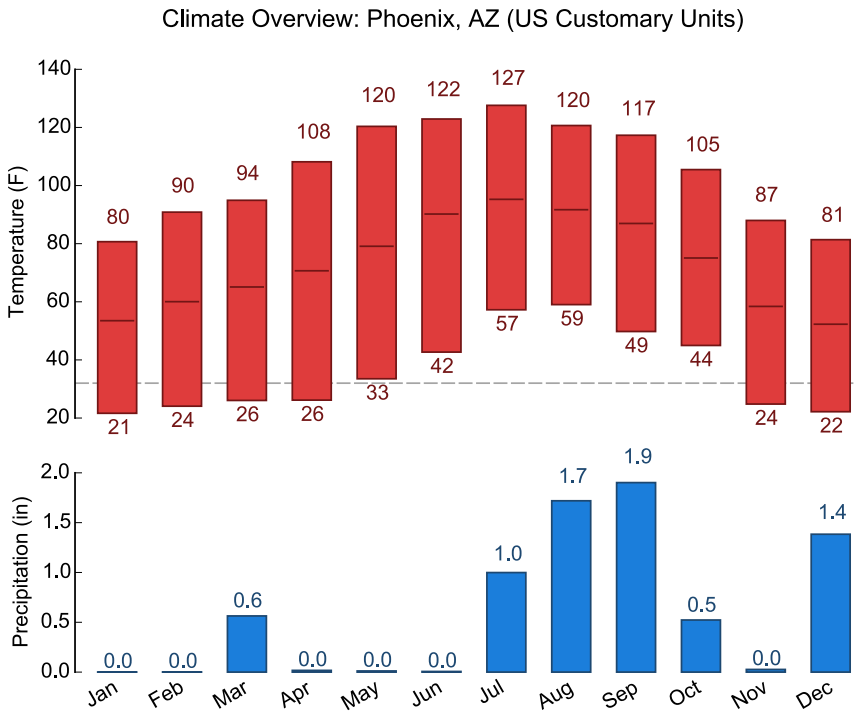
Predicting Rainfall: Wyoming

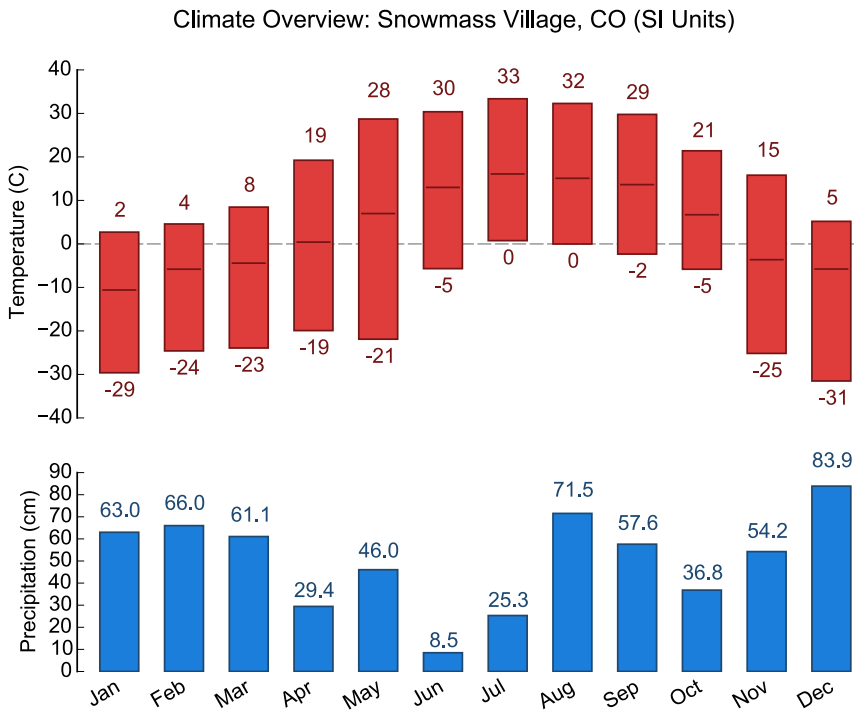


Contour Visualization



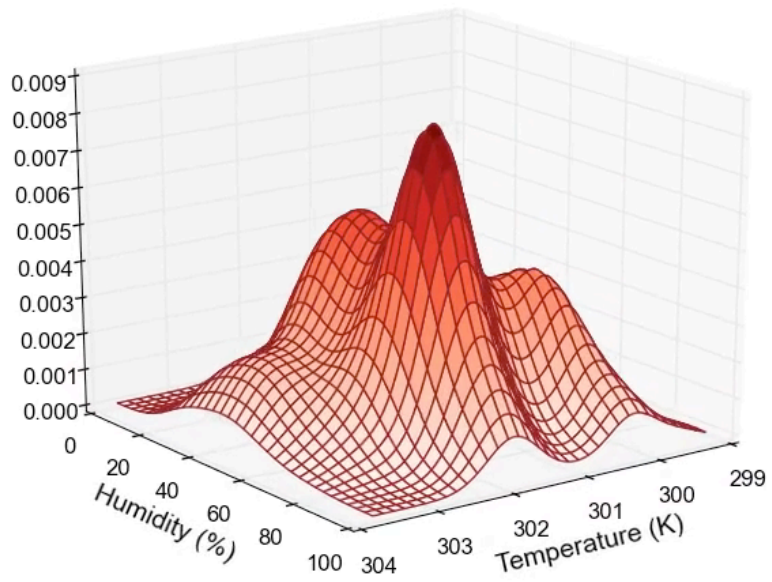
Climate Chart



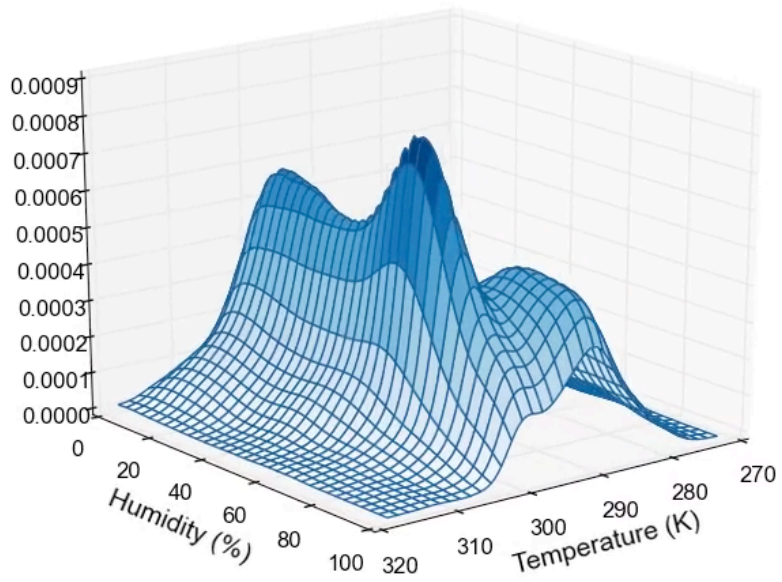


Relationships: Temp & Humidity

PDF(Temperature ∩ Humidity): Florida, USA



PDF(Temperature ∩ Humidity): Continental United States



Gathering Insights

- This dataset contains a wealth of information, but extracting insights from the data is challenging
- Multiple dimensions
- Storage requirements: where do we put all of it?
- Querying the data
  - (*knowledge discovery*)

Today's Agenda

- Q&A from previous class
- Introduction to spatiotemporal data
- **P2 dataset specifics**
- Geohash
- Spatial Indexes
- Spatial Queries

NAM Dataset Specifics

- The source data is stored in GRIB format
- Can be read and manipulated via the NetCDF library: https://www.unidata.ucar.edu/software/netcdf/
- We'll use a condensed version of the dataset for P2 instead

Condensed NAM

- Tab-delimited features (no fancy formats)
- We'll consider only 2D feature data
  - No elevation information
- Each .tdv file contains a month of readings **sampled** from the original source
  - nam_201501.tdv – January 2015

File Format

- Each line contains a single record
- Each record begins with:
  1. Time of reading (UNIX timestamp)
  2. Geohash location (more on this later!)
- Then, the remaining features (~56)
  - null if no reading existed

Some Interesting Features

- visibility
- pressure
- vegetation
- lightning
- temperature
- wind speed
- preciptation
- snow cover, snow depth

Today's Agenda

- Q&A from previous class
- Introduction to spatiotemporal data
- P2 dataset specifics
- **Geohash**
- Spatial Indexes
- Spatial Queries

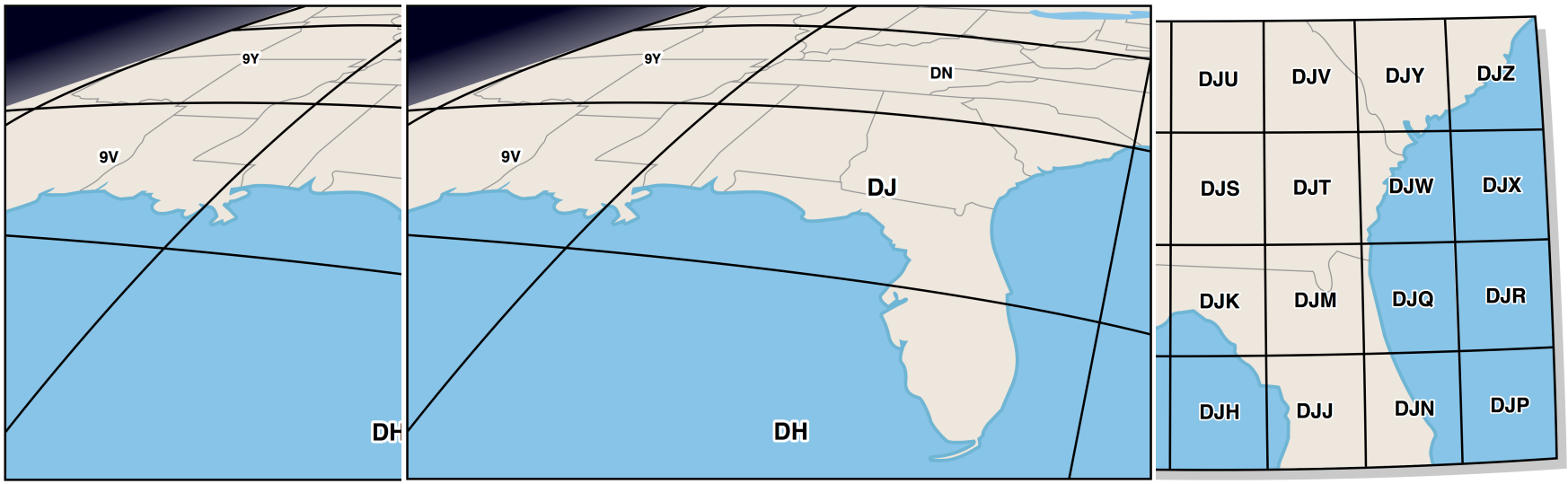
Spatial Queries

- Querying spatial data is a whole subject in itself
- If I gave you lat-lon pairs in the dataset, you could use those to perform simple spatial queries
  - If lat is >= something && lat <= something else:

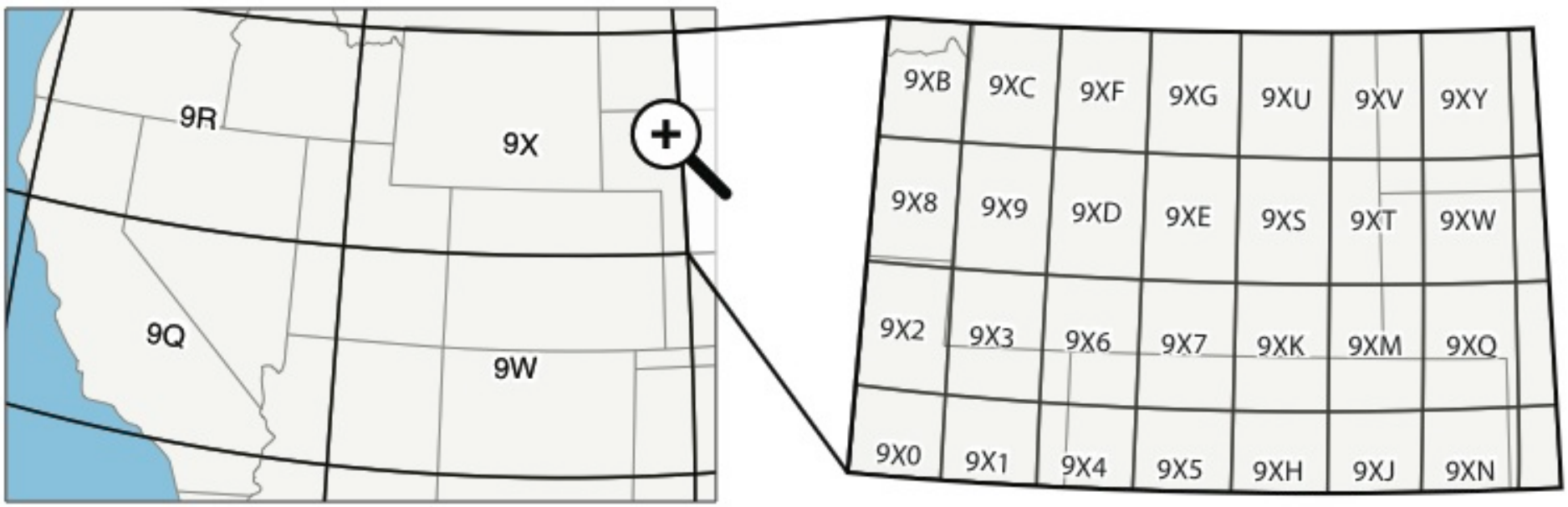
    blah blah blah();

    etc();
- A **better** option is to use the Geohash algorithm
  - Maps the earth to base32 strings
  - Defines a spatial **hierarchy** we can exploit

The Geohash Algorithm (1/2)



The Geohash Algorithm (2/2)



Geohash Details

- We use the ***Geohash*** algorithm to represent the spatial location associated with our sensor readings
  - Maps 2D spatial locations to 1D strings
  - Precision is determined by string length
- 9Q8YVT1F ➔ Kudlick Classroom
  - Similar string prefixes refer to similar locations
- Want to support range queries? Just match more or less of the string prefix

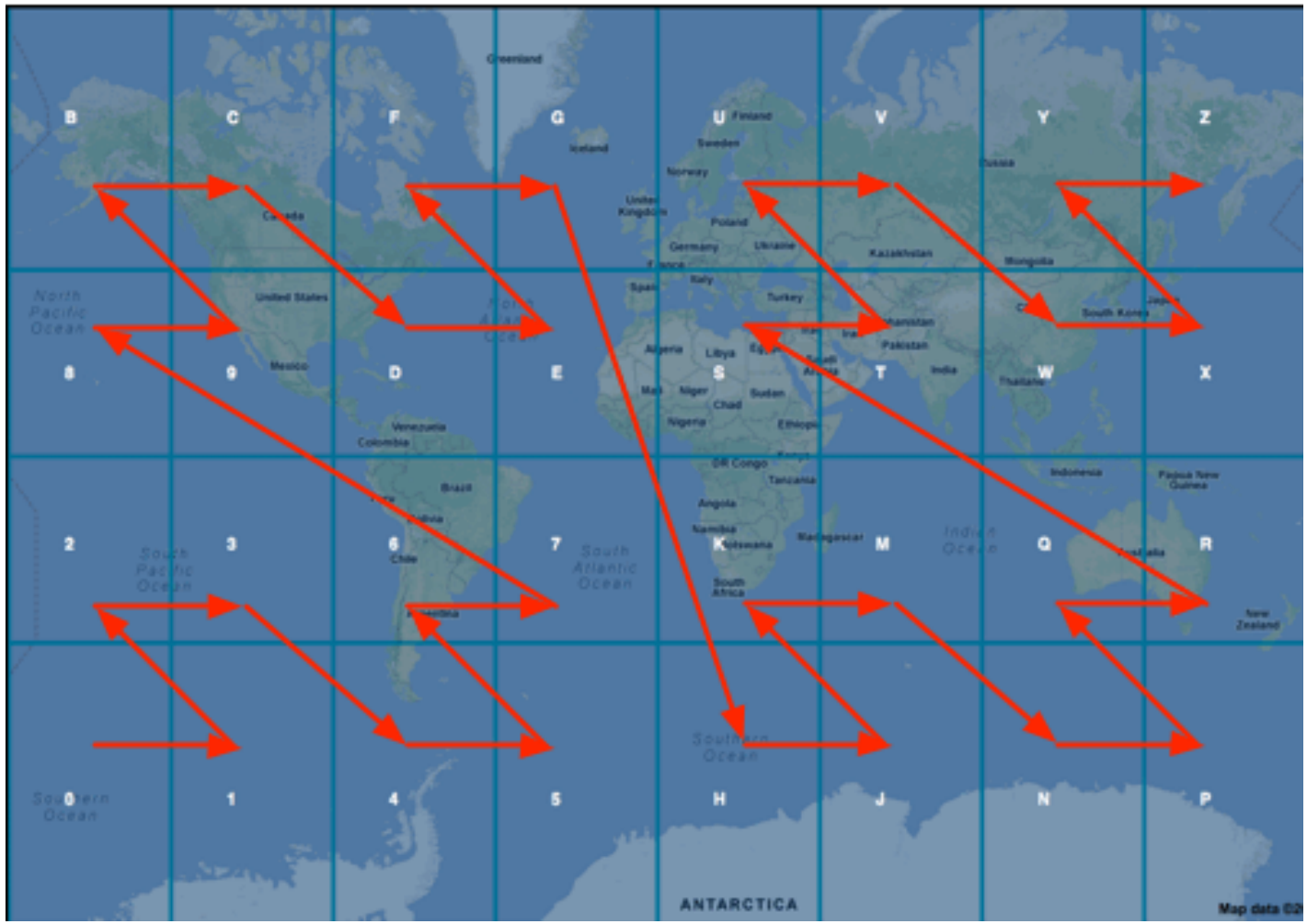
Geohash Resolutions

- Spatiotemporal data is not always evenly distributed
  - Compare the density of New York City and Glenwood Springs, Colorado
- Hash: 9XJQBF
  - 9XJQ = 20x30 km
  - 9X = 600x1000 km
- However, for our dataset, the data **IS** evenly distributed!
  - Fixed grid

Geohash Implementation

- Divides the bounding boxes in half with each binary bit added to the string
  - 1 bit = left or right half of the earth
  - 2 bits = top or bottom half of the left/right half
  - And so on…
- Uses 32 alphanumeric characters (Base 32)
  - 32 characters = 5 bits per character (5 divisions)
  - Omits some letters to avoid forming words

Z-Order Curve



Source: http://www.bigdatamodeling.org/2013/01/intuitive-geohash.html

Geohash Fun Facts

- Originally designed to allow users to share short URLs that represent locations
- Similar implementations have been used to identify locations for businesses, government
  - Ireland's proof-of-concept ***openpostcode*** can uniquely identify all locations within the UK
- Play with it! http://geohash.gofreerange.com

For Project 2

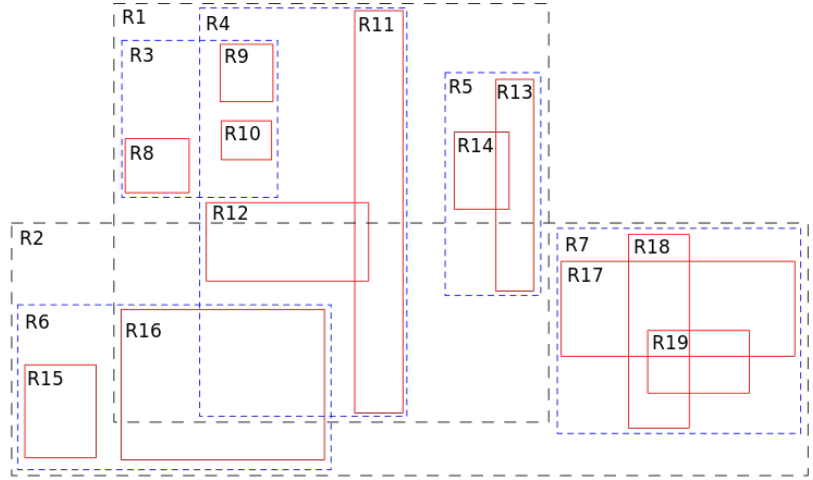
- I'll give you a class that can convert Geohashes to lat-longs, and vice versa
- It's written by me, so um, watch out!
  - (No seriously, it does work!)
- Feel free to use your own library or favorite spatial data structures library for this

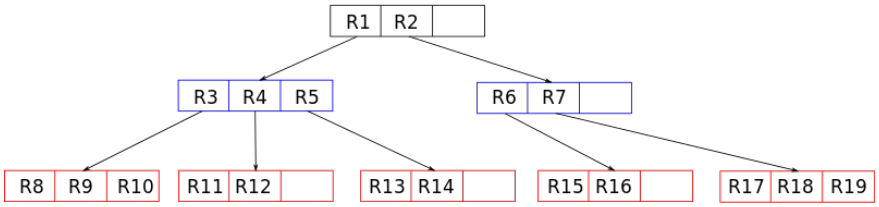
Today's Agenda

- Q&A from previous class
- Introduction to spatiotemporal data
- P2 dataset specifics
- Geohash
- **Spatial Indexes**
- Spatial Queries

Spatial Indexing: R-Trees

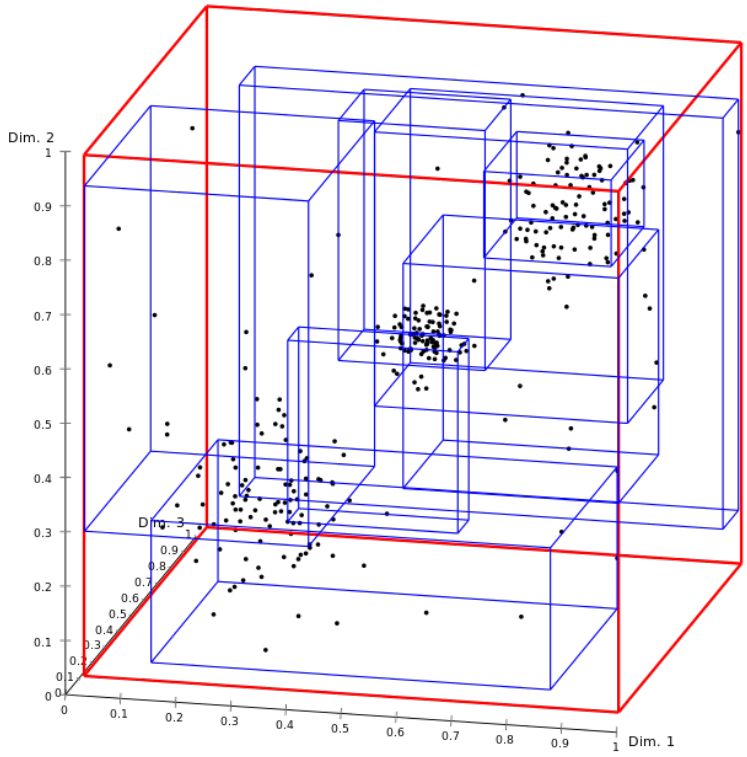
- **R-Trees** are a widely-used spatial index
- Share many similarities with B-Trees, but support spatial features:
  - Multiple dimensions
  - Intersection, containment queries
  - Nearest neighbor search





R-Tree Drawbacks

- R-Trees can be overwhelmed by extremely large datasets
- Query performance decreases as the number of leaves in the tree expands
  - Too much precision



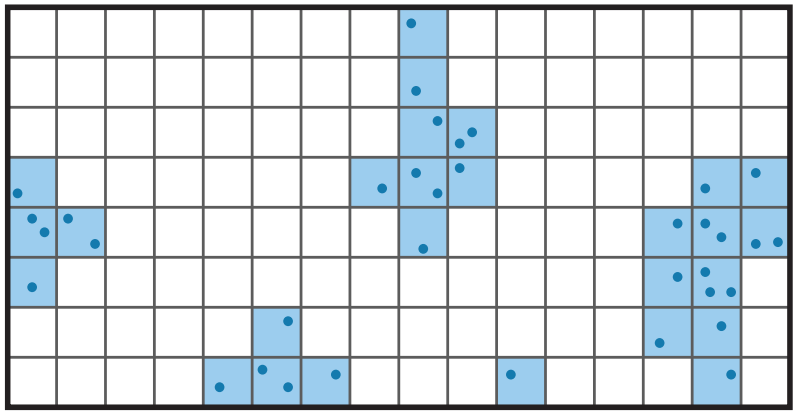
Alternative: Geoavailability Grid

- ***Bitmap index*** for spatial data
- Provides a coarse-grained representation of the spatial locations of information in the system
  - Limits the maximum memory consumption
  - May produce false positives
- Eliminates geographic regions from queries that do not contain relevant data

Bitmap Indexing

- Also known as: ***bit arrays*** or ***bitsets***
- Used in several areas:
  - Relational database management systems
  - Decision support systems
  - Data warehousing
- Just a stream of bits!
  - 01101000 01100101
  - 01101100 01101100
  - 01101111

Geoavailability Grid

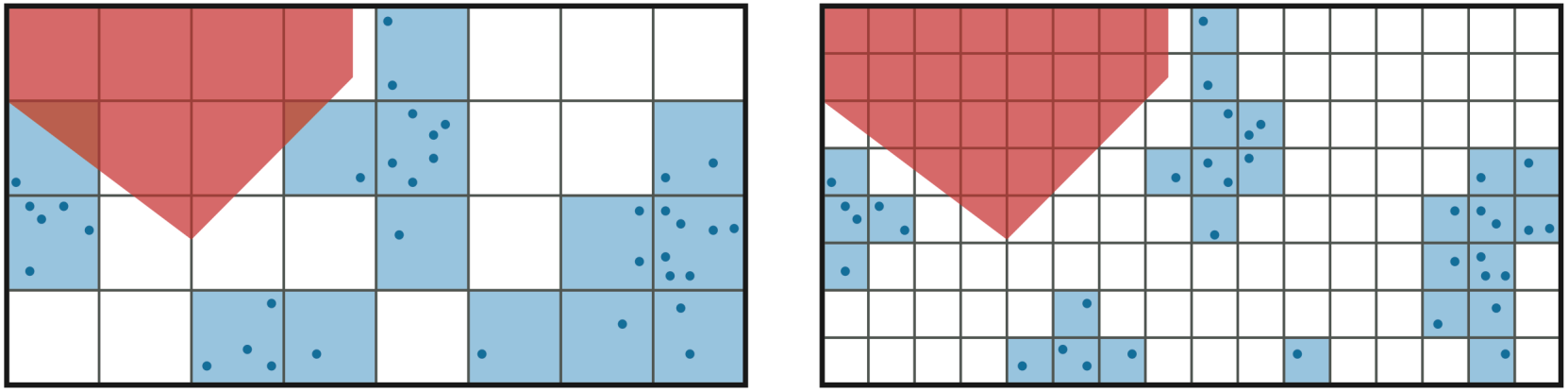


0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1
1	1	0	0	0	0	0	0	1	0	0	0	0	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	1	1	1	0	0	0	1	0	0	0	1	0

Bitmap Storage Requirements

- The resolution of the geoavailability grid determines the number of bits that must be stored
- Each node in the system maintains a geoavailability grid for each feature type it contains
- Higher precision maps require more bits, but provide greater reductions in search space

Resolution vs. Search Space Reduction



Bitmap Compression

- Bitmaps are compact, but massive datasets still require large amounts of bits
- The smaller the geoavailability grids, the more data can be held in memory
- Most bitmap implementations use run-length encoding (RLE) to reduce their size

Run-Length Encoding

- RLE is one of the simplest forms of compression
- Consider a binary string: 00000011111110001
- Run-length encoded: 60713011
- Geoavailability grids use Enhanced Word-Aligned Hybrid compression (EWAH)
  - More resilient to data with low sparsity
  - Better compression ratio than standard RLE
  - Increases the speed of bitwise operations

Today's Agenda

- Q&A from previous class
- Introduction to spatiotemporal data
- P2 dataset specifics
- Geohash
- Spatial Indexes
- **Spatial Queries**

Brute-force Approach

- Submit queries to all nodes, scan over records
  - **This is what we'll do in P2**
- Issues:
  - Wasted processing on nodes with no matches
  - Not scalable
  - High latencies
- A better approach: ***decompose*** the query and distribute it only to relevant nodes

Query Evaluation Process

1. User submits a polygon and feature constraints
   - "Give me humidity values for San Francisco in July"

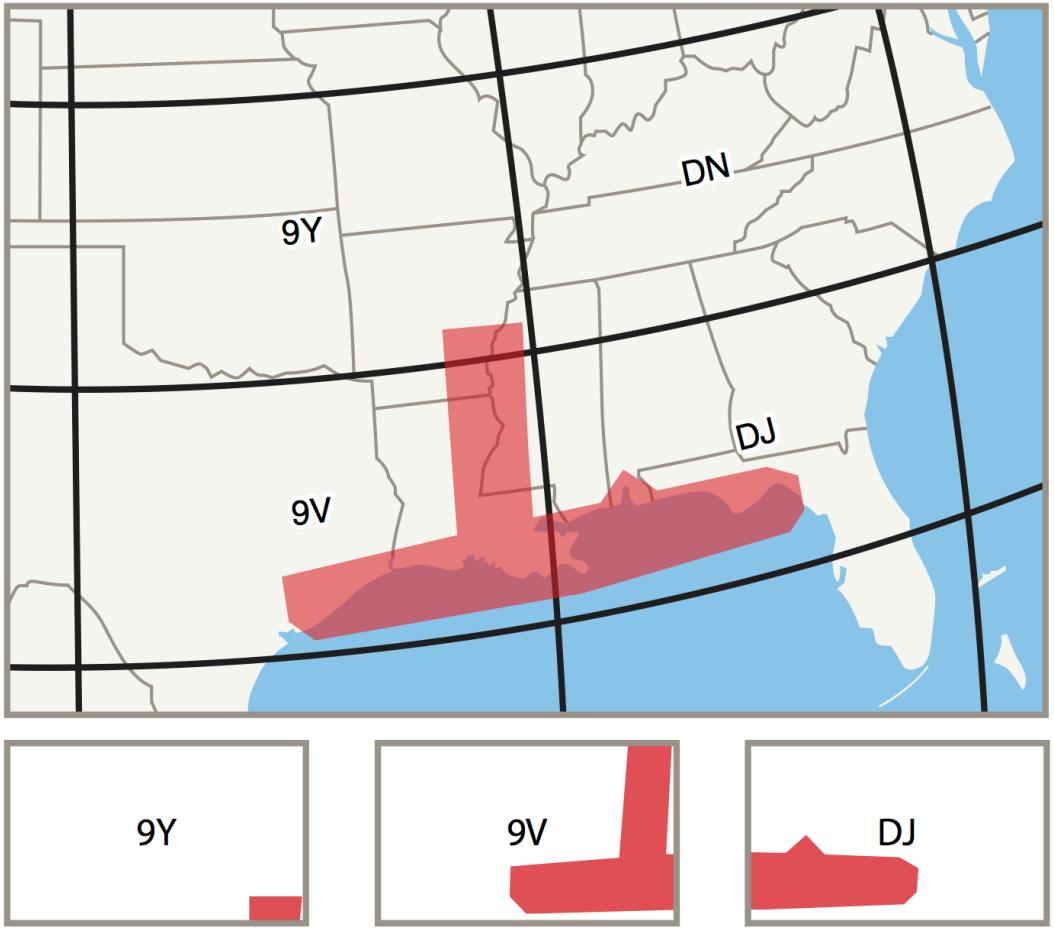
2. The query is *decomposed* into multiple *subqueries* based on Geohash boundaries

3. Subqueries are distributed to individual storage nodes for processing

Representing Polygon Queries

- Users are provided a scrollable, zoomable map with standard polygon/rectangle drawing tools
- Each component of the input polygon is represented by <latitude, longitude> pairs
- Coordinate pairs are stored by creation order and serialized to a binary format

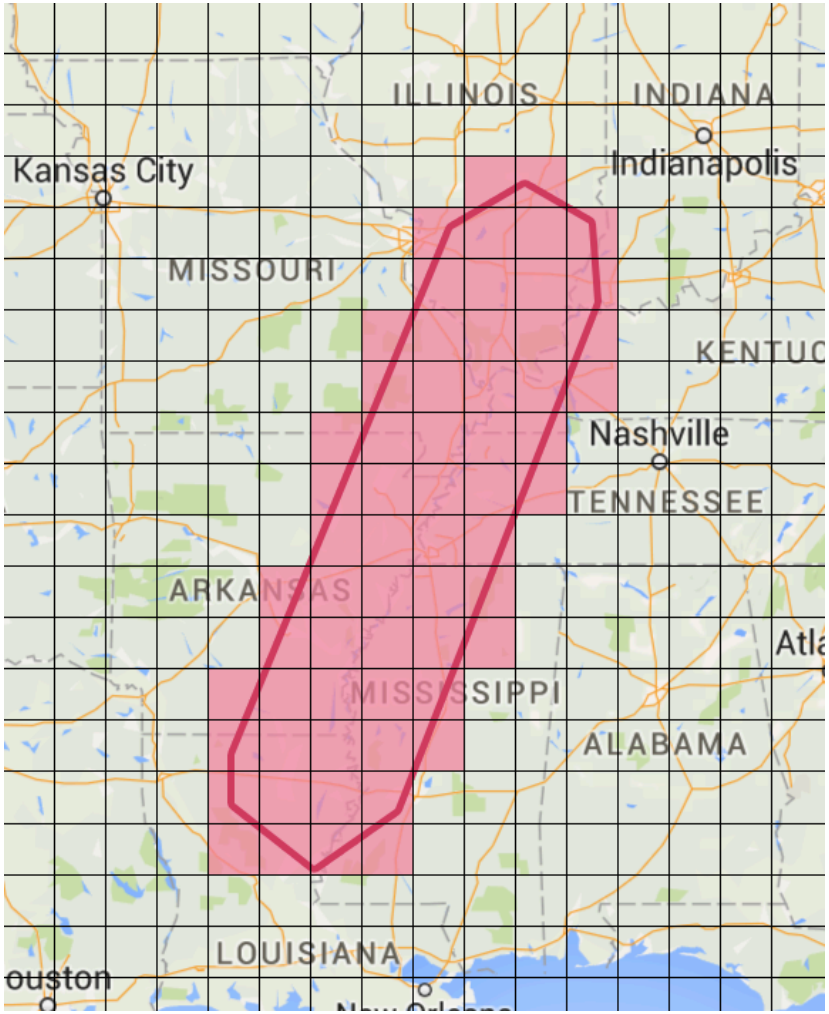
Spatial Decomposition



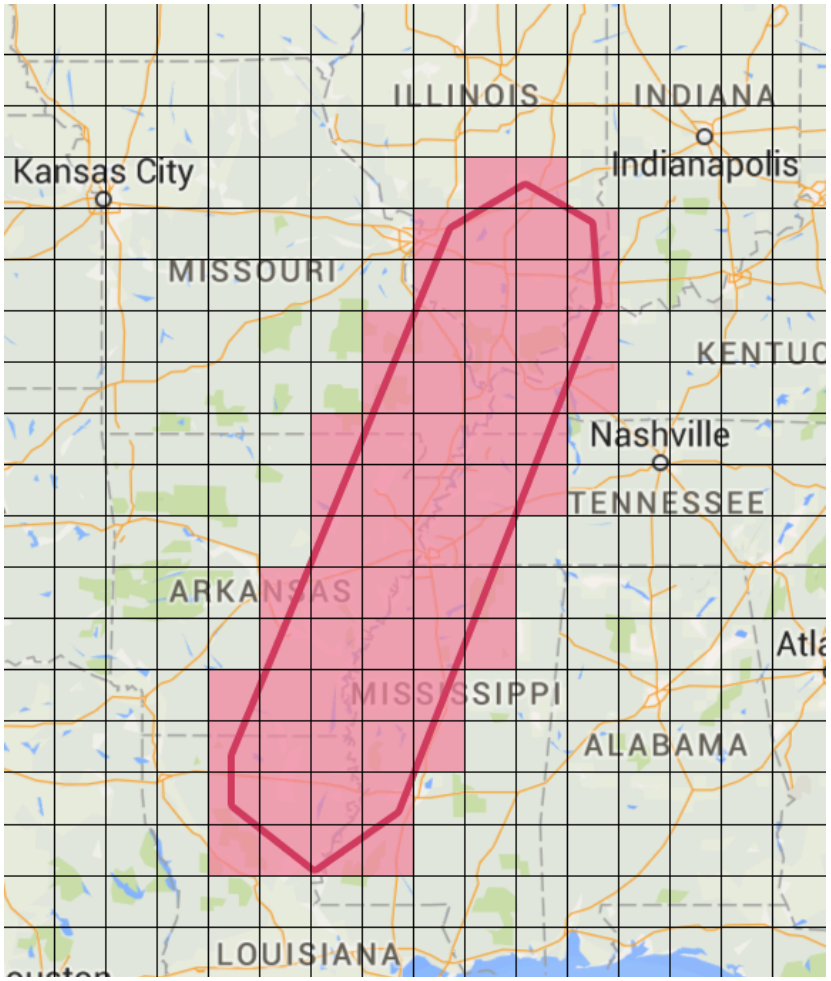
Geoavailability Evaluation

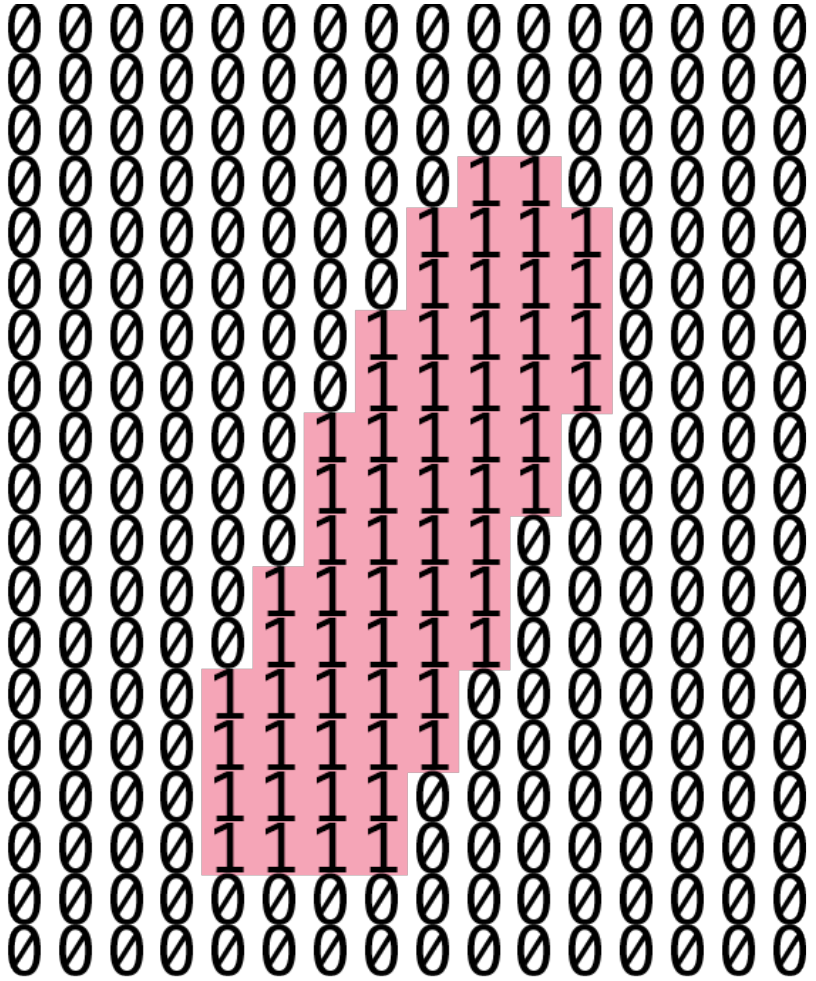
- Query polygons are transformed into query bitmaps
  - Uses standard graphics routines
  - Can be GPU accelerated
- A bitwise AND is performed between the query bitmap and each geoavailability grid
  - If the result is an empty set, the storage node does not contain relevant data

Polygon Transformation [1/2]



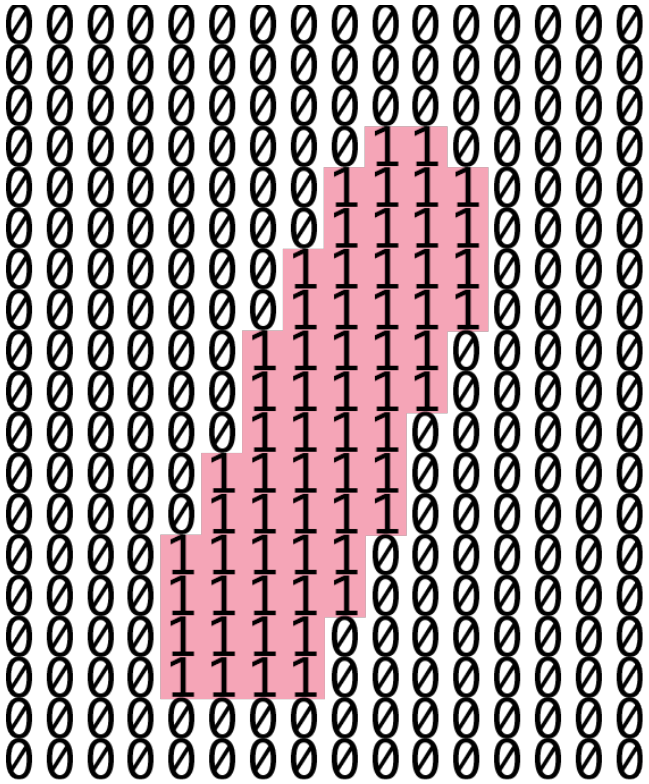
Polygon Transformation [2/2]



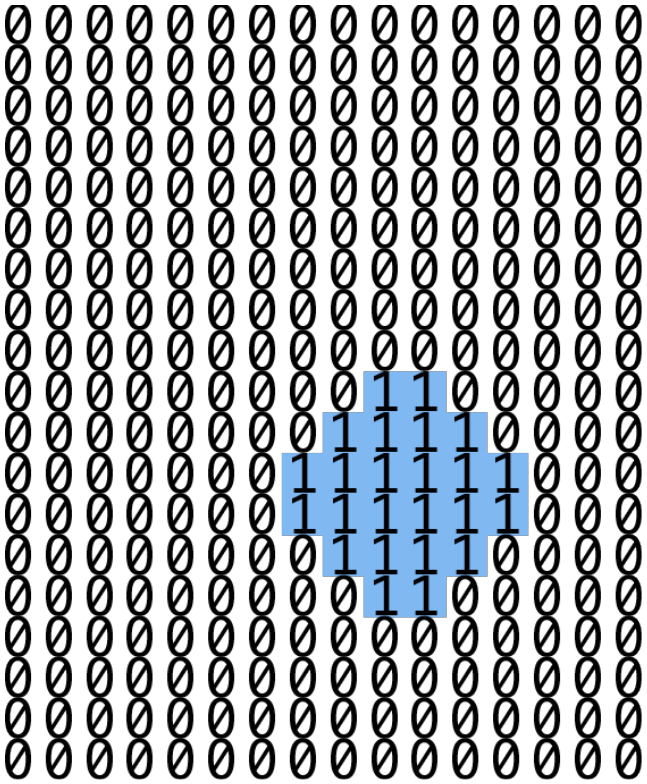


Intersection Queries

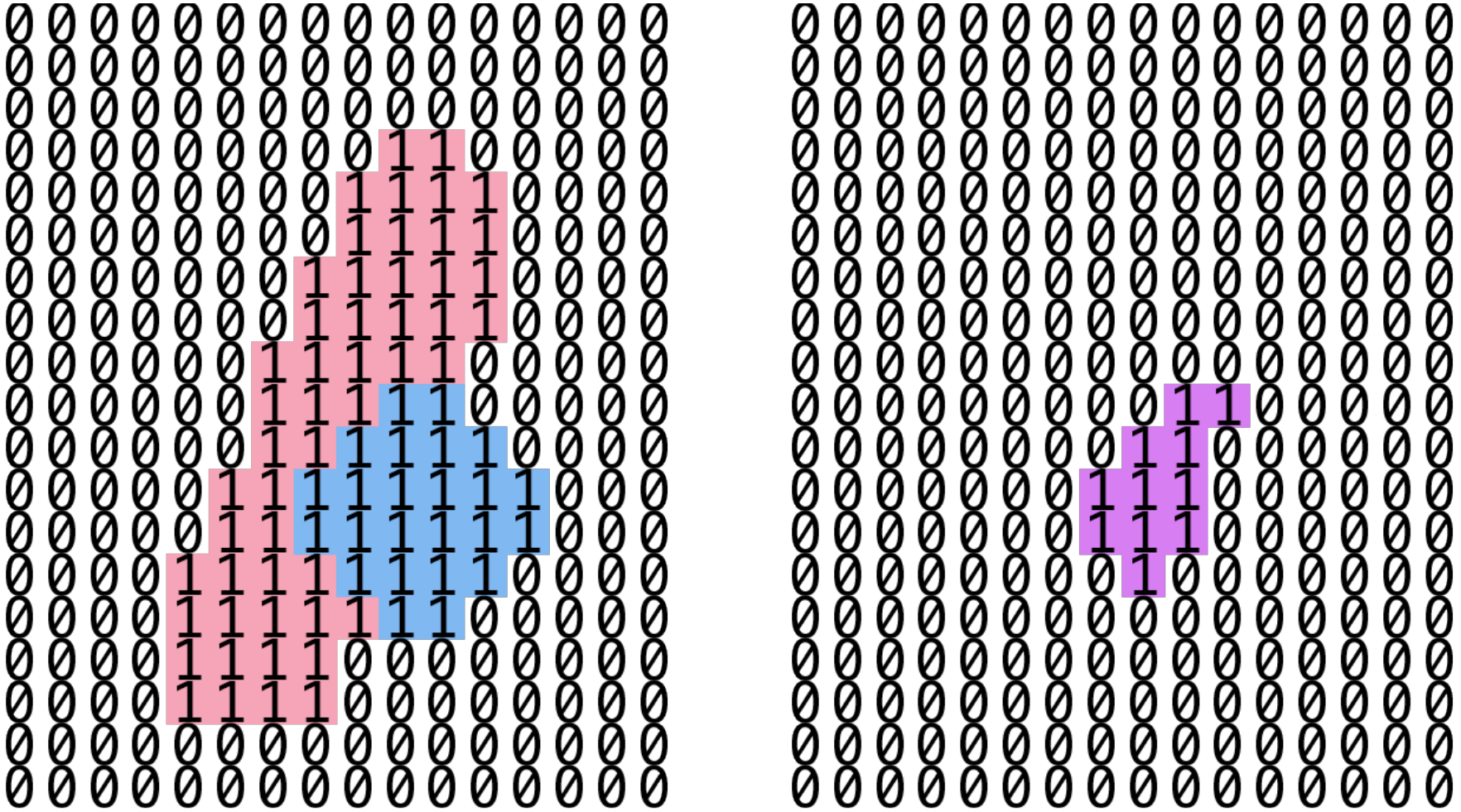
Query



Data



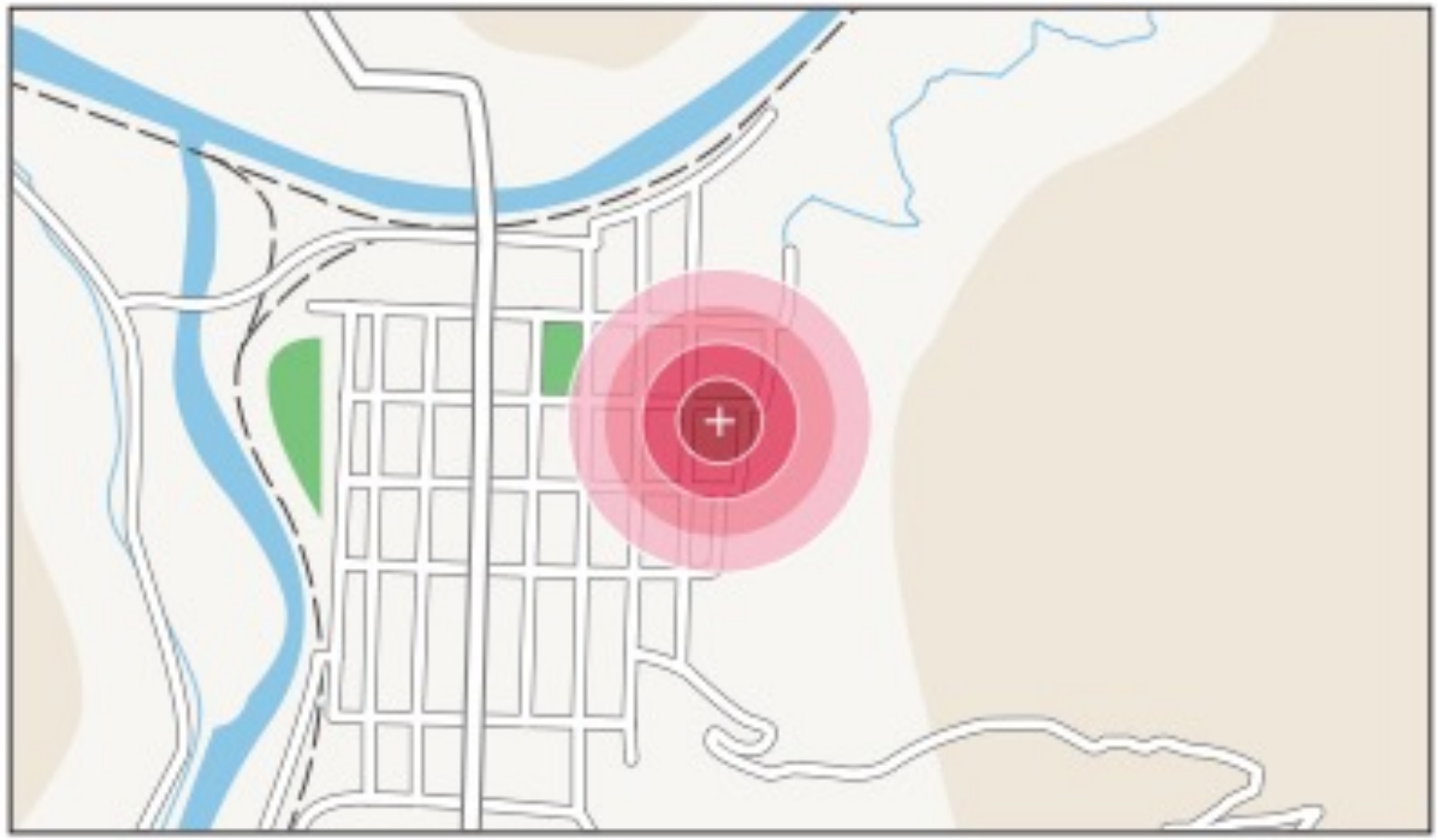
Bitwise AND



Proximity Queries

- Retrieves relevant records near a starting coordinate pair
  - "Where's the nearest coffee shop?"
- Successively larger *annuli* are generated around the starting location and evaluated against the geoavailability grid
- The search stops when a match is found, or a specified maximum area has been covered

Proximity Query Example



Constrained Proximity Queries

- In some cases, users may wish to constrain a proximity query to a particular region
  - "Find the nearest coffee shop in Santa Clara county"
- Requires constraining geometry
  - US **T**opologically **I**ntegrated **G**eographic **E**ncoding and **R**eferencing (TIGER) spatial dataset
  - Includes counties, states, administrative boundaries

Constrained Proximity Example

Performance Evaluation

- Geoavailability Grids are based on space efficient, fast data structures: **bitmaps**
- Query evaluation speeds can be boosted by using GPU acceleration
- The real question: are they faster than R-Trees?

Geoavailability Grid Lookup

Bitmap Resolution	Lookup Time (ms)	Standard Deviation (ms)
2^{25}	0.012	0.021
2^{30}	0.163	0.203
2^{35}	0.723	0.289

Query Performance Comparison