Grid Indexing	Simple | Utilizes hastable to make index file smaller | If the grid is dense in all parts, no empty cells	Empty cells | O(n^2) | Disk pages are not stored near each | Adjacent cells in a grid index cannot be adjacent on disk
Z-Ordering	Follows some ordering | neighboring cells close to each other | Improve seek time of disk pages	Cannot solve the sparsity problem | Cells are static, sensitive to the data skew.
Geohashing	Interleaved longitude (even) & latitude (odd)	Locations starting with the same prefix are within the same region and neighbors, but the converse is not true
K-d Tree	Adaptive to the distribution of data(1) | Log depth makes it efficient	Not balanced | Curse of Dimentionality | Only one element/node
Quad Tree	Grid index (1) | node -> multiple data | Doen't need reconstructed	Many Leaves | Complex param tuning | Rarely used as index based structure -> larger disk size
R Tree	Physically close objects together | balanced | Fixed node size | Min Oerlap mbrs | (1) | Handles all types of spatial object well | Bulk loading is better(2)	Performance degrades frequent updates | mbrs overlap search costly | may have too much empty space
H3 Indexing	20 separate 2Ds instead of 1 2D | only one distance between centers of neighboring cells | Convenient for modeling | (2)	Not convenient when equal area property is desired | The gnomonic projection used in H3 sacrifices accuracy in calculation | Hexagonal hierarchy produces spatial error when divided into smaller units

H3 - Mode 0 is reserved and indicates an invalid H3 index | Mode 1 is an H3 Cell (Hexagon/Pentagon) index. | Mode 2 is an H3 Unidirectional Edge (Cell A -> Cell B) index | Mode 3 is planned to be a bidirectional edge (Cell A <-> Cell B) | Mode 4 is an H3 Vertex. | 2 + 120*7^r

UDF - Approach 1 – set function as an input to UDF | Approach 2 – Register it with SparkUDF **RASTER DATA FORMATS:** it's a col type, a pixel size, a width and a height, a georeference, a variable number of band, a pixel type per band, etc

Why Sedona?: High Speed, Low Memory Consumption, Ease of Use. It introduces, SRDD – inbuilt support for geometrical, distance operations. Spatial Query Processing Layer, takes into account the execution time and produces a good query execution plan.

Partitioning Steps: Building global spatial grid, assigning cell id to each object, re-partitioning SRDD across the cluster.

K-dTree: knn-O(logn), O(n); **Range**- O(k.n1-1/k) | **QuadTree:** O(Log4N); O(Log4N); Knn- O(k logN), range-O(n)

NDVI = (nir-red)/(nir+red) | ndsi = (green-swir)/(green+swir) | norm-diff vegetation, snow | **NBRI** = (nir-swir)/(nir+swir) | **NDBI** = (swir-nir)/ (swir+nir) | BurnRatio, BurnIndex

||(P,o)|| = min[x1,x2,…,xn] (∑|Pi-xi|^2) | minmaxdist: Given a point P & an MBR R that encloses set O={o_i} of objects. ||(P,o)||<=minmaxdist(P,R)-It means that there exists an object that is no further away than minmaxdist(P,R) | MINDIST:mindist(P,R)<=||(P,o)||-actual closest point is at least mindist distance away.

Heuristics: 1. an mbr r is discarded -> another mbr such that mindist(p, r) > minmaxdist(p,r) ;

Heuristics: 2. O is discarded -> mbr r such that actual-dist(p,o)>minmaxdist(p,r) ;

Heuristics:3. MBR r is discarded -> an object O such that mindist(p,r) > actual-dist(p,o)

BranchNBound: nearestDist=infinite,traverse in dfs ; if curr=leaf compute dist to all objects in leaf, compare with nearestDist, update ; else sort childs based on mindist and put in abl, visit mbrs in sbl until empty, prune abl by h1, update minDist by using h2 on mbrs in abl, apply h3 on each mbr in abl, discard and recurse from 'if' ; return nearestDist.

BestBinF: Bins are looked in increasing order of dist from query point; Dist to a bin is defined as min dist to any point of its boundary; Implemented with a priority queue; Maintain distance to all entries in a pri-que based on mindist,Repeat till all remaining MBRs are pruned-(Inspect next MBR in the pri-que, Add the children to the list,reorder) . **Best bin first search** is optimal -> visits only necessary nodes, but -> store a very large priority queue in memory BB uses small lists for each node by pruning with both MINDIST and MINMAXDIST $ CRS: Vector:Geodatabase; KML/KMZ;OpenStreetMap;ShapeFile(.shp,.shx,.dbf,.prj,.xml);GeoJSON;WKB/WKT | Raster:JPEG;GIF;PNG; BMP;GeoTIFF $

Spark - Jobs, stages, tasks, DAG, Executor. Properties of Spark RDD• In-memory computation• Lazy evaluation• Fault tolerance• Immutable• Partitioning• Persistence• Coarse-grained operations. Spark RDD Operations• Transformations Takes an RDD as input and produce one or more RDDs as output• Actions. RDD operations that produce non-RDD values. One of the ways to send results from .executors to the driver.➢ Collect() ➢ Count() Advantages of Apache Spark • Speed • Developer friendly API • Advanced DAG execution engine • Numerous storage engines: HDFS, Cassandra, HBase, etc. Why apache sedona is eff => main components => Spatial RDD layer and Spatial Query Proc