# Geospatial Indexing & Search at Scale with Lucene

Nick Knize, PhD
Lucene PMC Member + Committer
@nknize

# Lets take a trip...

## ...through Geo advancements

1. Lucene Geo in Elasticsearch, an Introduction
2. Lucene Geo Search, Then & Now: Prefix Trees vs. BKD Trees
3. Improving Shape Search: BKD + Tessellation

# Mappings

## Geo Field Types

# geo_point mapping

## define

```
PUT crime/incidents/_mapping
{
    “properties” : {
     “location” : {
       “type” : “geo_point”,
       “ignore_malformed” : true,
     }
  }
}
```

# geo_point mapping

## insert

```
POST crime/incidents
{
  "location" : { "lat" : 41.12, "lon" : -71.34 }
}

POST crime/incidents
{
  "location" : "41.12, -71.34"
}

POST crime/incidents
{
  "location" : [[-71.34, 41.12], [-71.32, 41.21]]
}
```

# geo_shape mapping

## define

```
PUT police/precincts/_mapping
{
  "properties" : {
    "coverage" : {
      "type" : "geo_shape",
      "ignore_malformed" : false,
      "tree" : "quadtree",
      "precision" : "5m",
      "distance_error_pct" : 0.025,
      "orientation" : "ccw",
      "points_only" : false
    }
  }
}
```

# geo_shape mapping

## insert

```
POST police/precincts/
{
  "coverage" : {
    "type" : "polygon",
    "coordinates" : [[
      [-73.9762134, 40.7538588],
      [-73.9742356, 40.7526327],
      [-73.9656733, 40.7516774],
      [-73.9763236, 40.7521246],
      [-73.9723788, 40.7516733],
      [-73.9732423, 40.7523556],
      [-73.9762134, 40.7538588]
    ]]
  }
}
```

# geo_shape mapping

## insert

- Shapes are parsed using OGC and ISO standards definitions
  - OGC Simple Feature Access
  - ISO Geographic information — Spatial Schema (19107:2003)

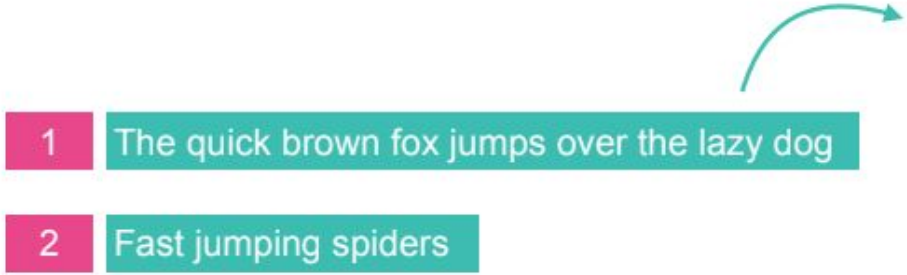
- Supports the following geo_shape types
  - Point, MultiPoint
  - LineString, MultiLineString
  - Polygon (with holes), MultiPolygon (with holes)
  - Envelope (bbox)

# Indexing

Under the hood

# A long, long time ago, in a galaxy far far away…

## …the Inverted [text] Index




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| Fast | 2 |
| The | 1 |
| brown | 1 |
| dog | 1 |
| fox | 1 |
| jumping | 2 |
| jumps | 1 |
| lazy | 1 |
| over | 1 |
| quick | 1 |
| spiders | 2 |
| the | 1 |

# And the open source community went bananas!

we 🫶🏽 Lucene text search!

# But what about 1D numbers...?

## Prefix Trees; precision

*1* myINT: 3881553173

1110 0111 0101 1011 1100 1101 0001 0101

*2* myINT: 2405166357

1000 1111 0101 1011 1110 1101 0001 0101

*3* myINT: 3205335297

1011 1111 0000 1101 1000 1001 0000 0001

*4* myINT: 2835679237

1010 1001 0000 0101 0000 1000 0000 0101

*5* myINT: 4177856517

1111 1001 0000 1101 0000 1000 0000 0101

| *terms dictionary (terms)* | *postings list (doc ids)* |
|---|---|
| *1* | *1, 2, 3, 4, 5* |
| *10* | *2, 3, 4* |
| *11* | *1, 5* |
| *100* | *2* |
| *101* | *3, 4* |
| *111* | *1, 5* |
| *1000* | *2* |
| *1010* | *4* |
| *1011* | *3* |
| *1110* | *1* |
| *1111* | *5* |

# But what about numbers…?

## Prefix Trees; precision

*1* myINT: 3881553173

1110 0111 0101 1011 1100 1101 0001 0101

*2* myINT: 2405166357

1000 1111 0101 1011 1110 1101 0001 0101

*3* myINT: 3205335297

1011 1111 0000 1101 1000 1001 0000 0001

*4* myINT: 2835679237

1010 1001 0000 0101 0000 1000 0000 0101

*5* myINT: 4177856517

1111 1001 0000 1101 0000 1000 0000 0101

| *terms dictionary (terms)* | *postings list (doc ids)* |
|---|---|
| *1* | *1, 2, 3, 4, 5* |
| *10* | *2, 3, 4* |
| *11* | *1, 5* |
| *100* | *2* |
| *101* | *3, 4* |
| *111* | *1, 5* |
| *1000* | *2* |
| *1010* | *4* |
| *1011* | *3* |
| *1110* | *1* |
| *1111* | *5* |

# And the Lucene community went…bananas…again…

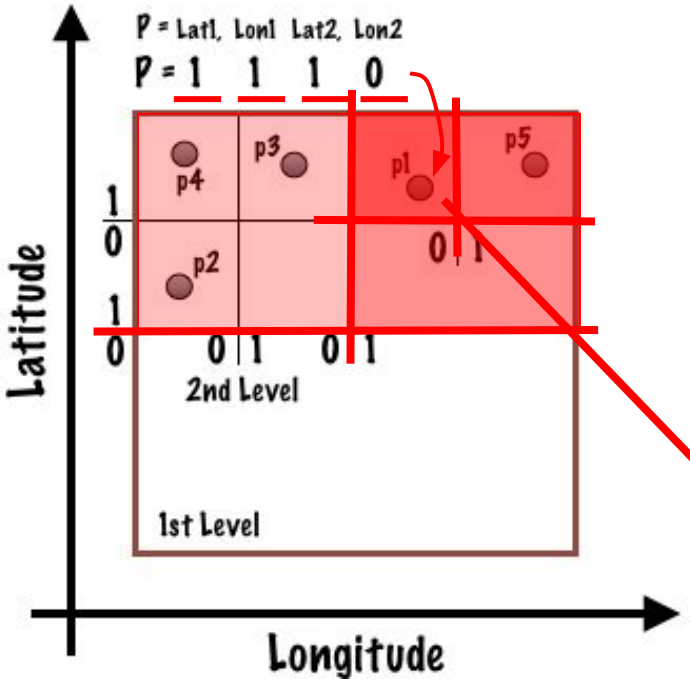
we 🫶 Lucene NUMERIC search!

# But, but, what about Geo?

…we’re cool too!

# How about Geo…terms?

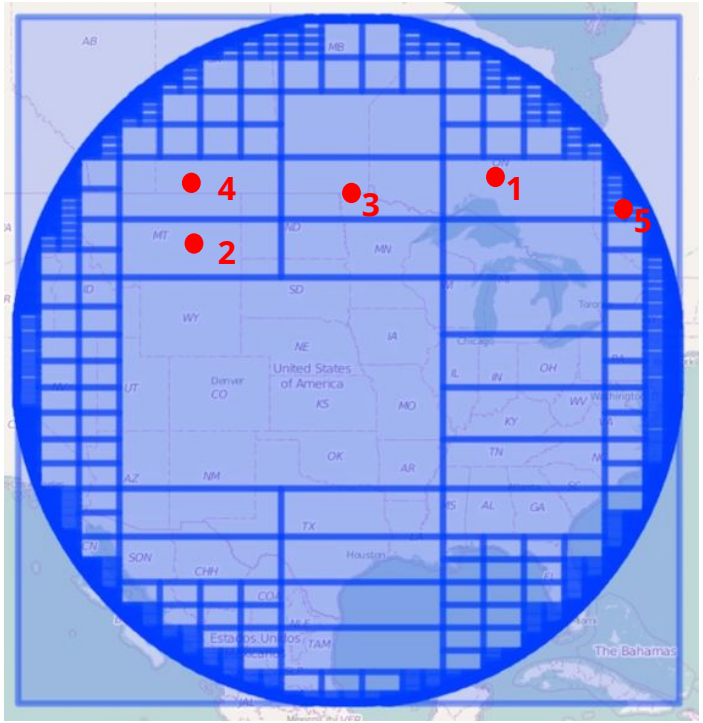
## Prefix Trees; round peg, square hole




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Searching for round pegs in square holes…

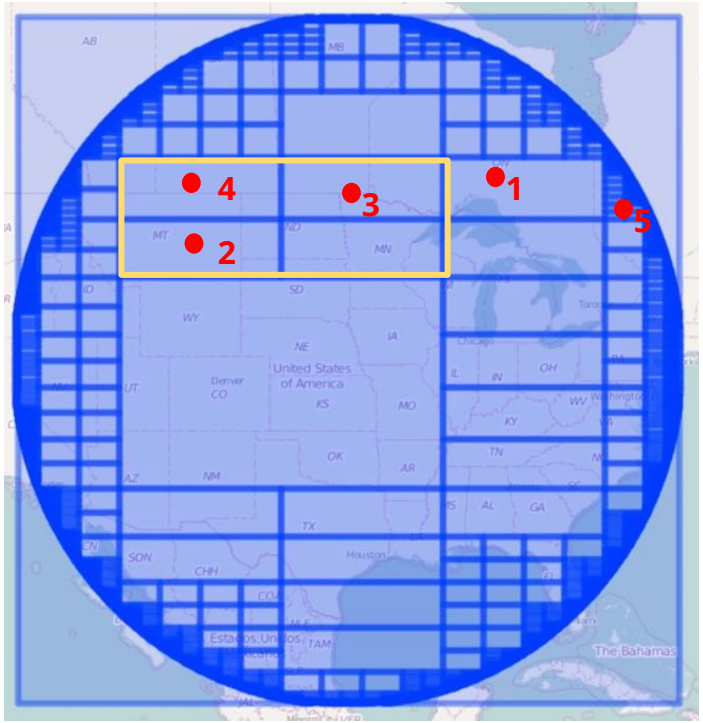
## Traversing the Prefix Tree…




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Searching for round pegs in square holes…

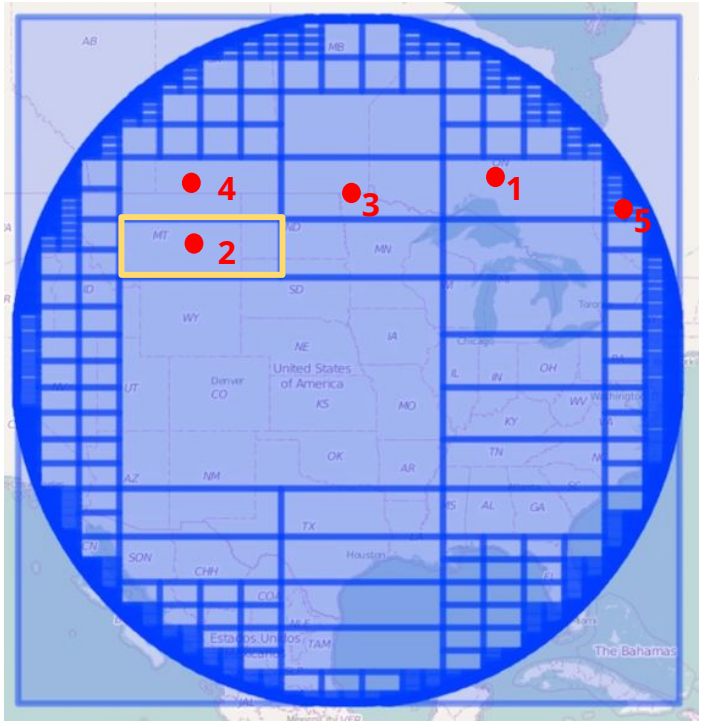
## Traversing the Prefix Tree…




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Searching for round pegs in square holes…

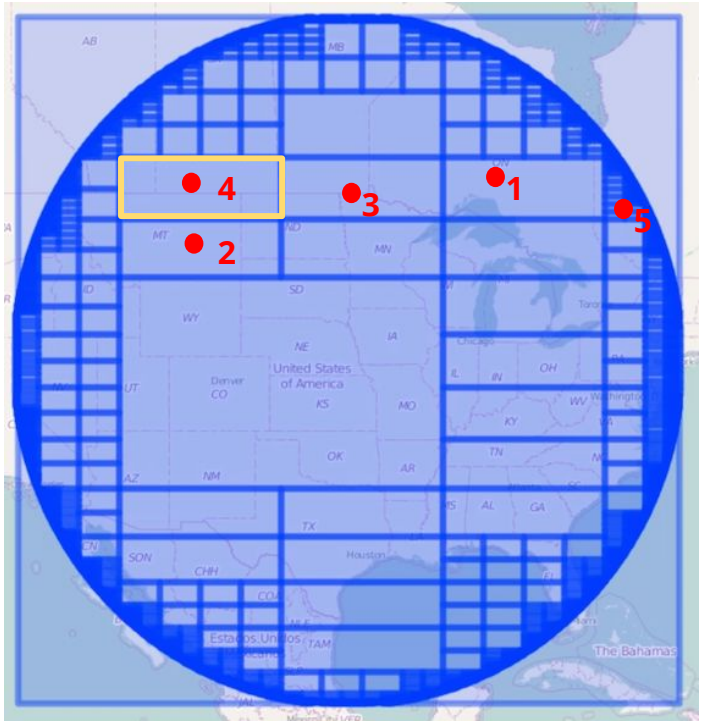
## Traversing the Prefix Tree…




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Searching for round pegs in square holes…

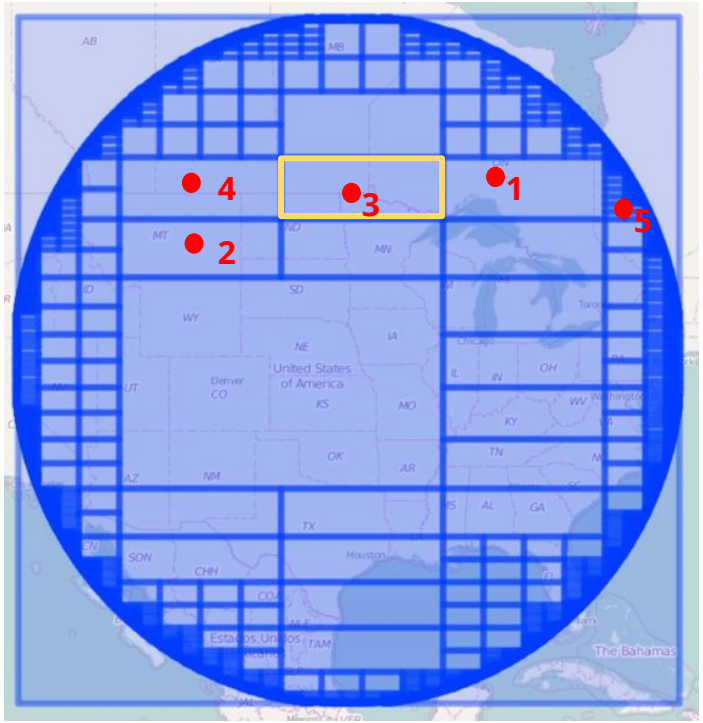
## Traversing the Prefix Tree…




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Searching for round pegs in square holes…

## Traversing the Prefix Tree…




| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Searching for round pegs in square holes…

## Traversing the Prefix Tree…of death?

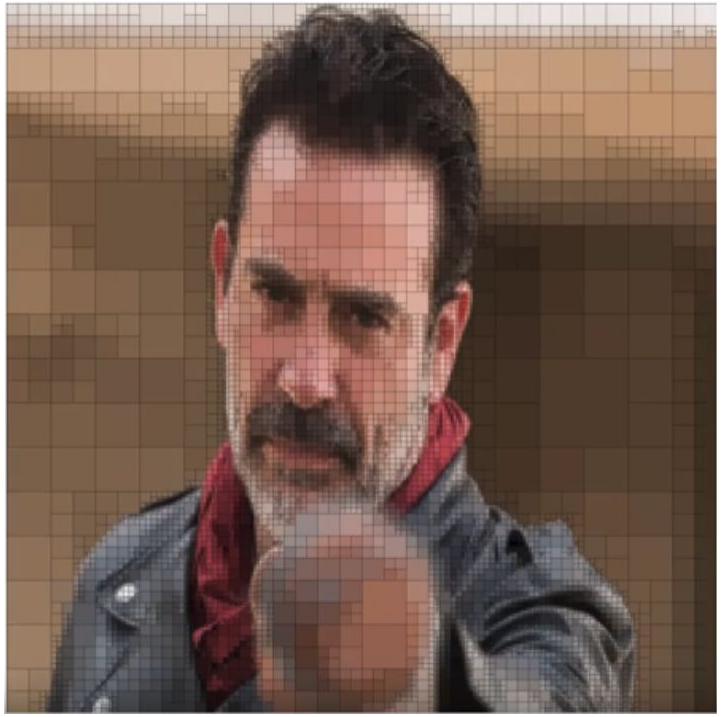
| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 2, 3, 4 |
| 11 | 1, 5 |
| 100 | 2 |
| 101 | 3, 4 |
| 111 | 1, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 1 |
| 1111 | 5 |

# Okay, cool! But what about GeoShapes?!

## Quad Trees!

# Okay, cool! But what about GeoShapes?!

## Use the Inverted [geo] Index...

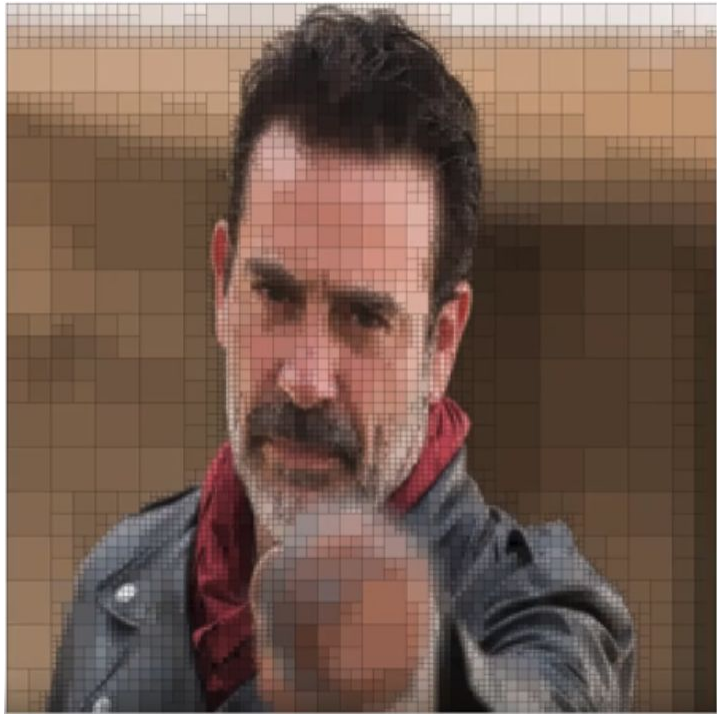


DocID: 6

| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5 |
| 10 | 1, 2, 4 |
| 11 | 3, 5 |
| 100 | 1 |
| 101 | 2, 4 |
| 111 | 3, 5 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 3 |
| 1111 | 5 |

# Okay, cool! But what about GeoShapes?!

## Use the Inverted [geo] Index...

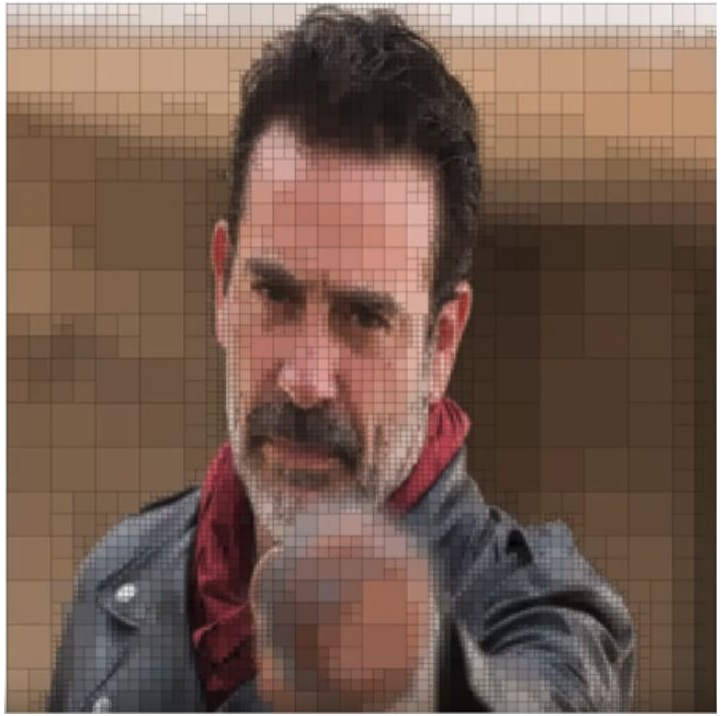


DocID: 6

| terms dictionary (terms) | postings list (doc ids) |
|---|---|
| 1 | 1, 2, 3, 4, 5, 6 |
| 10 | 1, 2, 4 |
| 11 | 3, 5, 6 |
| 100 | 1 |
| 101 | 2, 4 |
| 111 | 3, 5, 6 |
| 1000 | 2 |
| 1010 | 4 |
| 1011 | 3 |
| 1110 | 3, 6 |
| 1111 | 5, 6 |

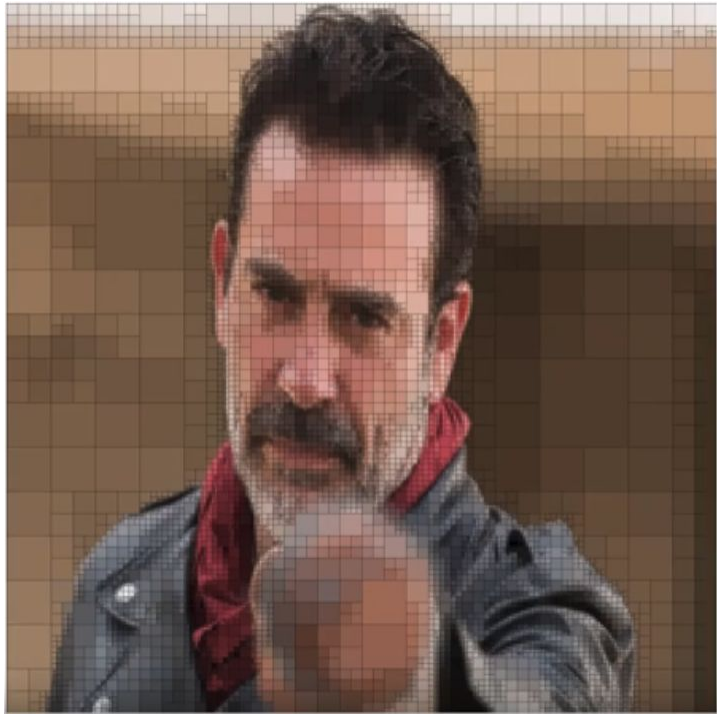
# Okay, cool! But what about GeoShapes?!

## Melt with the Inverted [geo] Index...



# Okay, cool! But what about GeoShapes?!

## Melt with the Inverted [geo] Index…

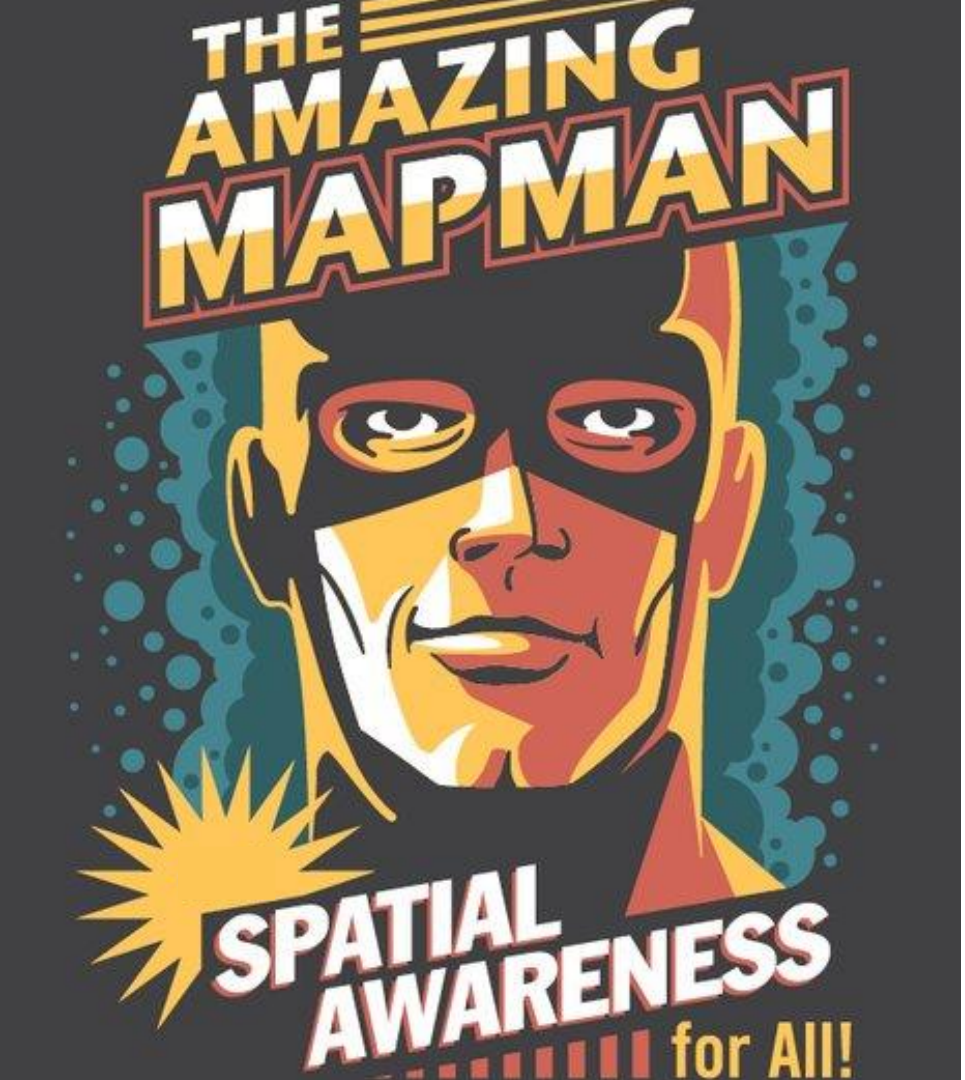


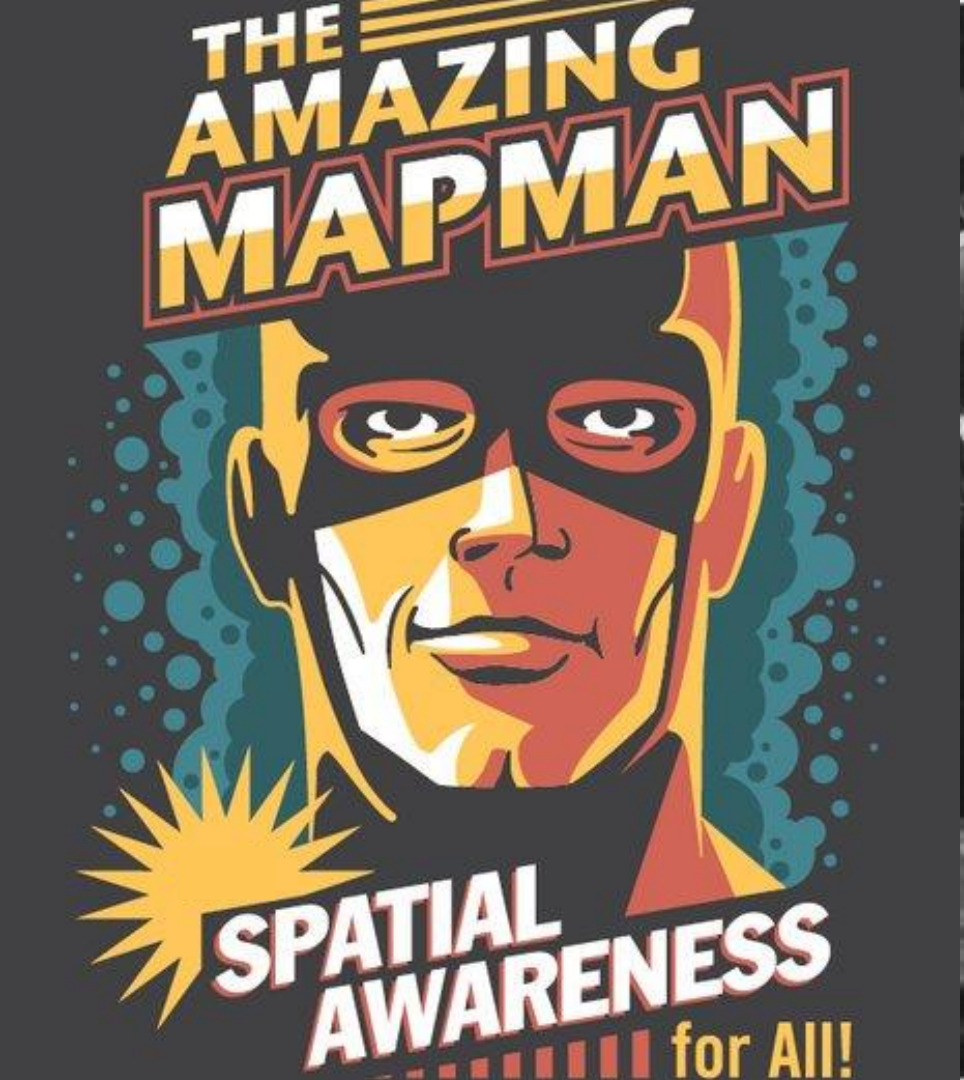
- Max tree_levels == 32 (2 bits / cell)
- distance_error_pct
  - “slop” factor to manage transient memory usage
  - % of the diagonal distance (degrees) of the shape
  - Default == 0 if precision set (2.0)
- points_only
  - optimization for points only shape index
  - short-circuits recursion

# geo search?

pssh… it's ~~simple!~~ WTF?!

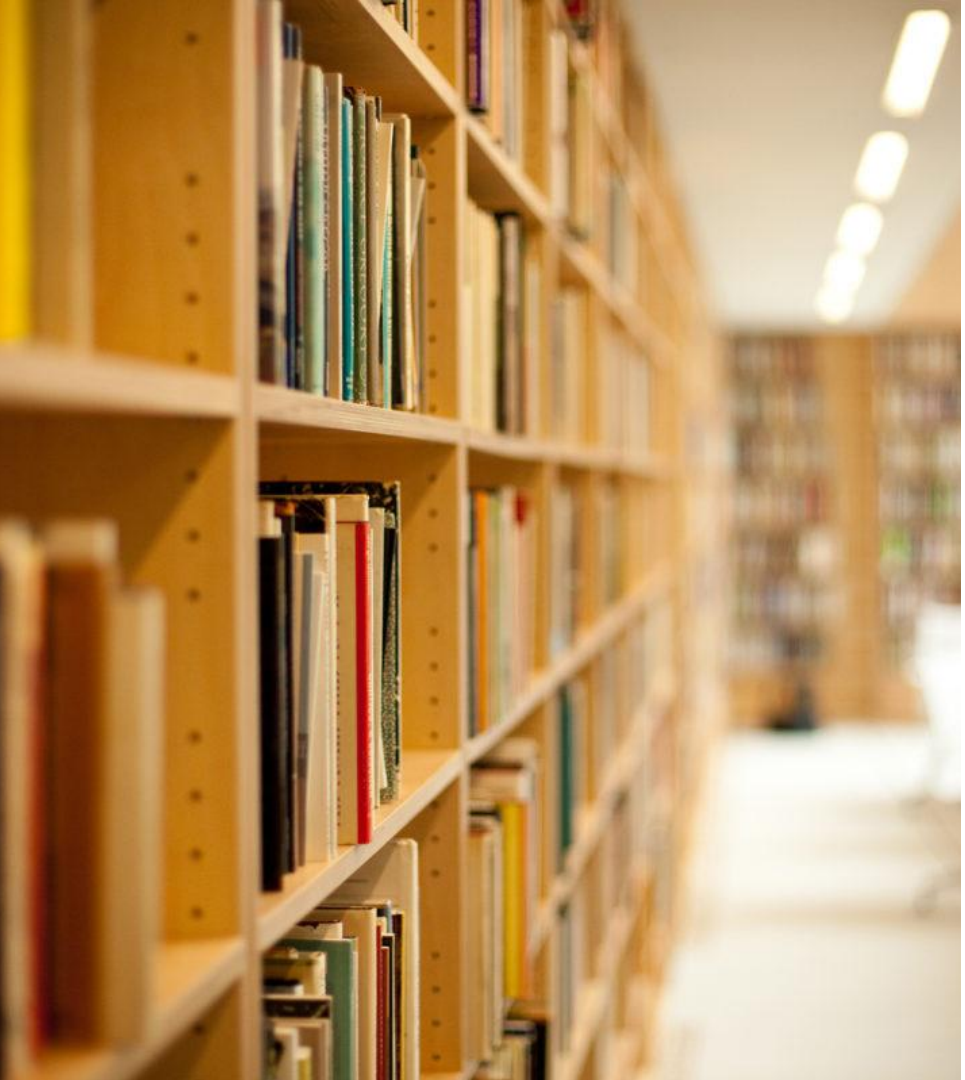

| | | |
|---|---|---|
| ¯_(ツ)_/¯ | `geohash` | Should the geo-point also be indexed as a geohash in the `.geohash` sub-field? Defaults to `false`, unless `geohash_prefix` is `true`. |
| ¯_(ツ)_/¯ | `geohash_precision` | The maximum length of the geohash to use for the `geohash` and `geohash_prefix` options. |
| ¯_(ツ)_/¯ | `geohash_prefix` | Should the geo-point also be indexed as a geohash plus all its prefixes? Defaults to `false`. |
| ¯_(ツ)_/¯ | `ignore_malformed` | If `true`, malformed geo-points are ignored. If `false` (default), malformed geo-points throw an exception and reject the whole document. |
| ¯_(ツ)_/¯ | `lat_lon` | Should the geo-point also be indexed as `.lat` and `.lon` sub-fields? Accepts `true` and `false` (default). |
| ¯_(ツ)_/¯ | `precision_step` | Controls the number of extra terms that are indexed for each lat/lon point. Defaults to `16`. Ignored if `lat_lon` is `false`. |



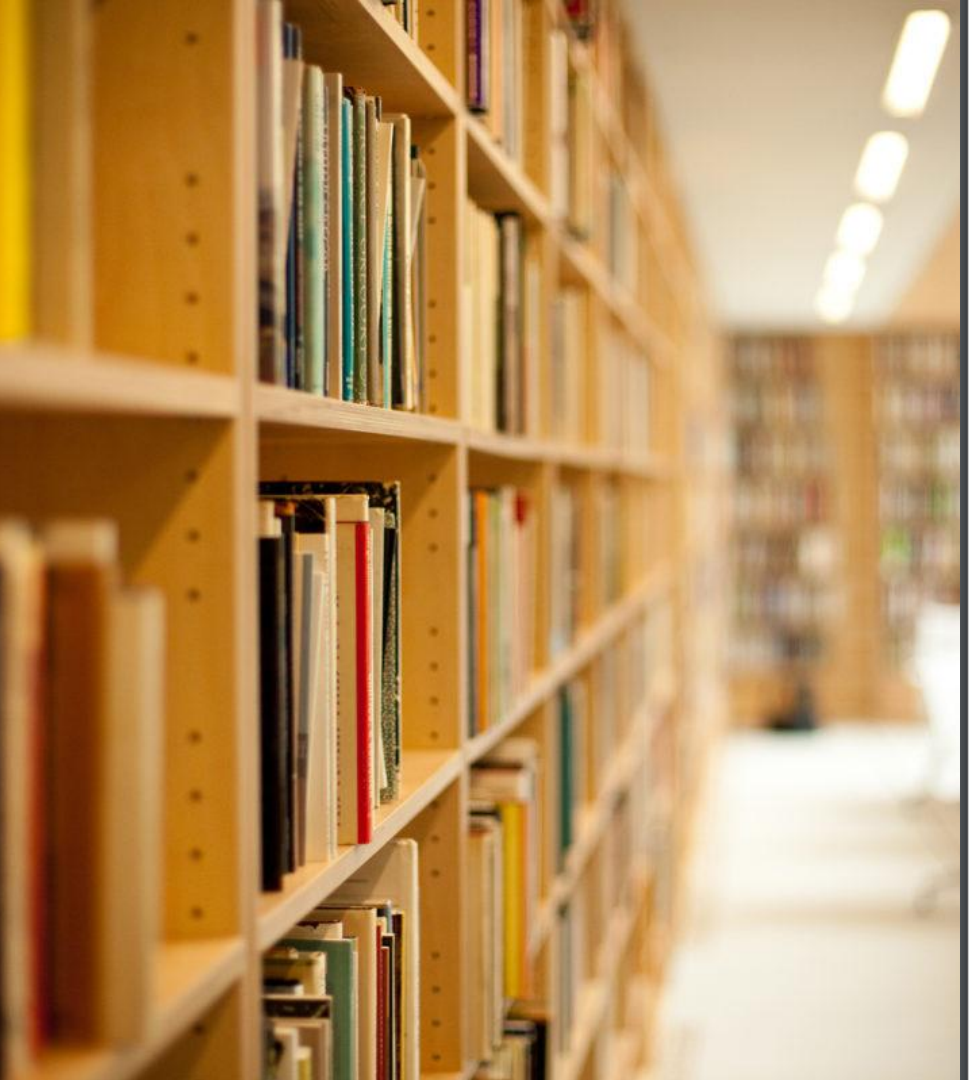

THE AMAZING MAPMAN
SPATIAL AWARENESS
for All!


For duty and humanity!


We're doing it wrong
RIDGID



THE APACHE SOFTWARE FOUNDATION



# Bkd-Tree: A Dynamic Scalable kd-Tree

Octavian Procopiuc[1] *, Pankaj K. Agarwal[1] **,
Lars Arge[1] ***, and Jeffrey Scott Vitter[2] †

[1] Department of Computer Science, Duke University
Durham, NC 27708, USA
{tavi,pankaj,large}@cs.duke.edu
[2] Department of Computer Science, Purdue University
West Lafayette, IN 47907 USA
jsv@purdue.edu

**Abstract.** In this paper we propose a new index structure, called the Bkd-tree, for indexing large multi-dimensional point data sets. The Bkd-tree is an I/O-efficient dynamic data structure based on the kd-tree. We present the results of an extensive experimental study showing that unlike previous attempts on making external versions of the kd-tree dynamic, the Bkd-tree maintains its high space utilization and excellent query and update performance regardless of the number of updates performed on it.

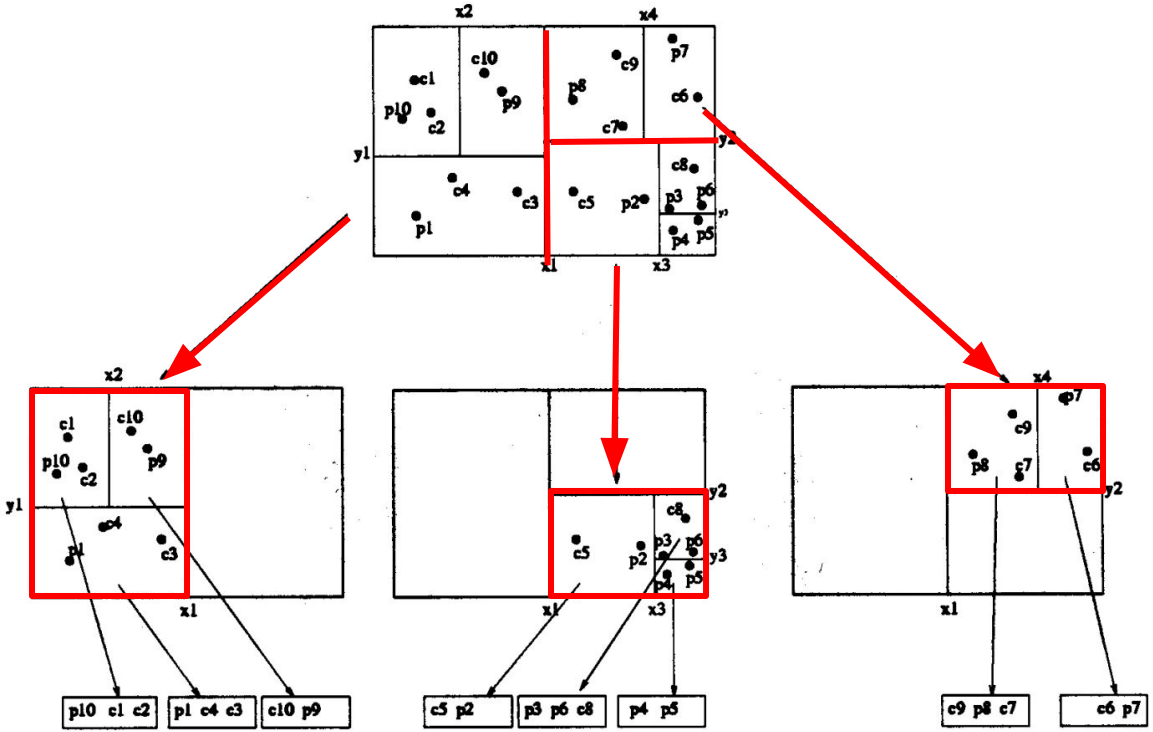
## 1 Introduction

The problem of indexing multi-dimensional point data sets arises in many applications and has been extensively studied. Numerous structures have been developed, highlighting the difficulty of optimizing multiple interrelated requirements that such multi-dimensional indexes must satisfy. More precisely, an efficient index must have high space utilization and be able to process queries fast, and these two properties should be maintained under a significant load of updates. At the same time, updates must also be processed quickly, which means that

* Supported by the National Science Foundation through research grant EIA–9870734 and by the Army Research Office through MURI grant DAAH04–96–1–0013. Part of this work was done while visiting BRICS, University of Aarhus, Denmark.
** Supported by Army Research Office MURI grant DAAH04–96–1–0013, by a Sloan fellowship, by NSF grants ITR–333–1050, EIA–9870724 and CCR–9732787 and by a grant from the U.S.-Israeli Binational Science Foundation.
*** Supported in part by the National Science Foundation through ESS grant EIA–

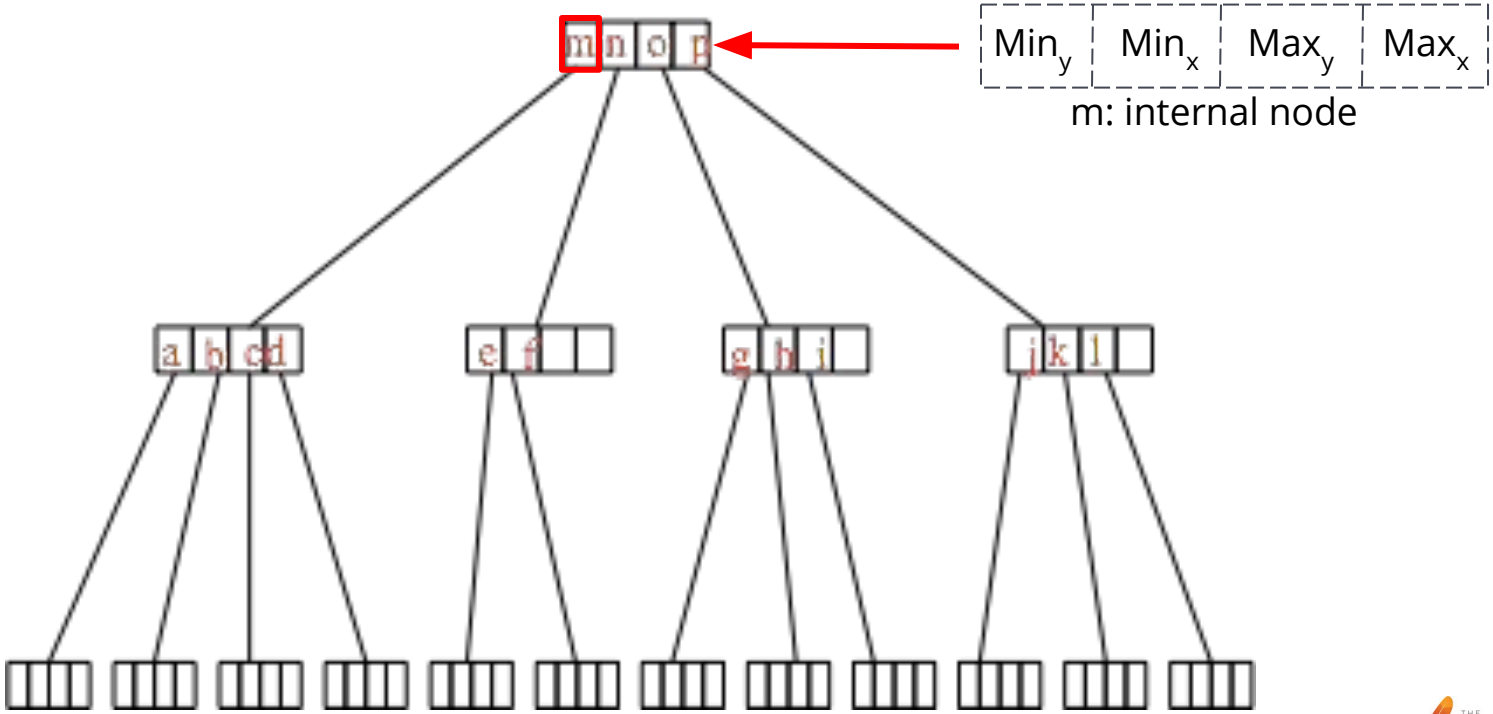
# Block K Dimensional Trees (Bkd) to the rescue!

…the right tool!



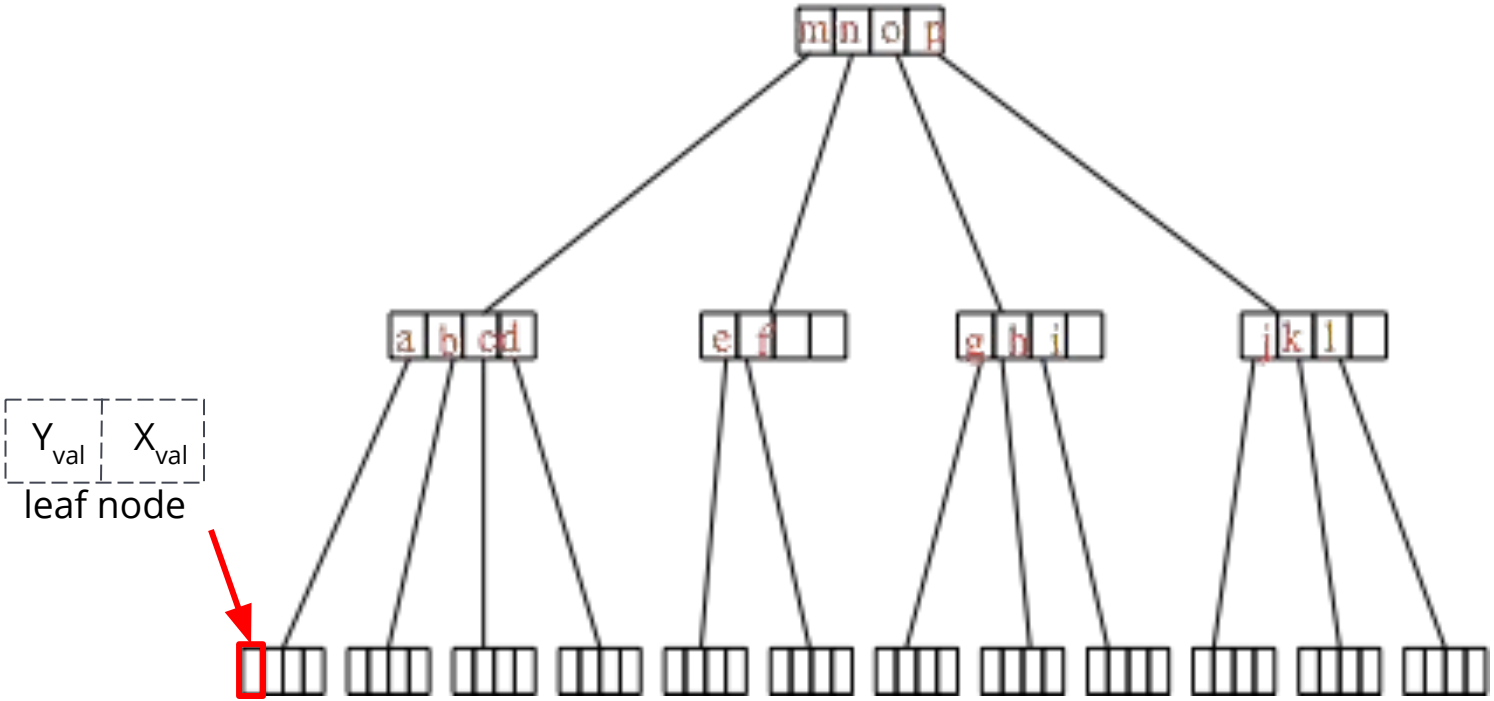
# Block K Dimensional Trees (Bkd) to the rescue!

## ...the right tool!



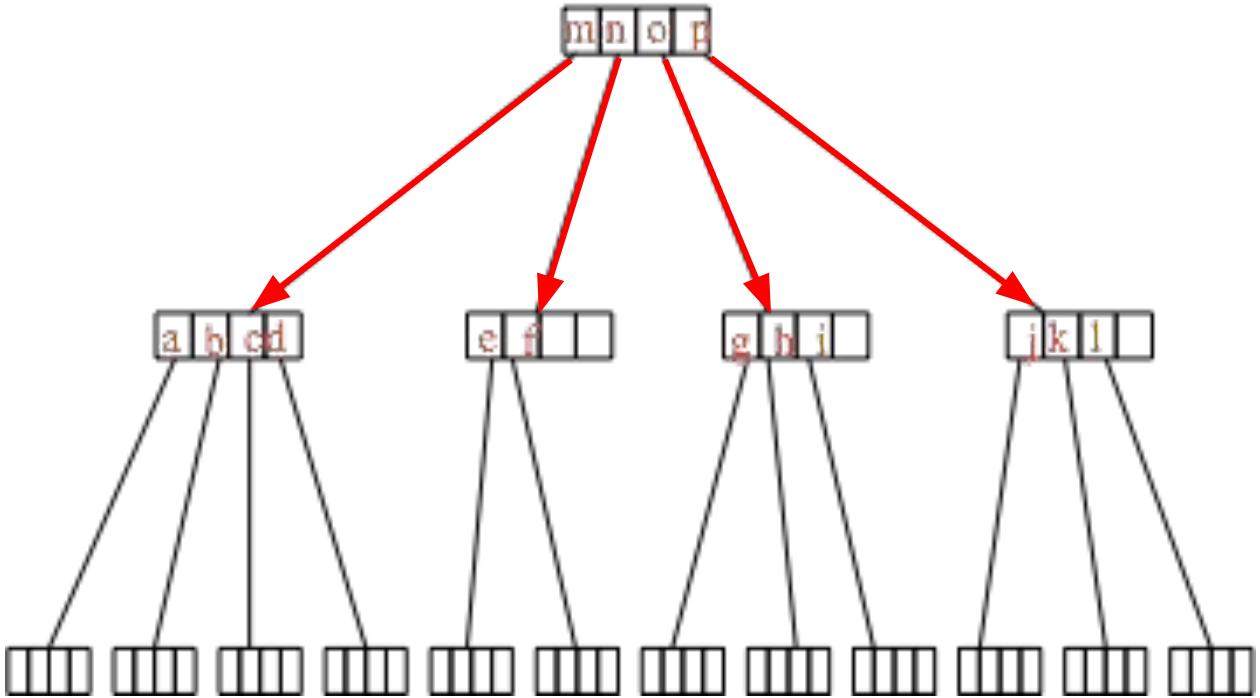
# Block K Dimensional Trees (Bkd) to the rescue!

...the right tool!



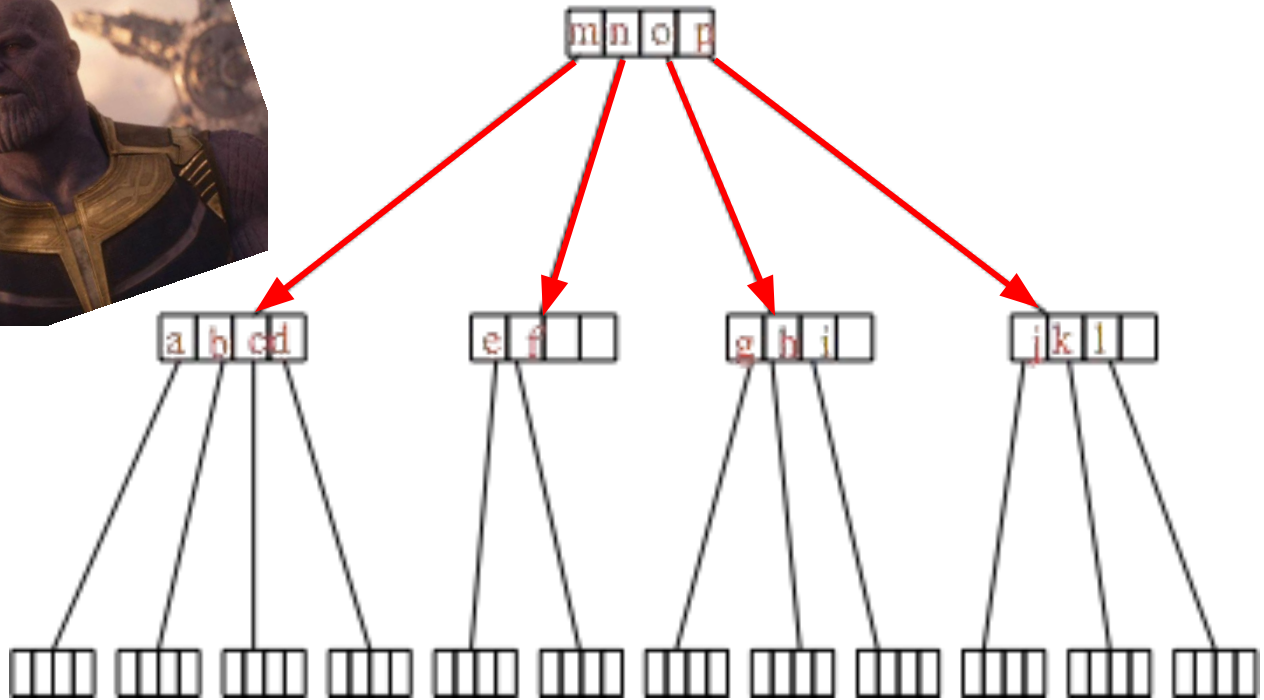
# Block K Dimensional Trees (Bkd) to the rescue!

multi-way...



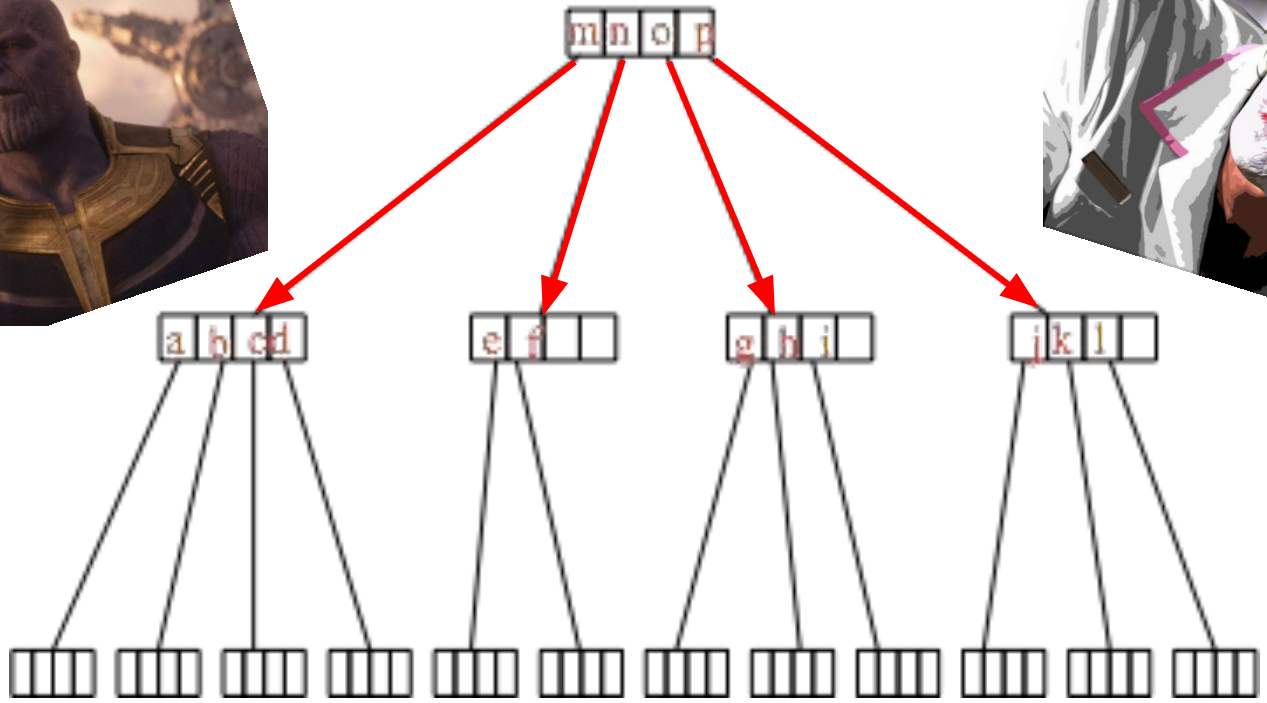
# Block K Dimensional Trees (Bkd) to the rescue!

multi-way… perfectly balanced…



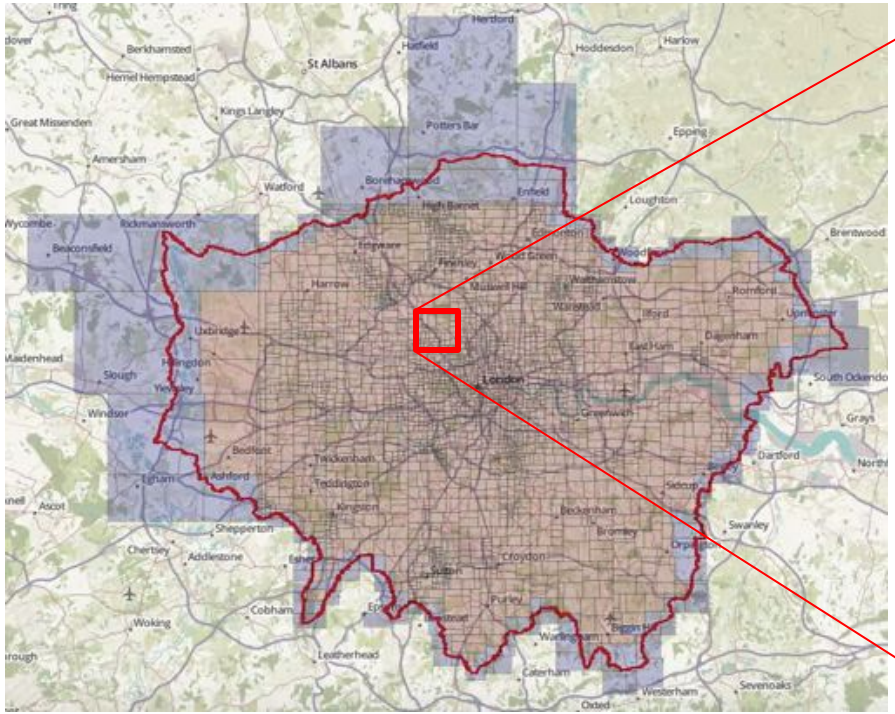
# Block K Dimensional Trees (Bkd) to the rescue!

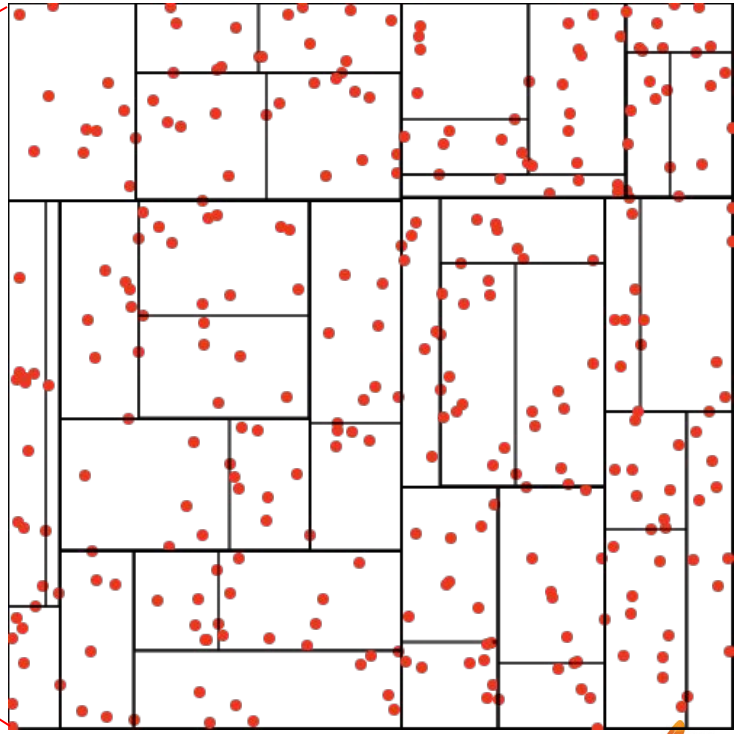
multi-way... perfectly balanced...FAIR...



# Block K Dimensional Trees (Bkd) to the rescue!

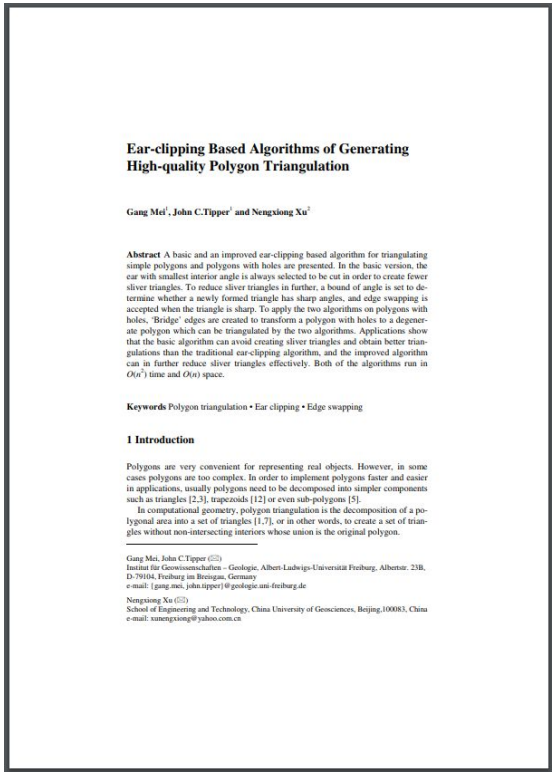
## ...the right tool!



# Okay, cool! But what about GeoShapes?!

## Tessellation!



**Ear-clipping Based Algorithms of Generating High-quality Polygon Triangulation**

**Gang Mei[1], John C.Tipper[1] and Nengxiong Xu[2]**

**Abstract** A basic and an improved ear-clipping based algorithm for triangulating simple polygons and polygons with holes are presented. In the basic version, the ear with smallest interior angle is always selected to be cut in order to create fewer sliver triangles. To reduce sliver triangles in further, a bound of angle is set to determine whether a newly formed triangle has sharp angles, and edge swapping is accepted when the triangle is sharp. To apply the two algorithms on polygons with holes, 'Bridge' edges are created to transform a polygon with holes to a degenerate polygon which can be triangulated by the two algorithms. Applications show that the basic algorithm can avoid creating sliver triangles and obtain better triangulations than the traditional ear-clipping algorithm, and the improved algorithm can in further reduce sliver triangles effectively. Both of the algorithms run in $O(n^2)$ time and $O(n)$ space.

**Keywords** Polygon triangulation • Ear clipping • Edge swapping

**1 Introduction**

Polygons are very convenient for representing real objects. However, in some cases polygons are too complex. In order to implement polygons faster and easier in applications, usually polygons need to be decomposed into simpler components such as triangles [2,3], trapezoids [12] or even sub-polygons [5].

In computational geometry, polygon triangulation is the decomposition of a polygonal area into a set of triangles [1,7], or in other words, to create a set of triangles without non-intersecting interiors whose union is the original polygon.

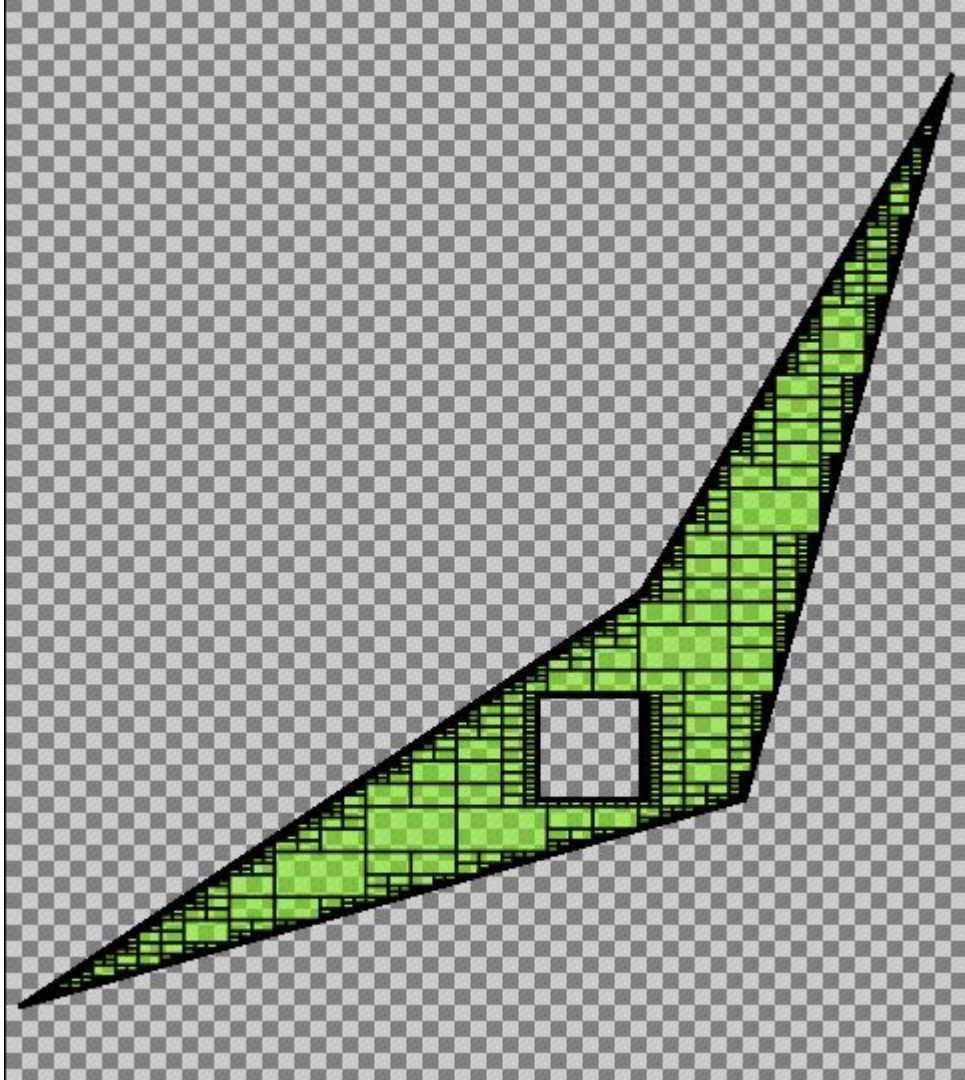
Gang Mei, John C.Tipper (✉)
Institut für Geowissenschaften - Geologie, Albert-Ludwigs-Universität Freiburg, Albertstr. 23B, D-79104, Freiburg im Breisgau, Germany
e-mail: {gang.mei, john.tipper}@geologie.uni-freiburg.de

Nengxiong Xu (✉)
School of Engineering and Technology, China University of Geosciences, Beijing,100083, China
e-mail: xunengxiong@yahoo.com.cn

# Quad Tree Decomposition

## Simple polygon example

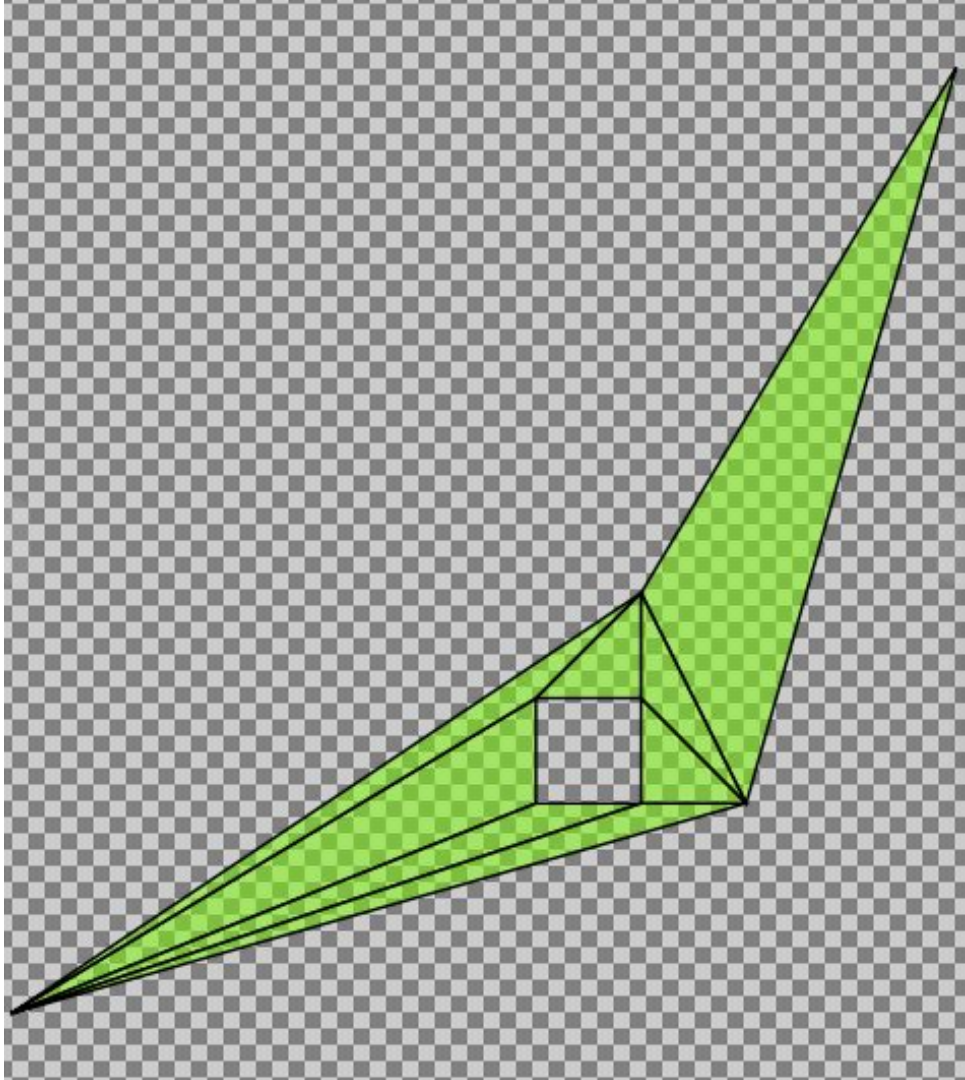
- 8 vertex polygon
- 1º x 1º coverage area
- 3m quad cell resolution
- 1,105,889 terms



# Tessellation Decomposition

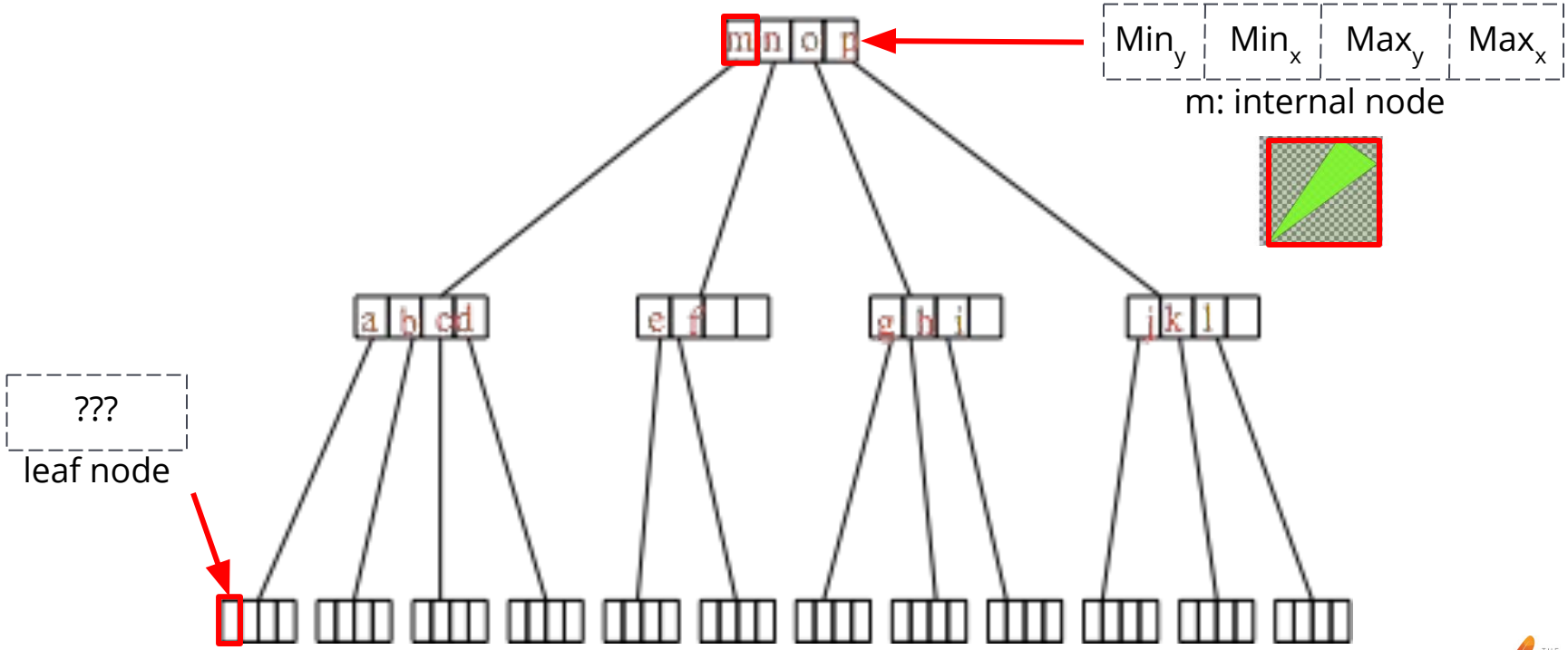
## Simple polygon example

- 8 vertex polygon
- 1° x 1° coverage area
- ~~3m quad cell resolution~~
- 1.11 cm resolution
- ~~1,105,889 terms~~
- 8 terms
- 138,236 : 1 term ratio
- \ (•◡•) /  smaller, faster index!



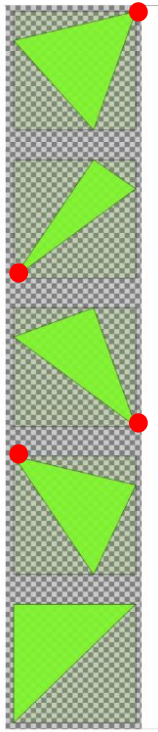
# Tessellation + BKD

## a match made in... Lucene 7.4



# Tessellation + BKD

## a **seven dimension** match made in... Lucene 7.4

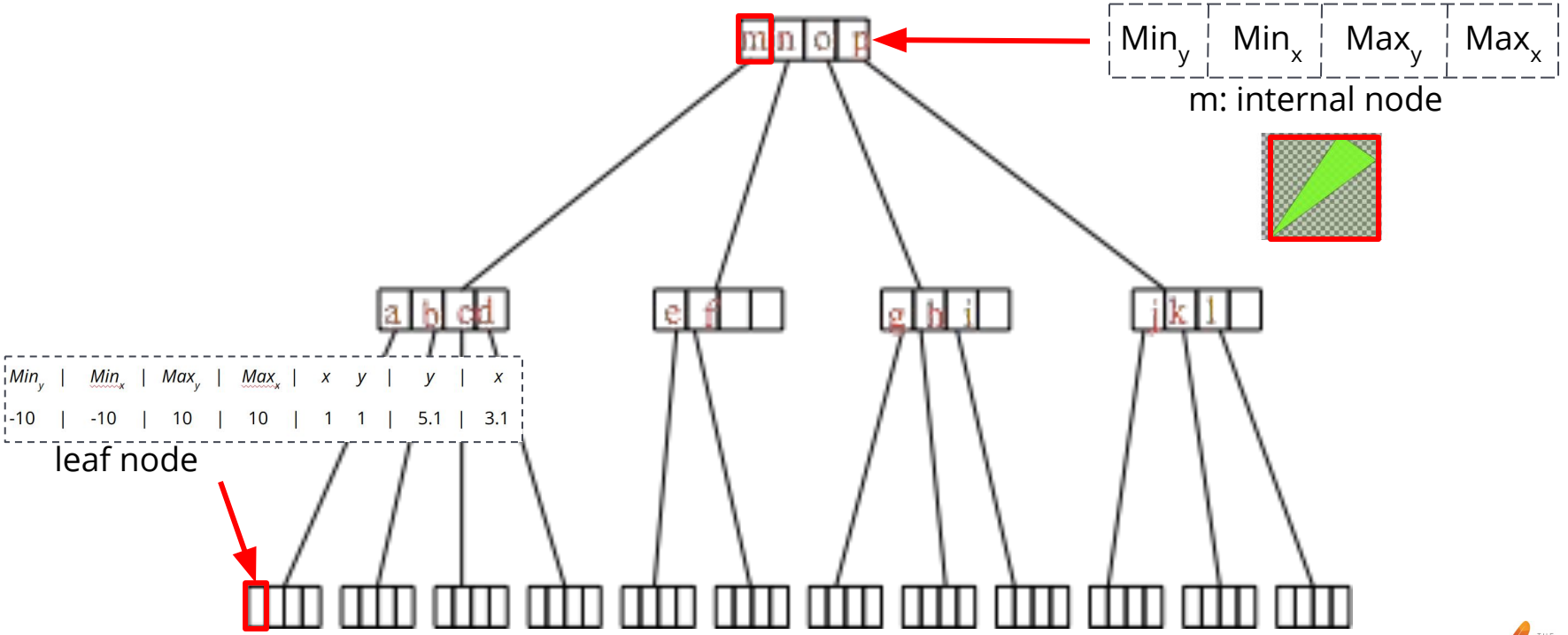


| $Min_y$ | $Min_x$ | $Max_y$ | $Max_x$ | x | y | y | x |
|---|---|---|---|---|---|---|---|
| -10 | -10 | 10 | 10 | 1 | 1 | 5.1 | 3.1 |
| -10 | -10 | 10 | 10 | 0 | 0 | 5.1 | 3.1 |
| -10 | -10 | 10 | 10 | 0 | 1 | 5.1 | 3.1 |
| -10 | -10 | 10 | 10 | 1 | 0 | 5.1 | 3.1 |
| -10 | -10 | 10 | 10 | 0 | 0 | 10 | 10 |

Index Dimensions | Data Dimensions

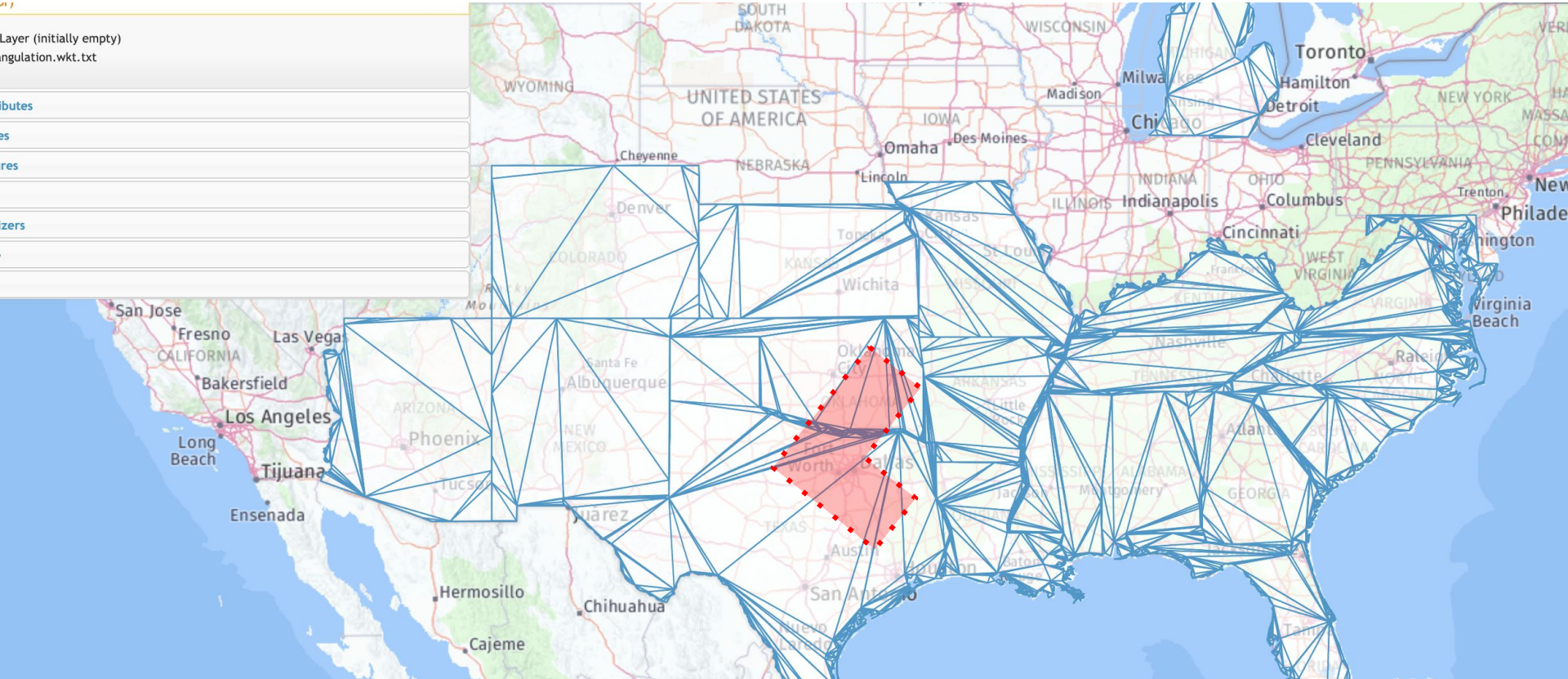
# Tessellation + BKD

## a match made in… Lucene 7.4



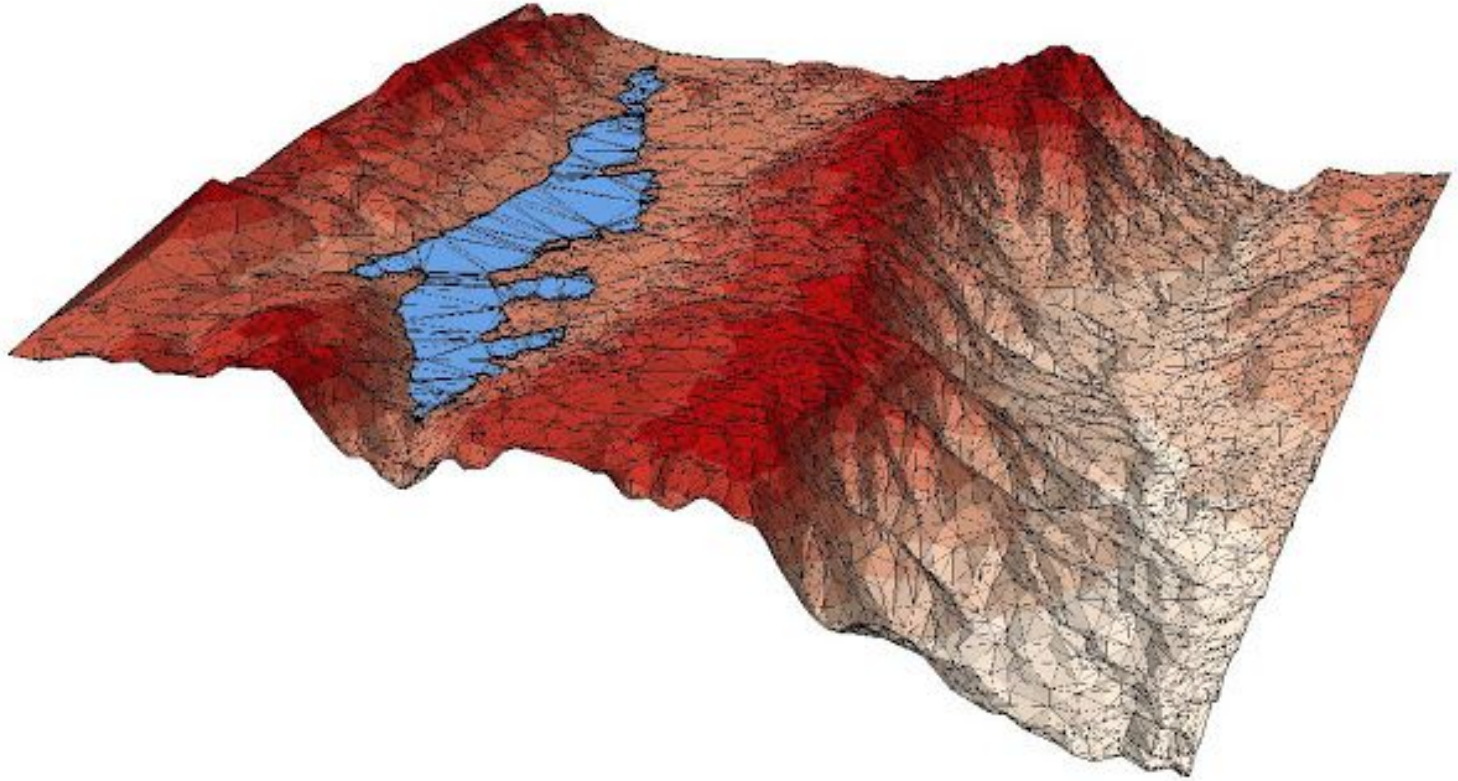
# Smaller, faster, stronger…

## Searching triangle intersections



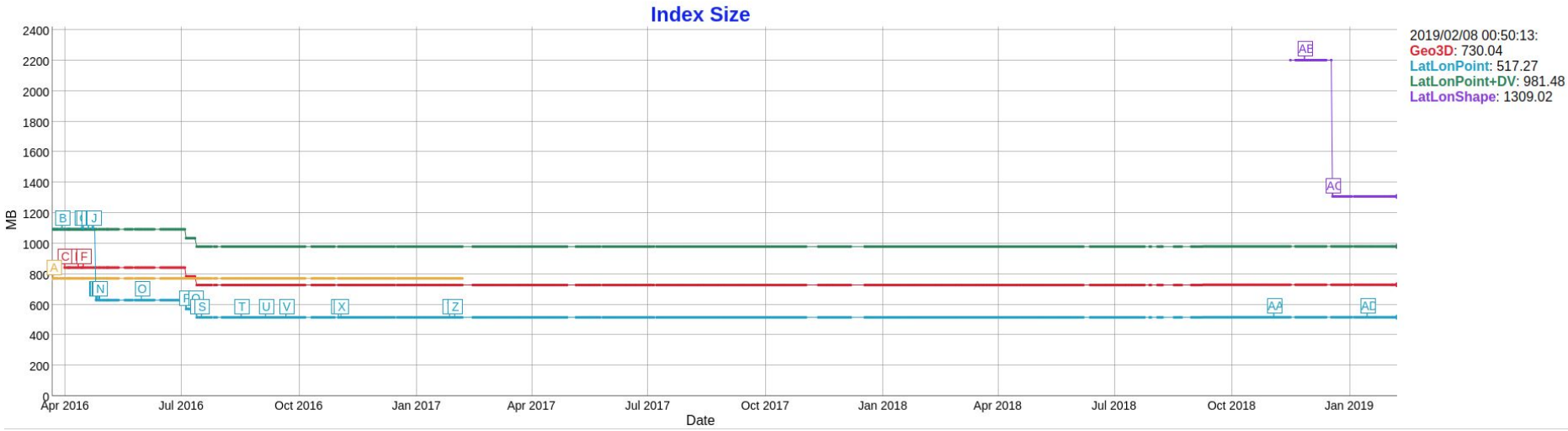
# XYShape for Spatial…in Lucene 8.2+

## Tessellating & Indexing Virtual Worlds



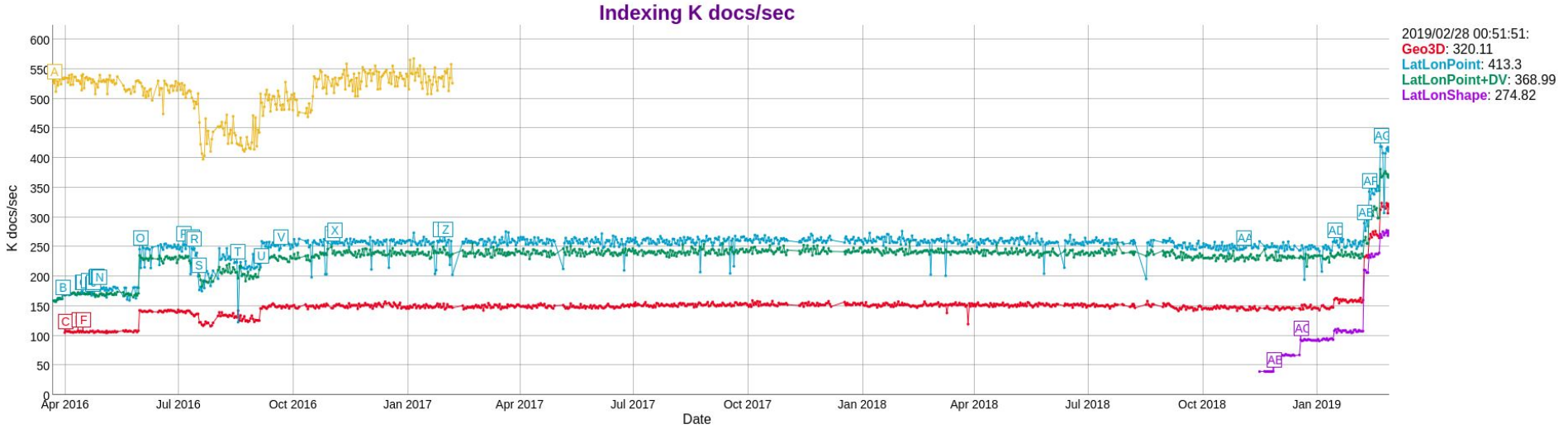
# Smaller, faster, stronger… at scale

## Smaller index…



# Smaller, faster, stronger… at scale

## Faster indexing - Nearly as fast as points!!



# Smaller, faster, stronger… at scale

## Positive feedback…

@imotov: "...test that was crashing a shard on a node with 16GB heap after running for 4.5 hours now finishes in less than 1sec"

@esri: "results show that the **insert operation is 25 to 30 times faster in Elasticsearch 6.6 compare to Elasticsearch 6.5.0/6.4.2.** ...along with better performance we also get accurate results for spatial queries"

# geo search?

pssh… it's simple!

WTF?!


¯_(ツ)_/¯

¯_(ツ)_/¯

¯_(ツ)_/¯

¯_(ツ)_/¯

¯_(ツ)_/¯

¯_(ツ)_/¯

| | |
|---|---|
| `geohash` | Should the geo-point also be indexed as a geohash in the `.geohash` sub-field? Defaults to `false`, unless `geohash_prefix` is `true`. |
| `geohash_precision` | The maximum length of the geohash to use for the `geohash` and `geohash_prefix` options. |
| `geohash_prefix` | Should the geo-point also be indexed as a geohash plus all its prefixes? Defaults to `false`. |
| `ignore_malformed` | If `true`, malformed geo-points are ignored. If `false` (default), malformed geo-points throw an exception and reject the whole document. |
| `lat_lon` | Should the geo-point also be indexed as `.lat` and `.lon` sub-fields? Accepts `true` and `false` (default). |
| `precision_step` | Controls the number of extra terms that are indexed for each lat/lon point. Defaults to `16`. Ignored if `lat_lon` is `false`. |

# So, what's next??

## Use case application goodies…

- Non-geo use cases (`XYShape` type)
  - CAD Drawings *(Design, County / City Planning)*
  - Venue Mapping *(Conferences, Hotels, Theme Parks, Schools)*
  - Sports analysis (*Chicago Cubs, Baseball Advanced Media*)
  - Virtual Gaming & Mapping (*Blizzard, Cyber Security / SIEM*)

***Available 8.2***

- Other coordinate systems (datum / projection support)
  - Non terrestrial planet models
  - Localized projections
  - Custom projections
- Beyond 2D
  - Space - Time *(OGC Moving Features)*
  - *Elevation Modelling (DEM, DTM)*
  - LiDAR *(3D Modelling)*

***In Work***

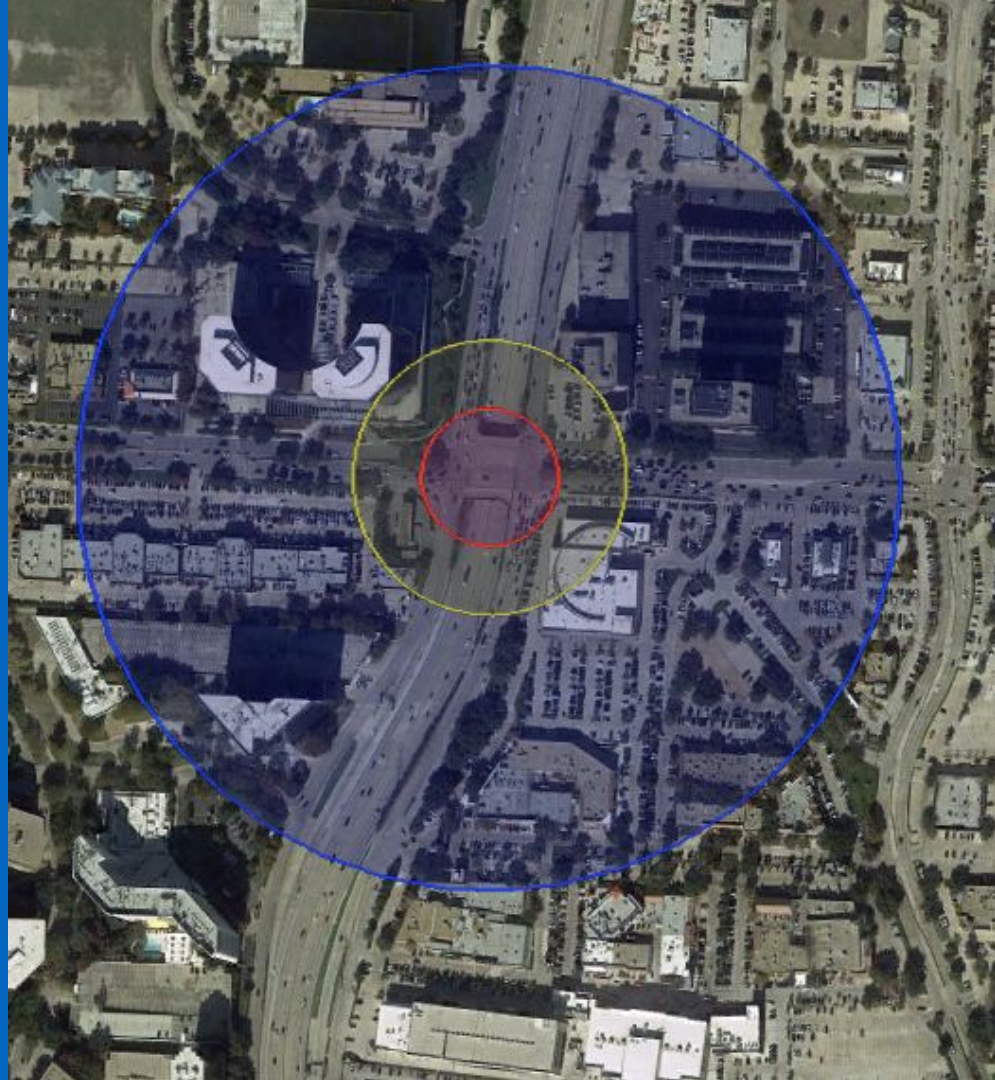
# Aggregations

## Geo & Analytics

# GeoDistance Agg

```
{
   "aggs" : {
      “sf_rings" : {
         "geo_distance" : {
            "field" : "location",
            "origin" : [32.95,
                        -96.82],
            "ranges" : [
               { "to" : 50 },
               { "from" : 50,
                 "to" : 100 },
               { "from" : 100,
                 "to" : 300}
            ]
         }
      }
   }
}
```

# GeoDistance Agg



# GeoGrid Agg

```
{
   "aggs" : {
      “crime_cells" : {
         "geohash_grid" : {
            "field" : "location",
            "precision" : 8
         }
      }
   }
}
```

# GeoGrid Agg


Jena
Gainesville
Palm Coast
Ocala
Daytona Beach
Deltona
Orlando
Titusville
Spring Hill
Kissimmee
Lakeland
Palm Bay
Sebastian
FLORIDA
Vero Beach
Bradenton
Port St. Lucie
North Port
LaBelle
Cape Coral
West Palm Beach
Boca Raton
Naples
75
Fort Lauderdale
Miami
Homestead

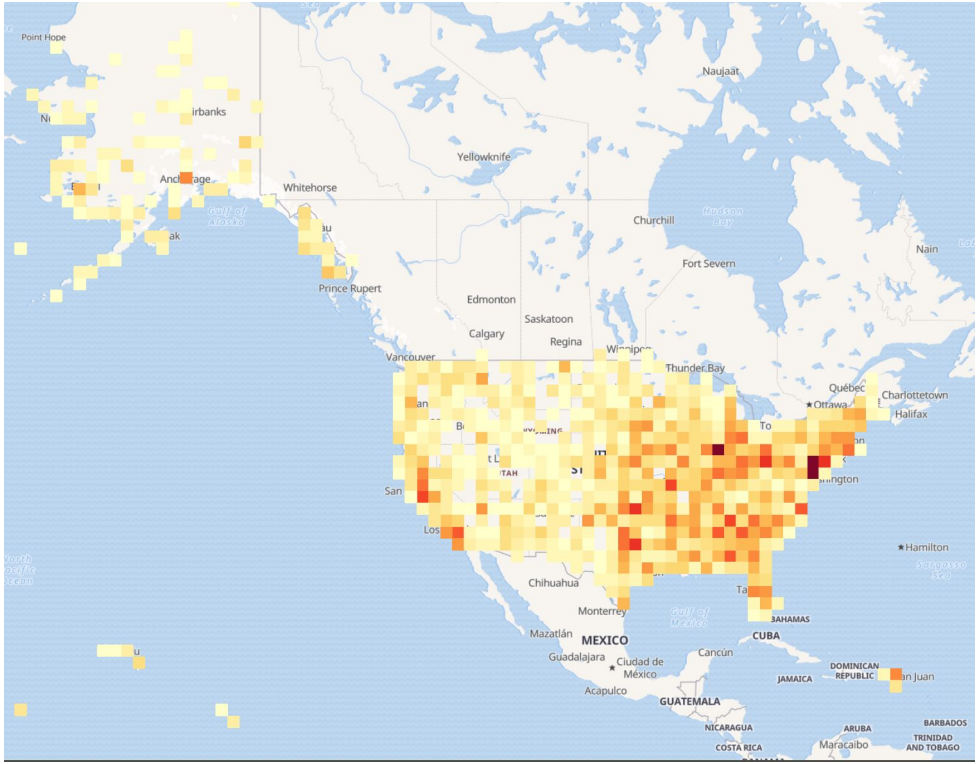
# geotile_grid Aggregation

## Finally, a grid designed for maps!

7.0 introduces geo_tile aggregation

- matches the tiling scheme of well known tile maps in the Web Mercator Projection (EPSG:3857)

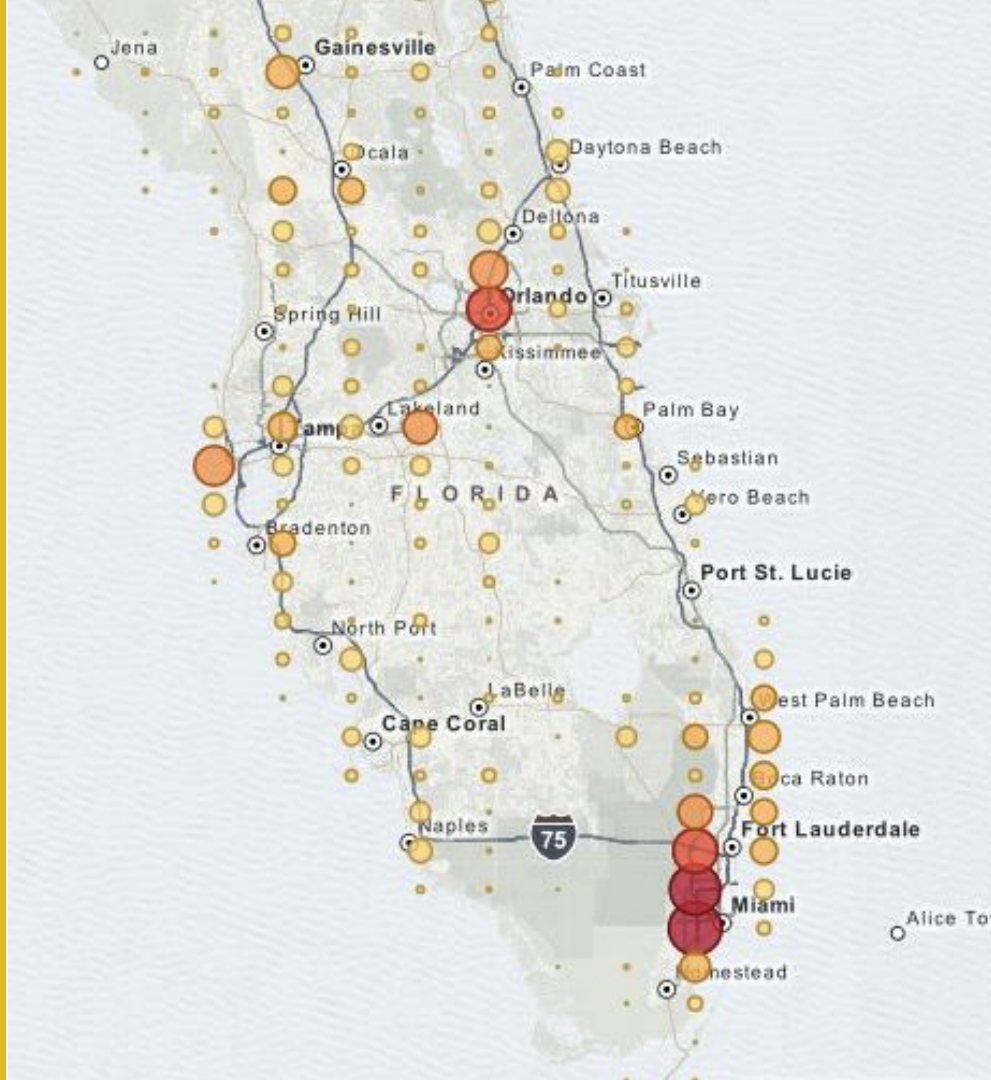
- on web mercator maps, grid cells are
  - actually square
  - preserve an identical aspect ratio at all scales and latitudes



# GeoCentroid Agg

```
"query" : {
  "match" : {
    "crime" : "burglary"
  }
},
"aggs" : {
  "towns" : {
    "terms" : { "field" : "town" },
    "aggs" : {
      "centroid" : {
        "geo_centroid" : {
          "field" : "location"
        }
      }
    }
  }
}
```

# GeoCentroid Agg



# GeoCentroid Agg

# matrix_stats Agg

```
{
    "aggs": {
        "statistics": {
            "matrix_stats": {
                "fields": ["poverty", "income"]
            }
        }
    }
}
```

# matrix_stats Agg


```
"aggregations": {
      "statistics": {
         "doc_count": 50,
         "fields": [{
            "name": "income",
            "count": 50,
            "mean": 51985.1,
            "variance": 7.383377037755103E7,
            "skewness": 0.5595114003506483,
            "kurtosis": 2.5692365287787124,
            "covariance": {
               "income": 7.383377037755103E7,
               "poverty": -21093.65836734694
            },
            "correlation": {
               "income": 1.0,
               "poverty": -0.8352655256272504
            }
         }, {
            "name": "poverty",
            "count": 50,
            "mean": 12.732000000000001,
            "variance": 8.637730612244896,
            "skewness": 0.4516049811903419,
            "kurtosis": 2.8615929677997767,
            "covariance": {
               "income": -21093.65836734694,
               "poverty": 8.637730612244896
            },
            "correlation": {
               "income": -0.8352655256272504,
               "poverty": 1.0
            }
         }]
      }
```

# Geo Aggregations

## more available, and coming soon...

- pca - ML foundation plugin
  - dimensionality reduction
  - image analysis
  - classification / recognition
- geo_stats - In work...
  - Moran's I - measuring spatial auto-correlation
  - Getis-Ord - spatial hot spot analysis


Maps
Flight Data - Dark
Full screen
Inspect
Save
Search
KQL
Last 30 days
Show dates
Refresh
Layers
Add layer
Weather
Raw Live Flights
Live Flights
Airports
Historic Route
Live Flights by Country
Custom Dark TMS
LAYER ACTIONS
Fit to data
Show layer
Go to
Made with NaturalEarth, Elastic Maps Service

# THANK YOU

Follow for updates:

***@nknize***

*apache/lucene-solr*