GEOSPATIAL DATA MANAGEMENT IN APACHE SPARK

Presented by:
Jia Yu
Mohamed Sarwat

About Us

- Data Systems Lab: a research lab at Arizona State University
- Geospatial databases, distributed geospatial data management
- GeoSpark: open-sourced in April 2015

GeoSpark

http://datasystemslab.github.io/GeoSpark/

Google “GeoSpark ASU”

Arizona State University

- Spatial RDD, SQL, DataFrame on Apache Spark
- Distributed map visualization (Viz SQL)
- Integrated with Apache Zeppelin
- Built-in large-scale traffic simulator

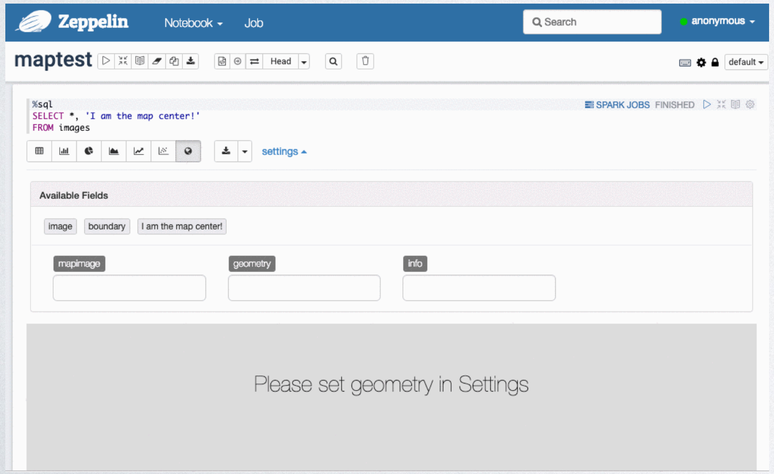
mobike 摩拜单车

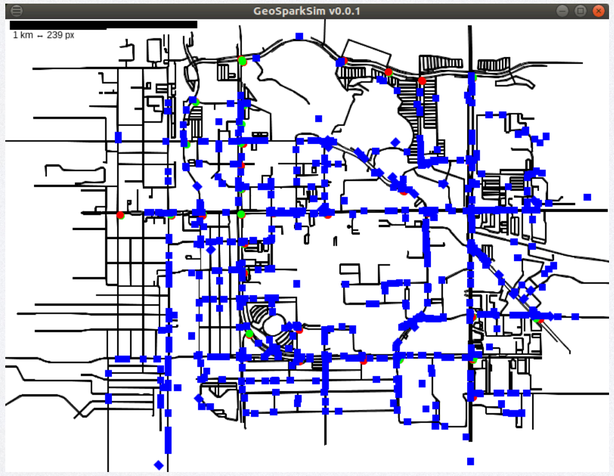
GYANA

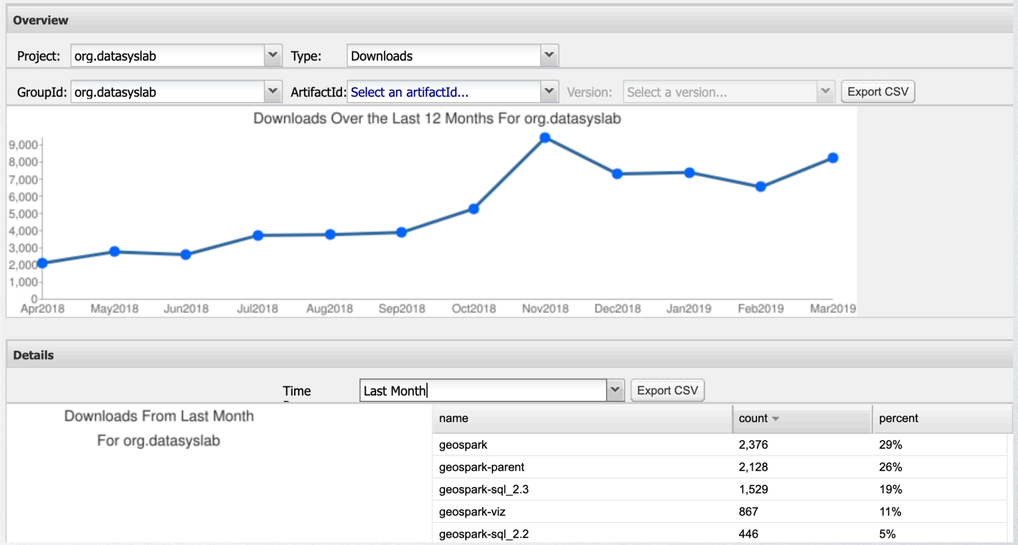
Blue DME

In production!

8K - 10K monthly downloads







Outline

Big Geospatial Data

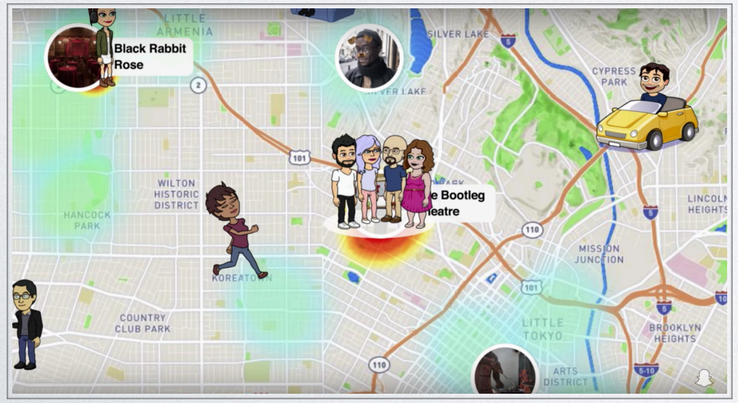
Manage Spatial Data

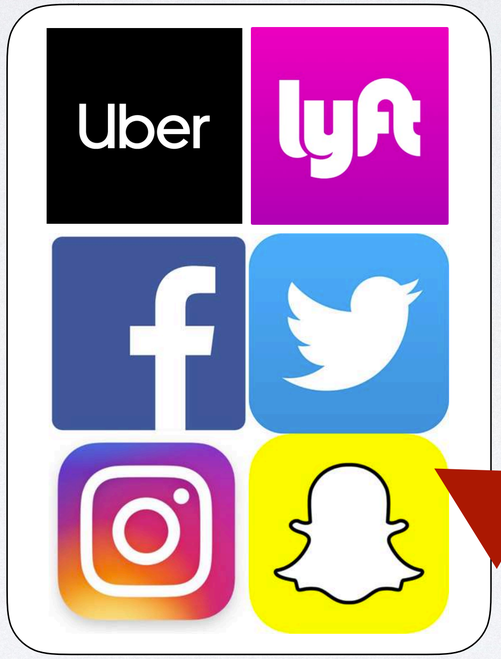
Manage Spatio-Temporal Data

Spatial Data Analytics in Spark

Geospatial Data

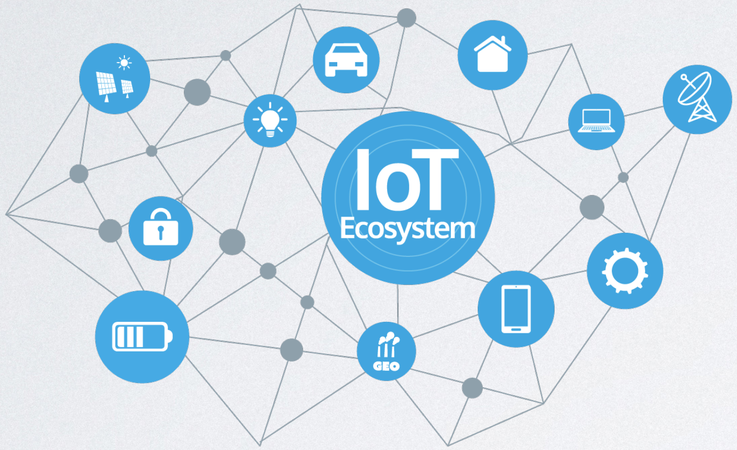
- Mobile devices - 4.68 billion in 2019

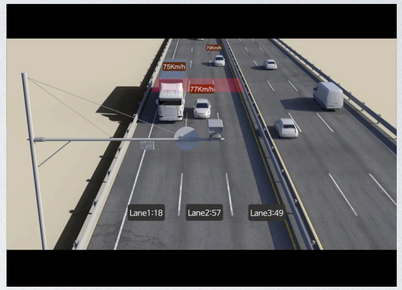




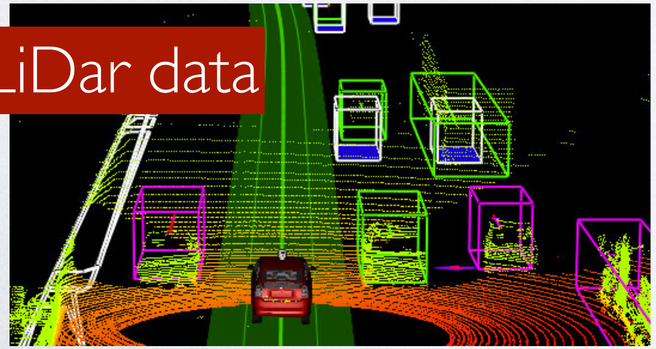
Geospatial Data

- IoT sensors in Smart City: 7 billion in 2019





Massive LiDar data



1.5 billion taxi trips

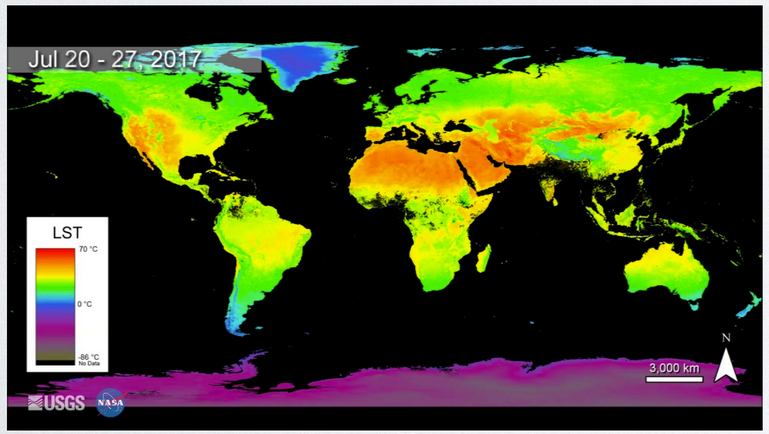
NYC
Taxi & Limousine Commission

Geospatial Data

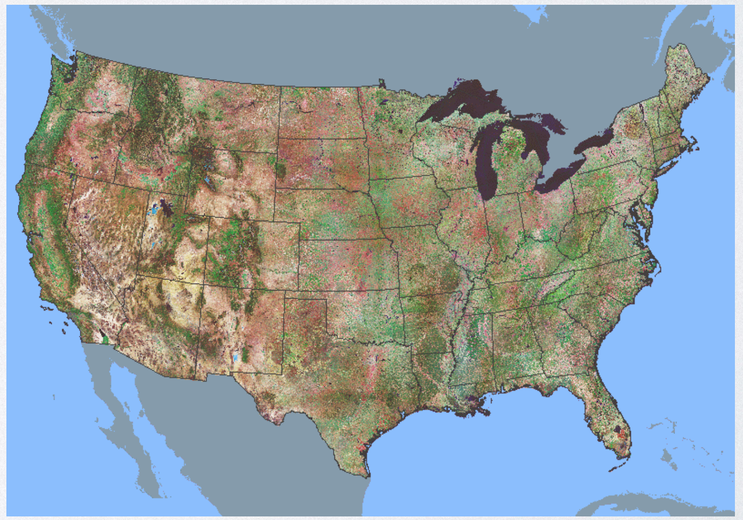
- Climate monitoring: 22 PB satellite imagery data

Raster array format: GeoTiff and HDF format

Land, Ocean, Atmosphere
data from spacecraft

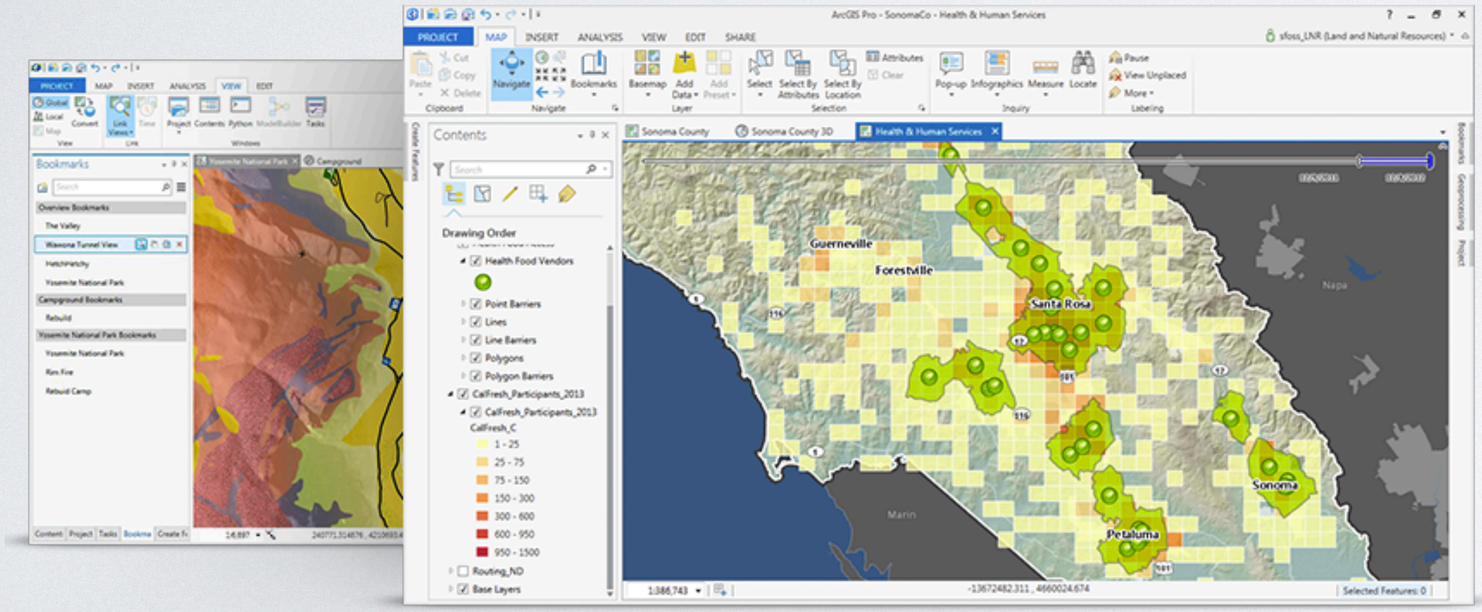


MODIS Land Surface Temperature



Geospatial Data Frameworks

- Classic - single machine DBMS or GIS tools



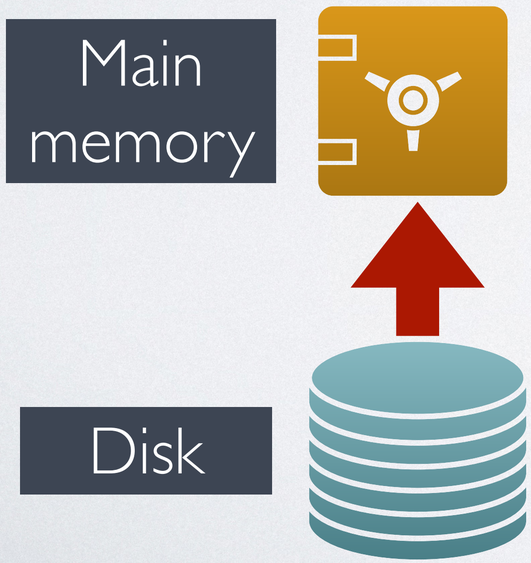
ArcGIS

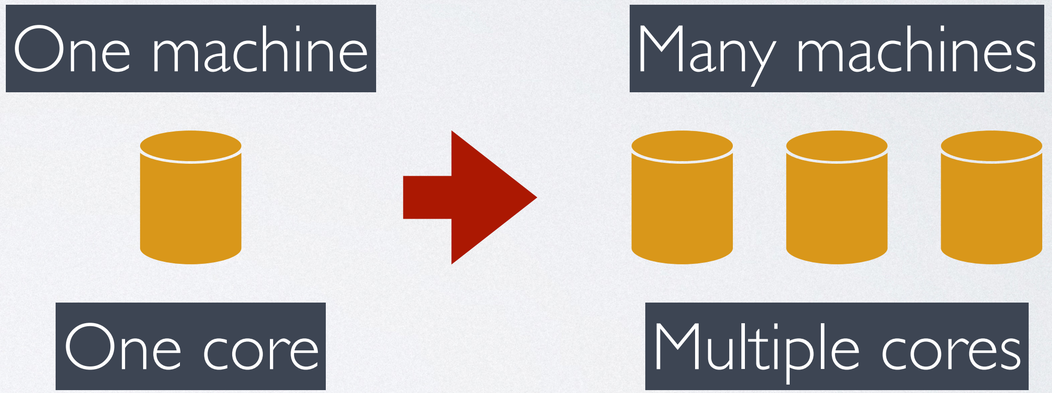
QGIS

PostgreSQL + PostGIS

Geospatial Data Frameworks

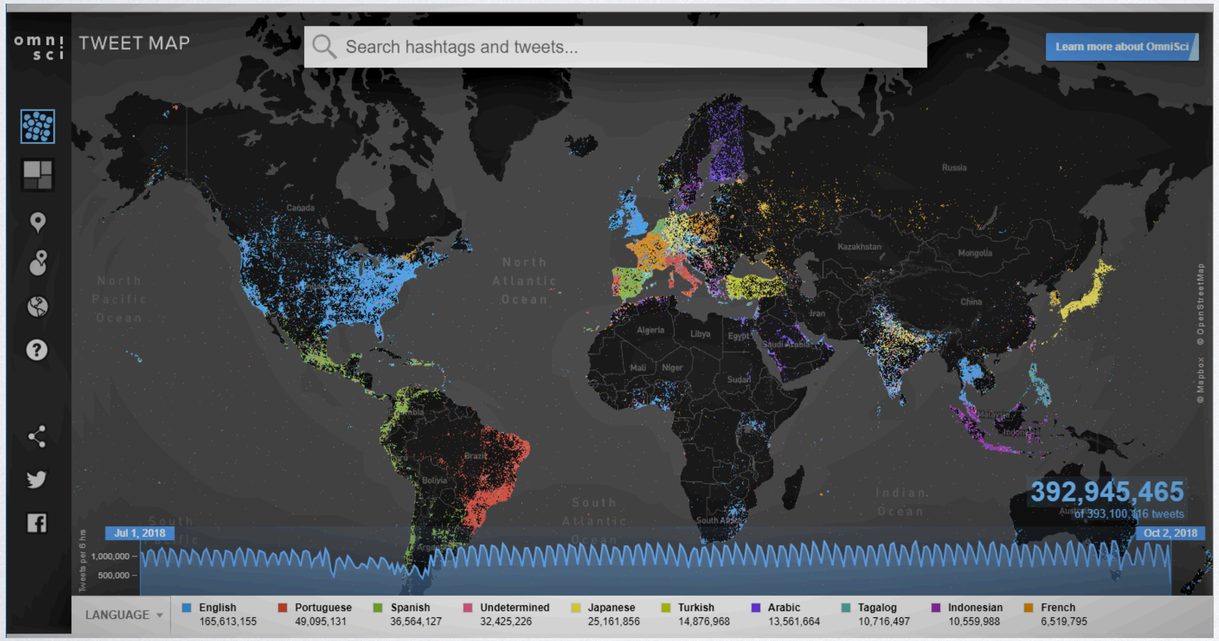
- Single machine solutions suffer from the **scalability issue**
- In Database community, something is happening..
  - Parallel execution
  - In-memory computation





New DBMS Approaches

- Parallel execution
  - GPU acceleration

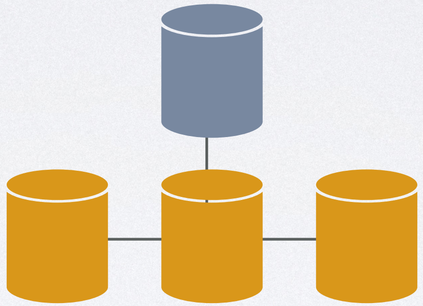


kinetica

MAPD

omni sci

Cluster (Distributed) Computing Approaches



Manage Spatial Data in Spark?

I want to manage spatial data in Spark!

Not that easy!

- No spatial data type support
- No spatial index
- No spatial query

Outline

Big geospatial data

Manage spatial data

Manage Spatio-Temporal Data

Spatial Data Analytics in Spark

Manage Spatial Data

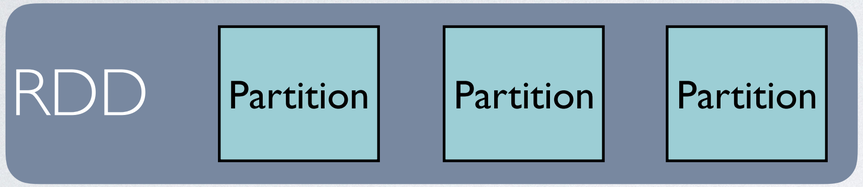
+ APACHE Spark™

- Spatial data partitioning
- Spatial indexing
- Spatial queries
- Optimization
- Language, spatial object support

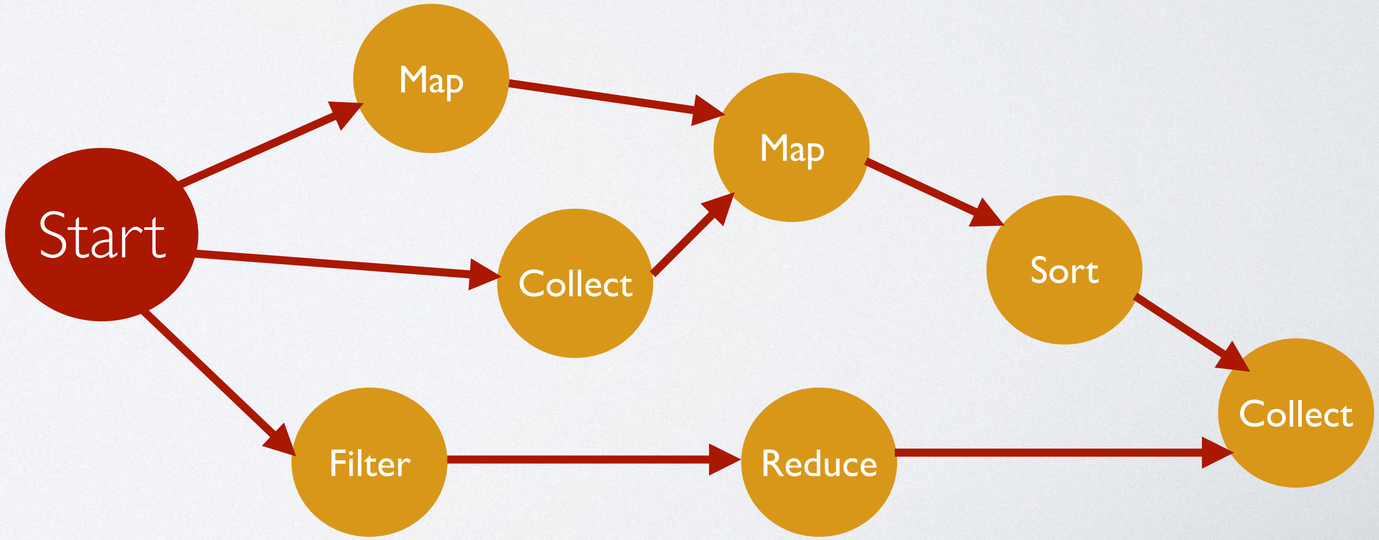
Spark in a Nutshell

Arizona State University

- Resilient Distributed Dataset

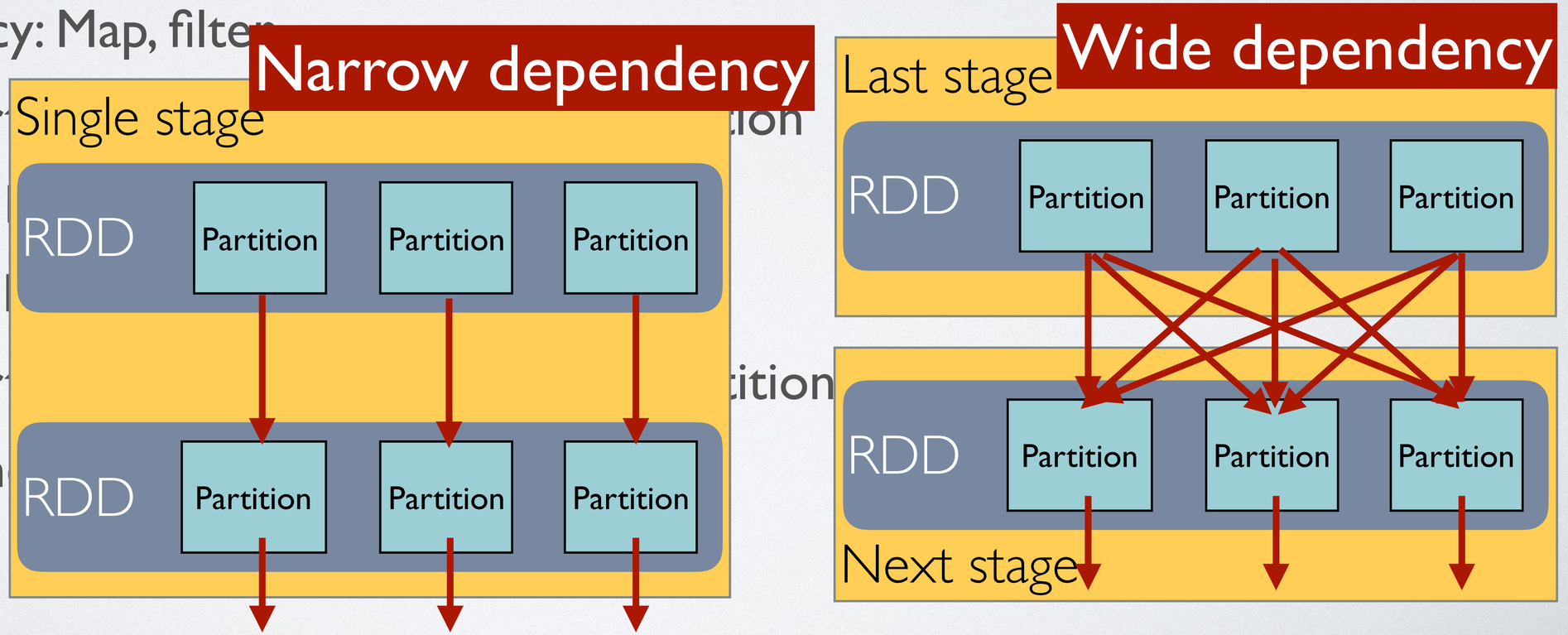


- Intermediate data in-memory
- Directed Acyclic Graph (DAG) scheduler
- Spark SQL / DataFrame
- Spark Structured Streaming
- Spark GraphX / GraphFrame



Spark in a Nutshell

- Action / Transformation
  - Action: Count, Take
  - Transformation: yield new RDD, such as map, filter, reduce, join, GroupBy
- Narrow dependency: Map, filte
  - Parent RDD par
  - No data shuffle,
- Wide dependency:
  - Parent RDD par
  - Data shuffle, gen



Spatial in Spark: Design Goal

Arizona State University

Reduce wide dependencies

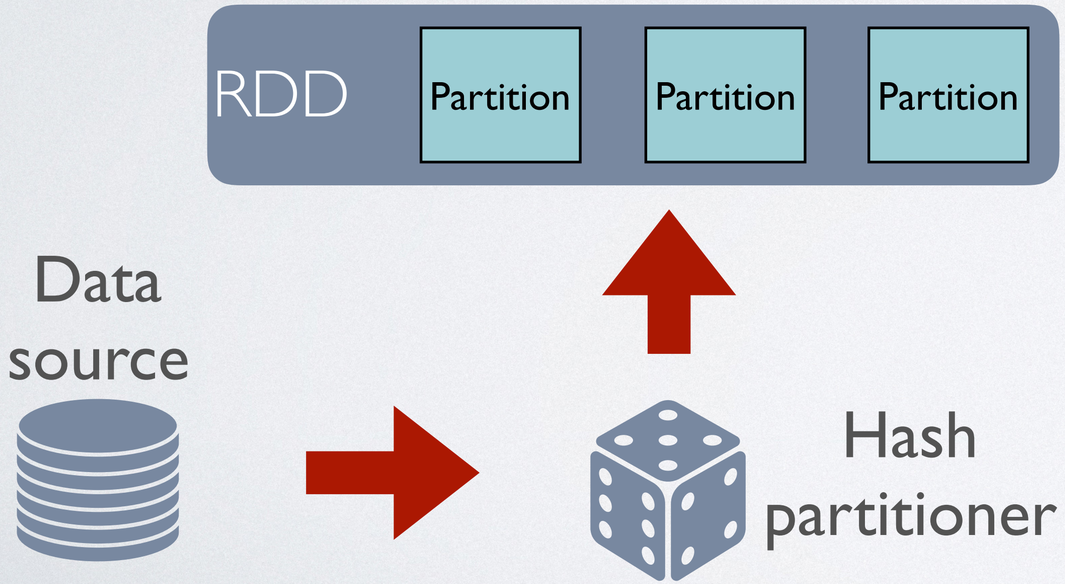
Speed up local computation

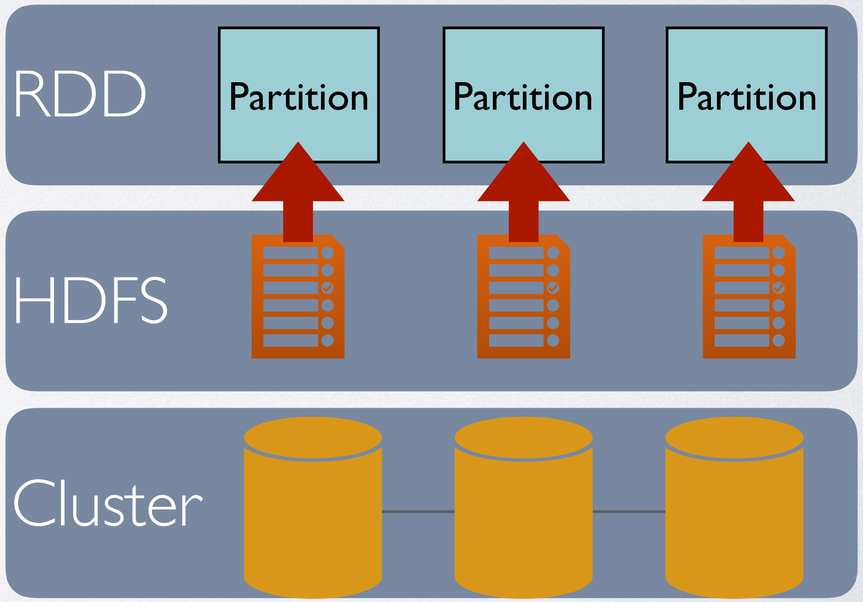
Reduce the memory footprint

Manage Spatial Data

Load Data Into Spark RDD

- Loading data into Spark RDD or DataFrame
  - Partition data into 64 MB chunks using Hash partitioner
  - If the data is already partitioned, keep the original partitions

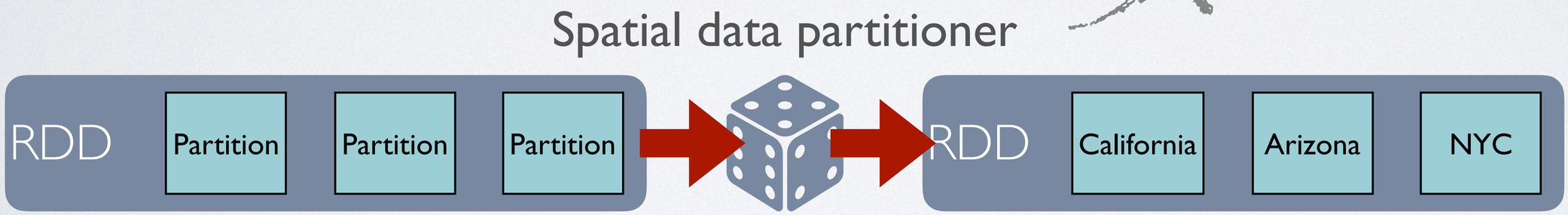




https://spark.apache.org/docs/2.3.1/api/java/org/apache/spark/HashPartitioner.html

Spatial Data Partitioning

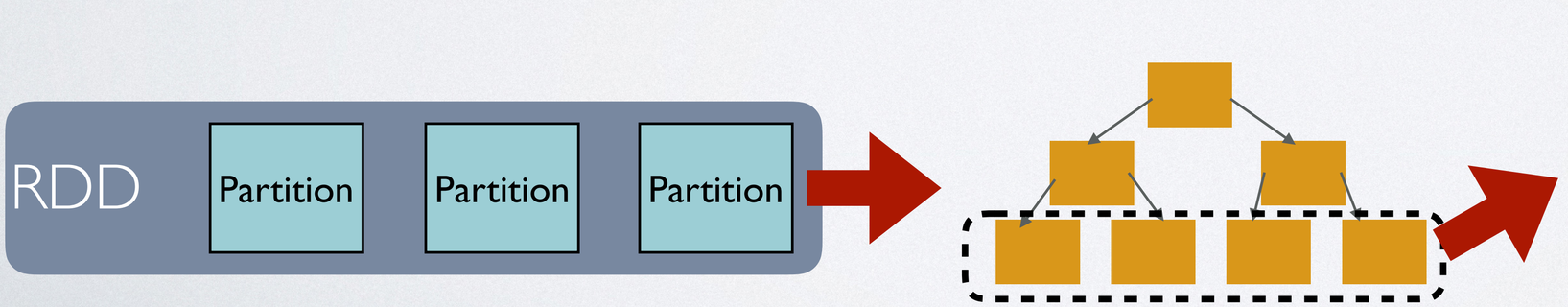
- Repartition data in RDD
  - Partition by spatial proximity
  - Still achieve load balance
  - API: CustomPartitioner

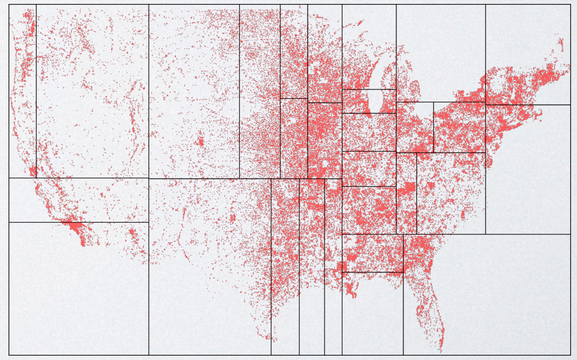


Yu, Jia, Zongsi Zhang, and Mohamed Sarwat. "Spatial data management in apache spark: the GeoSpark perspective and beyond." *GeoInformatica* (2018): 1-42.

Spatial Data Partitioning

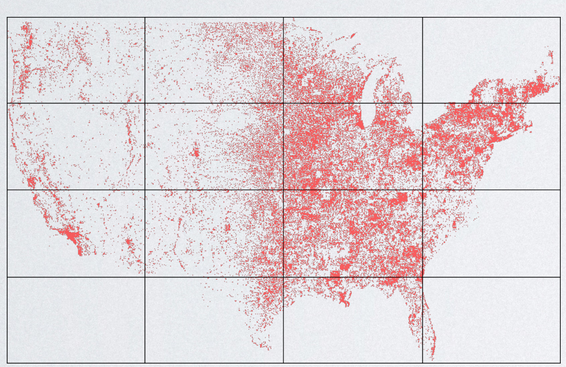
- Spatial partitioning algorithm
  - Randomly sample the RDD
  - Build a KD-Tree/Quad-Tree/R-Tree on the sample
  - Take the leaf nodes of the tree as the global partition file
  - Re-partition the RDD according to the partition file



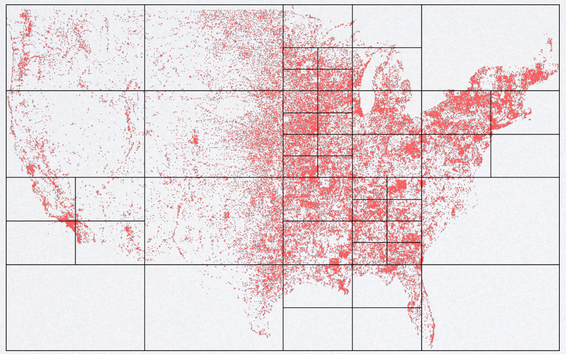


Spatial Data Partitioning

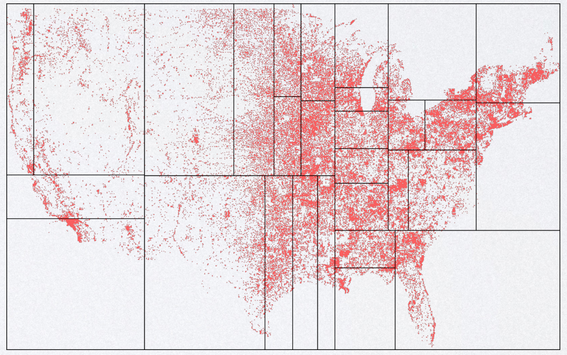
- Common spatial partitioning grids
  - Space partition: Uniform, KD-Tree, Quad-Tree
  - Data partition: R-Tree, an overflow partition due to sampling



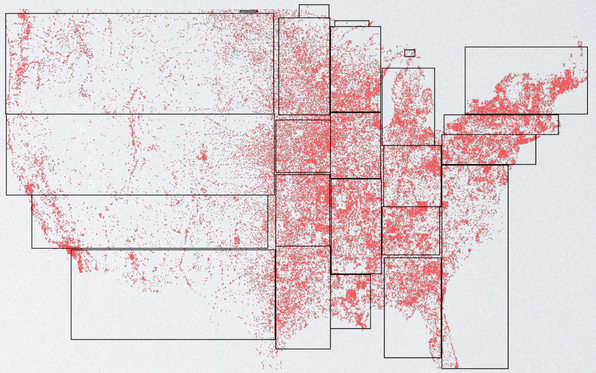
Uniform grids



Quad-Tree



KD-Tree

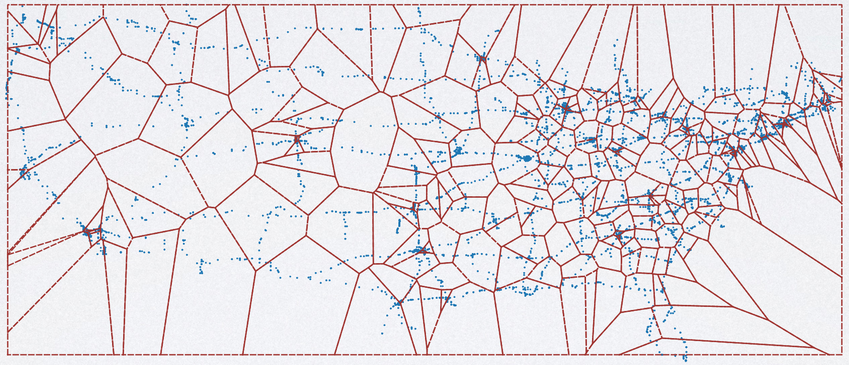


R-Tree

Xie, Dong, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. "Simba: Efficient in-memory spatial analytics." In *Proceedings of the 2016 International Conference on Management of Data*, pp. 1071-1085. ACM, 2016.

Spatial Data Partitioning

- Other common spatial partitioning grids
  - Voronoi diagram, Z-curve, Hilbert-curve

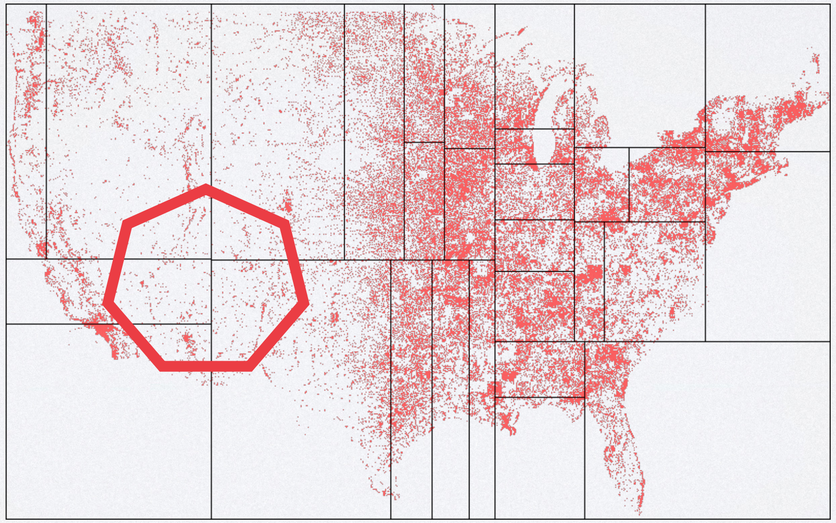


Magellan: https://github.com/harsha2010/magellan

Whitman, Randall T., Michael B. Park, Bryan G. Marsh, and Erik G. Hoel. "Spatio-Temporal Join on Apache Spark." In *SIGSPATIAL* 2017.

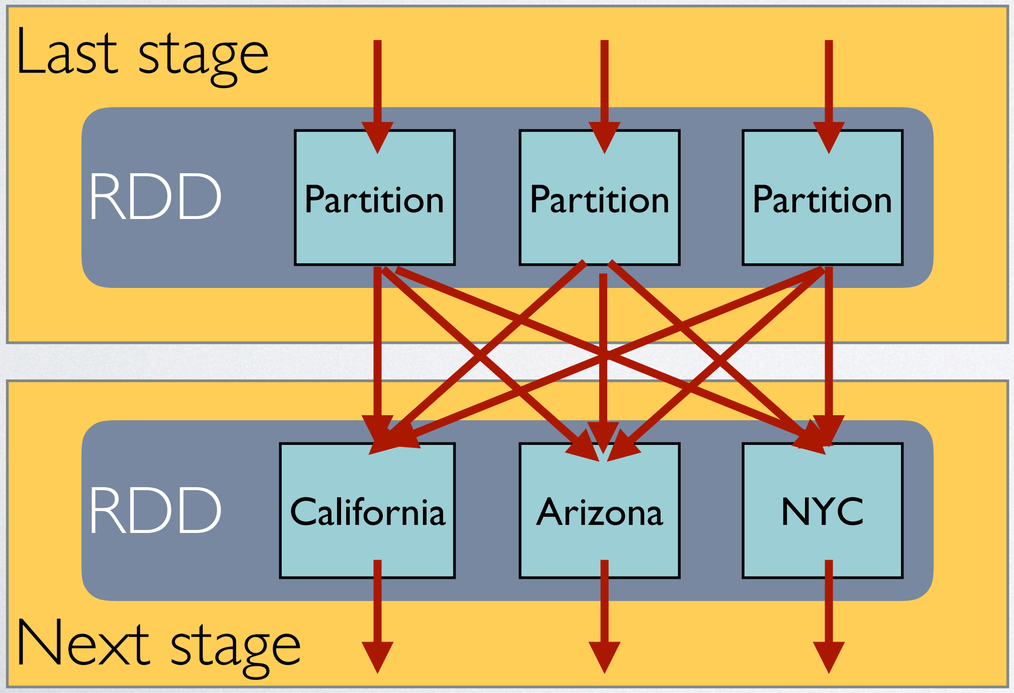
Spatial Data Partitioning

- Objects that intersect many boundaries
  - Duplicate them to all intersected partitions
  - Need duplicate removal after queries



Spatial Data Partitioning

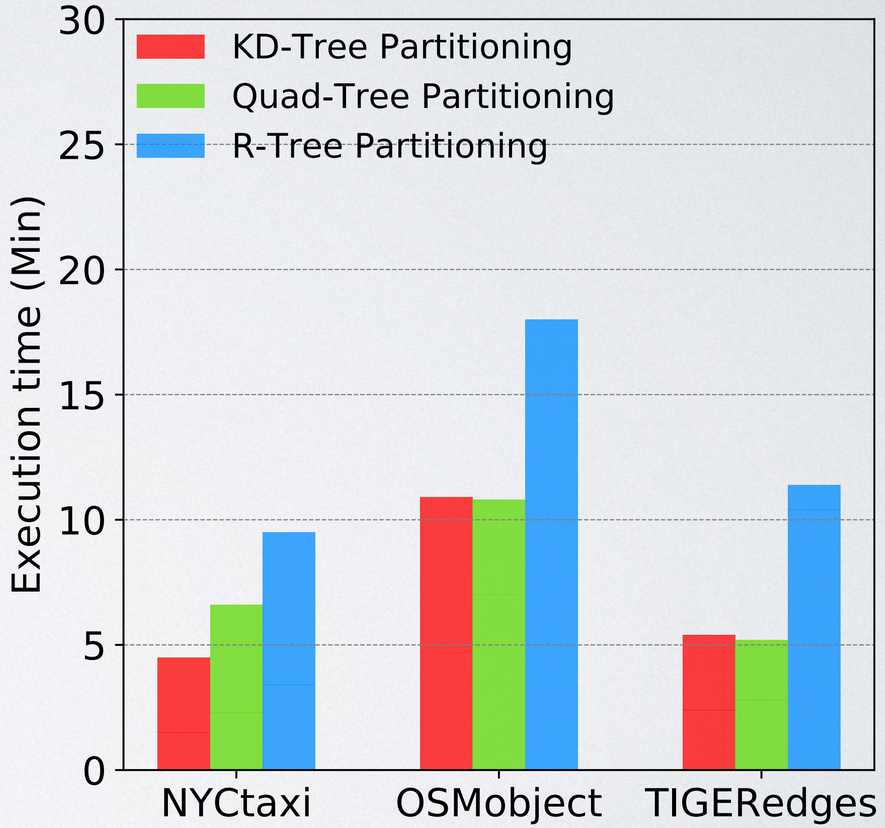
- DAG and data shuffle:
  - Each spatial partitioning is a wide dependency
  - Wide dependency will incur a data shuffle



Wide dependency

Spatial Data Partitioning

- Performance
  - Measured using spatial join query
  - Join with 171 thousand polygons
    - NYCtaxi: 1.3 billion points
    - OSMobject: 263 million polygons
    - TIGERedges: 72.7 million line strings
- Cluster settings: Four workers, one master, 192 cores, 400 GB Memory



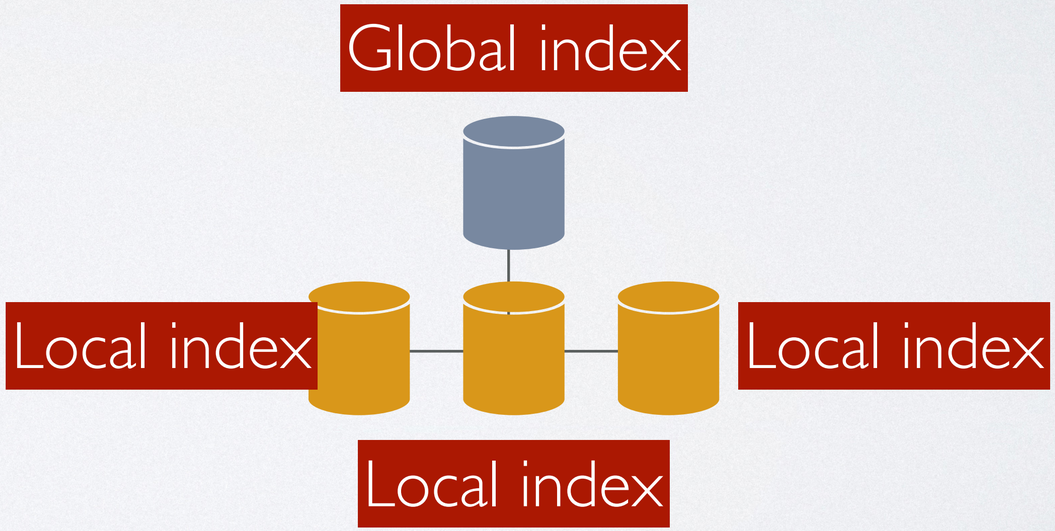
Manage Spatial Data

+

- Spatial data partitioning
- **Spatial indexing**
- Spatial queries
- Optimization
- Language, spatial object support

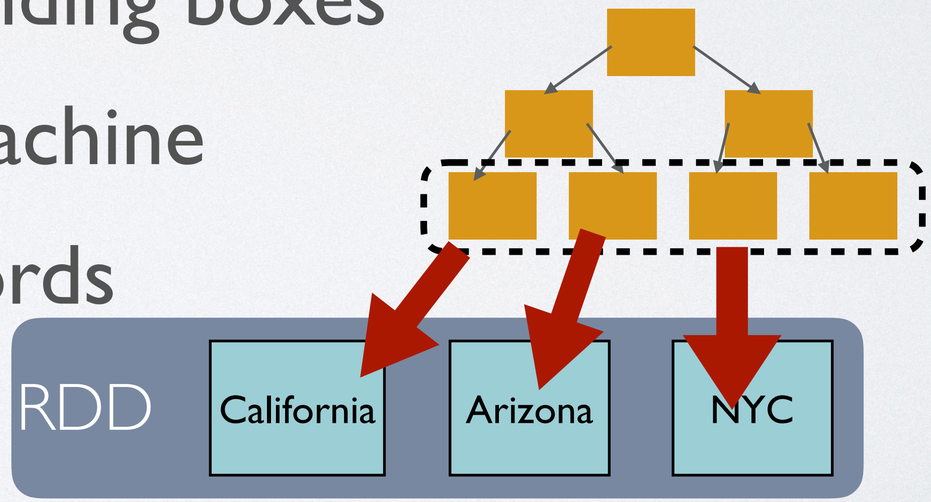
Spatial Indexing

- Traditional indexing
  - Not work because of the huge storage overhead
  - Data in different partitions
- Distributed spatial indexing
  - Global index
  - Local index



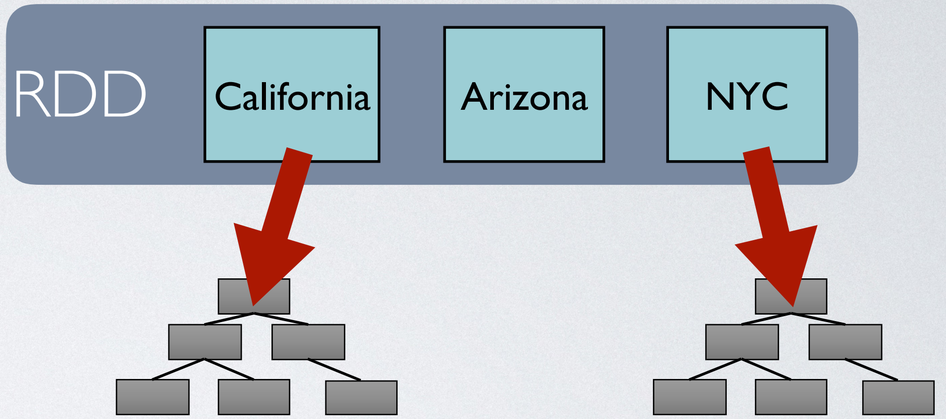
Spatial Indexing

- Global index
  - Remember the tree built for spatial partitioning?
    - Two birds, one stone!
  - Use it to index partition bounding boxes
  - Lightweight, on the master machine
  - No entries for individual records



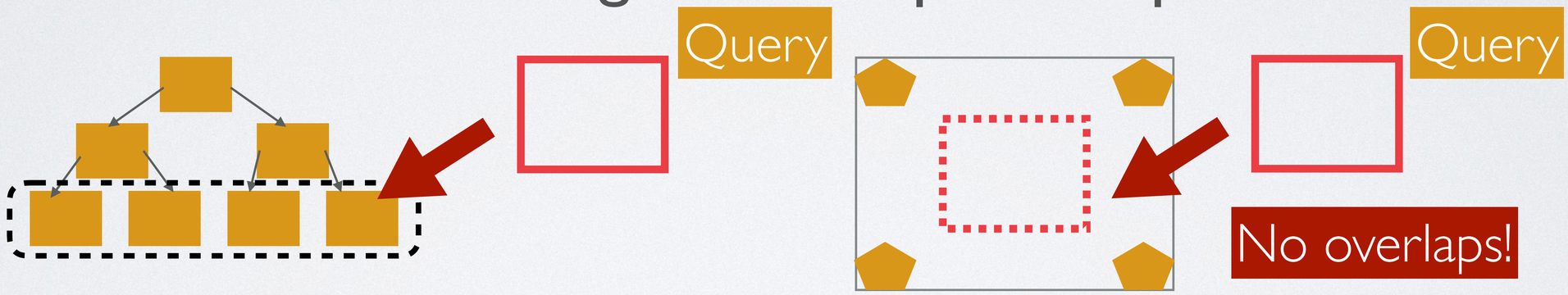
Spatial Indexing

- Local indexing
  - On each RDD partition
  - R-Tree, Quad-Tree,…
  - Has entries for individual records
  - Queries that use spatial index requires a refinement phase based on the real shapes of objects



Spatial Indexing

- Partition range index (Spatial Hippo, spatial bloom filter)
  - Global index only indexes bounding boxes not internal content
  - Queries sometimes still go to false positive partitions

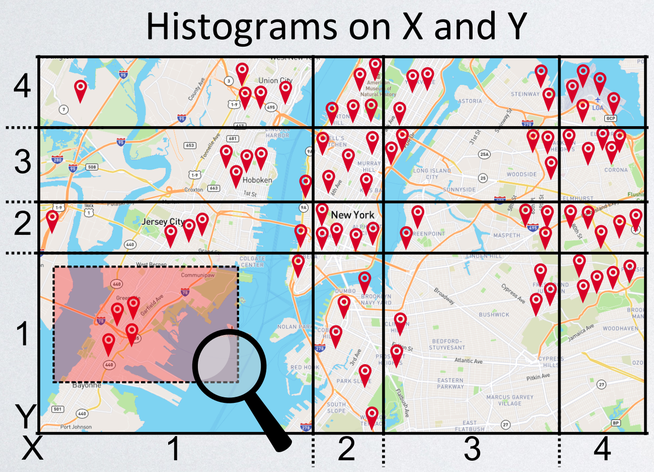


Tang, Mingjie, Yongyang Yu, Qutaibah M. Malluhi, Mourad Ouzzani, and Walid G. Aref. "Locationspark: A distributed in-memory data management system for big spatial data." *PVLDB 2016*

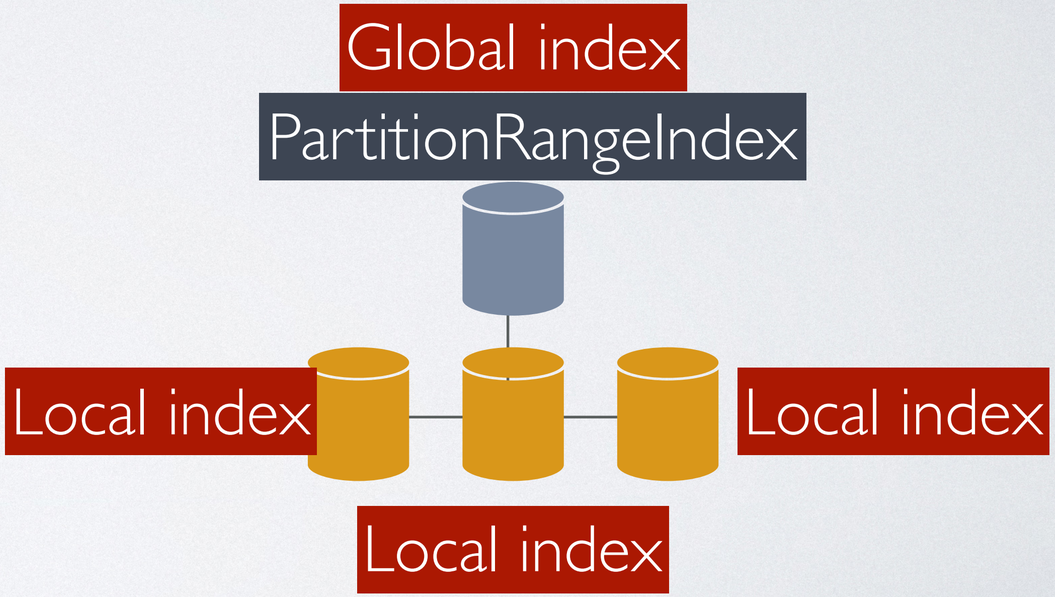
Yu, Jia, and Mohamed Sarwat. "Indexing the Pickup and Drop-Off Locations of NYC Taxi Trips in PostgreSQL–Lessons from the Road." In *International Symposium on Spatial and Temporal Databases*, pp. 145-162. Springer, Cham, 2017.

Spatial Indexing

- Partition range index (Spatial Hippo, spatial bloom filter)
  - On master machine, a concise histogram for each partition
  - Reduce false positive partitions

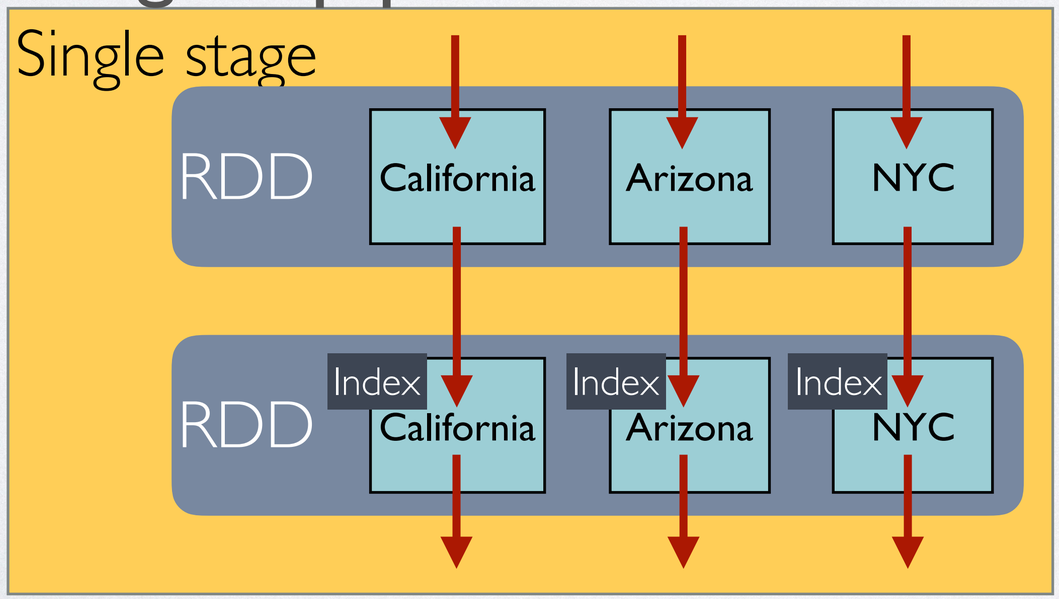


PartitionID	Bucket(1,1)	Bucket(1,2)	Bucket(1,3)	…
0	1	0	0	
1	0	1	0	
2	0	1	0	
3	0	0	1	



Spatial Indexing

- DAG and data shuffle: 1 RDD transformations
  - Global indexing: done with the spatial data partitioning (including partition range index)
  - Local indexing: Map per Partition, Narrow dependency

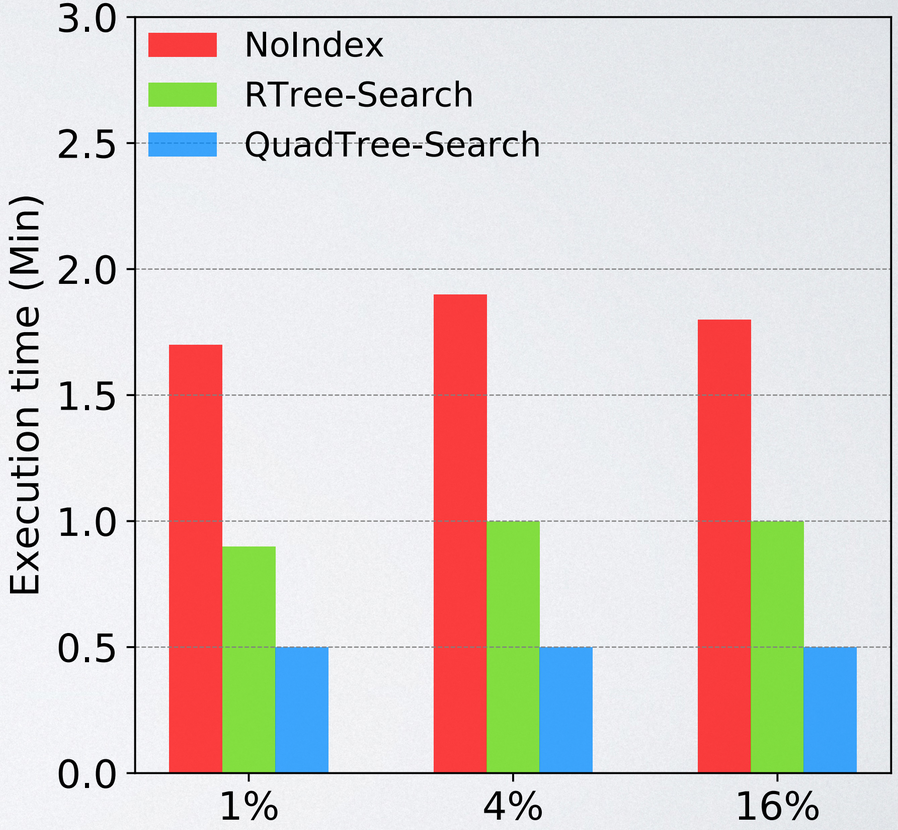


Narrow dependency

No shuffle

Spatial Indexing

- Performance on different local indexes
  - Measured using spatial range query
  - Range area from 1% to 16%
  - OSMobject: 263 million polygons
- Cluster settings: Four workers, one master, 192 cores, 400 GB Memory



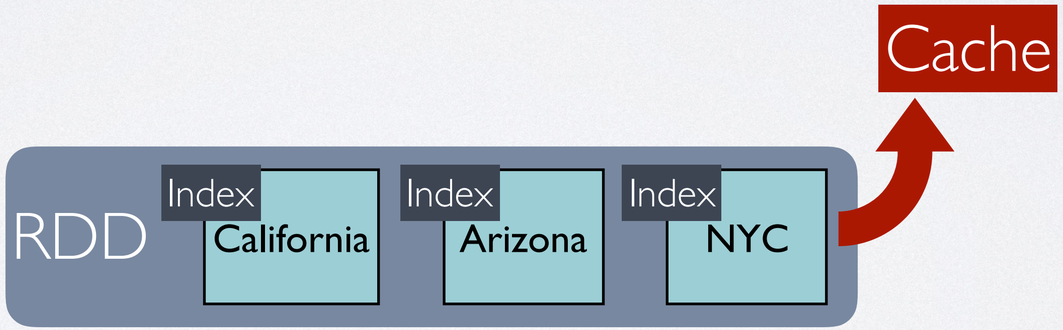
Manage Spatial Data

+ APACHE Spark™

- Spatial data partitioning
- Spatial indexing
- **Spatial queries**
- Optimization
- Language, spatial object support

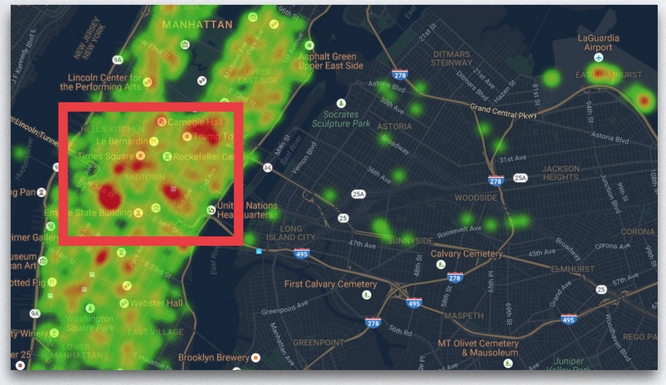
Spatial Queries

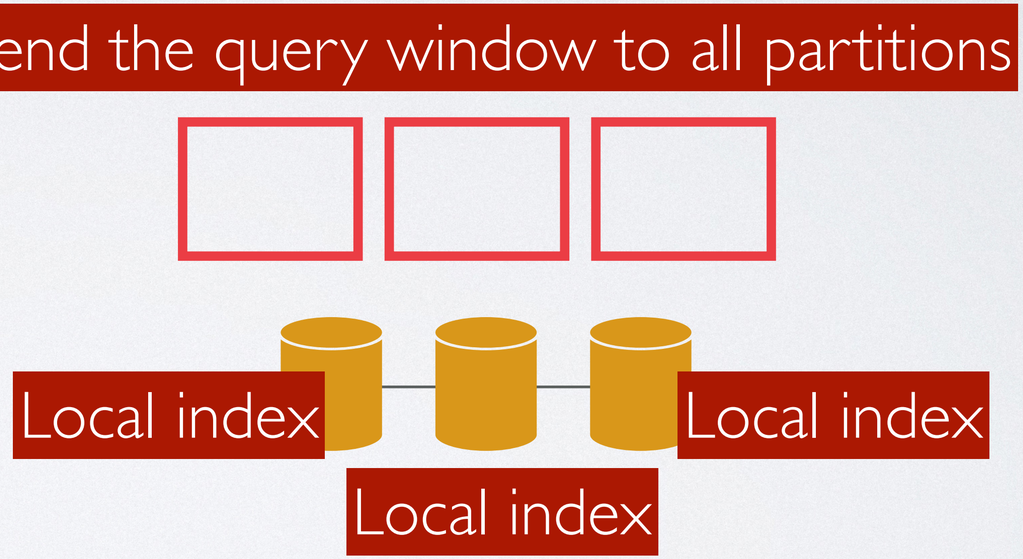
- Spatial queries should utilize spatial partitioning and spatial indexing
- Cache the indexed spatial partitioned RDD
- The cached RDD cannot be updated. It is expected to be used many times



Spatial Queries

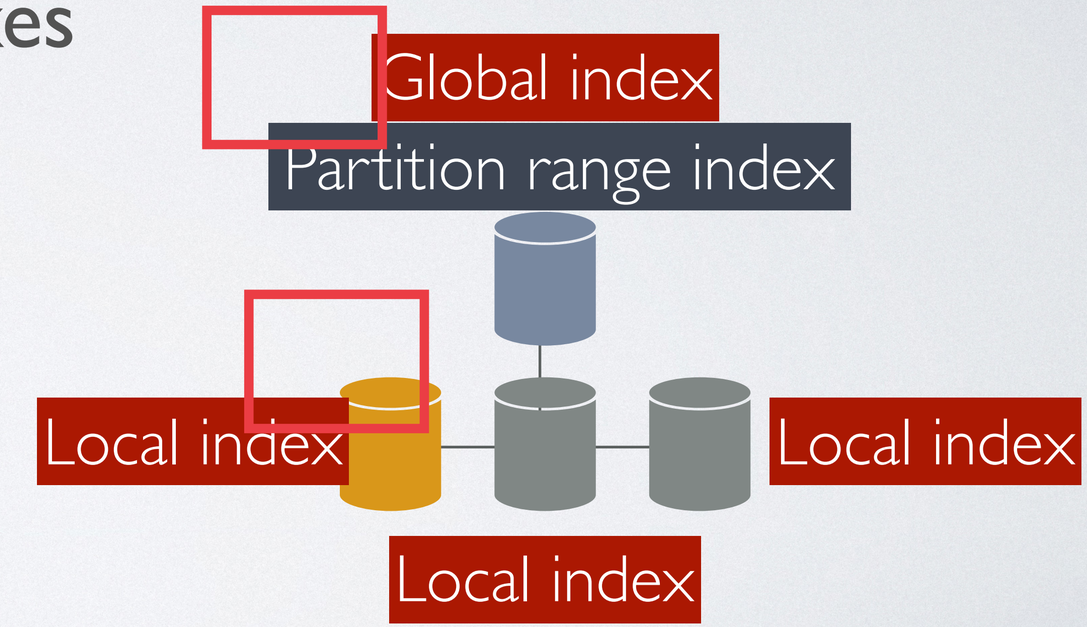
- Spatial range query: a straightforward way





Spatial Range Query

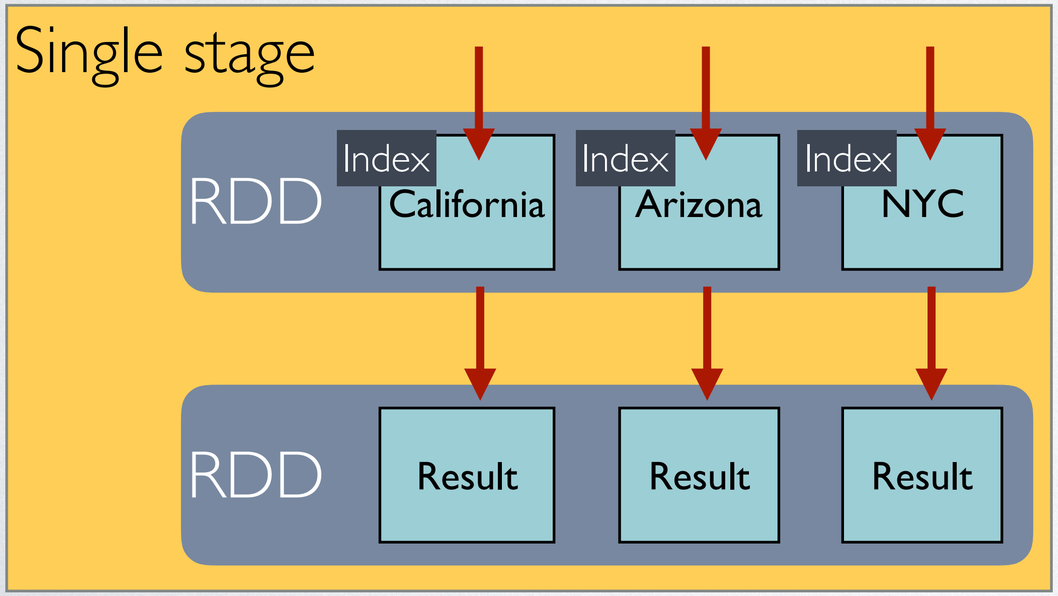
- Prune partitions based on the global index, on master machine
- Prune partitions using partition range index, on master machine
- Go to partitions and check local indexes
- API: rdd.PartitionPruningRDD



Tang, Mingjie, Yongyang Yu, Qutaibah M. Malluhi, Mourad Ouzzani, and Walid G. Aref. "Locationspark: A distributed in-memory data management system for big spatial data." *PVLDB 2016*

Spatial Range Query

- DAG and data shuffle: 1 RDD transformations
  - Checking global indexing -> on master machine
  - Checking local indexing -> a MapPartition operation, no shuffle

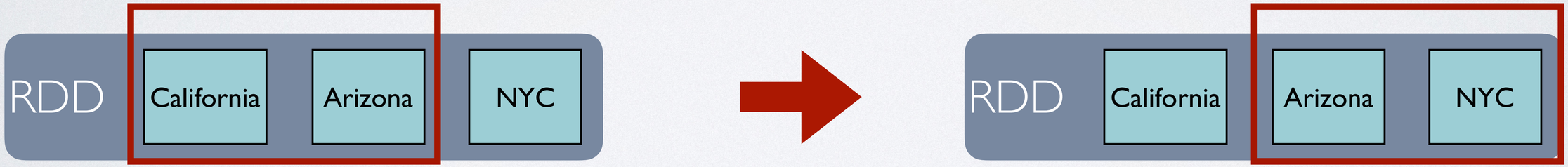


Narrow dependency

No shuffle

Load Spatial Data in Batches

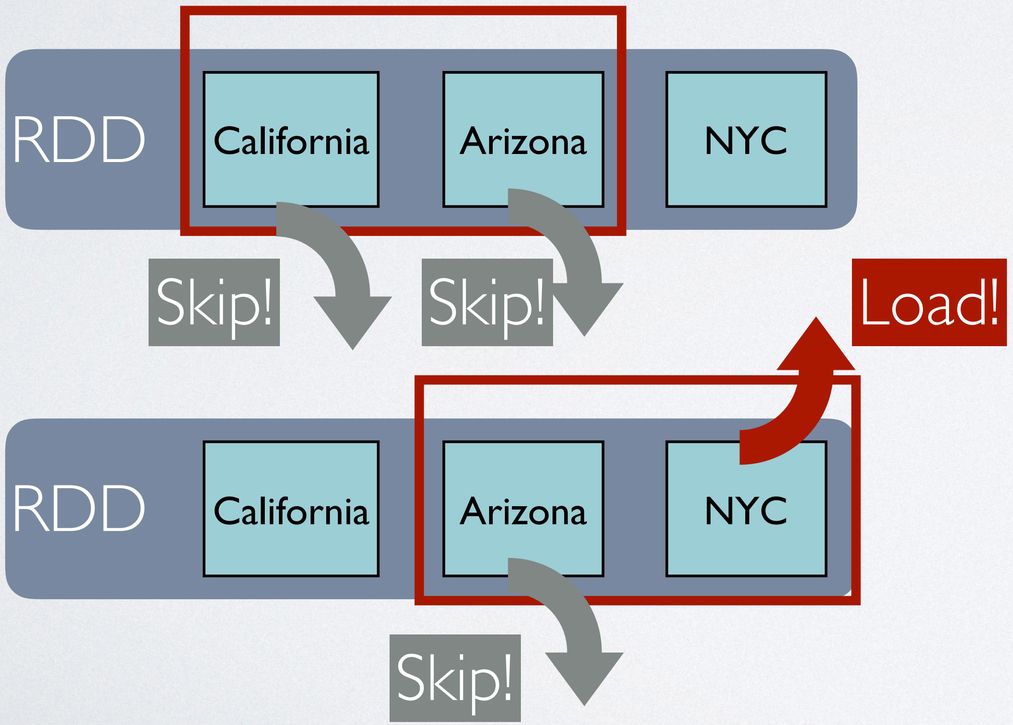
- You are generally tight on memory budget
- Spark needs a great deal of memory ✗
- Use a sliding window to load spatial data in batches
- Sliding window: size = num of partitions, decide it based on mem



Baig, Furqan, Hoang Vo, Tahsin Kurc, Joel Saltz, and Fusheng Wang. "Sparkgis: Resource aware efficient in-memory spatial query processing." *SIGSPATIAL 2017*

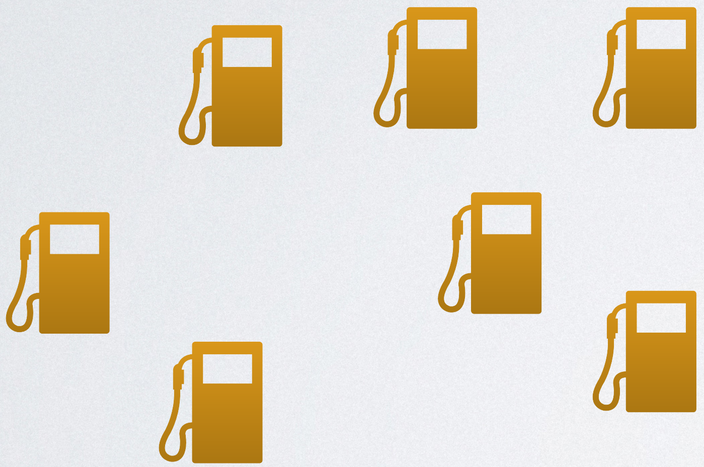
Load Spatial Data in Batches

- Use a sliding window to load spatial data in batches
- Load a partition only if its bounding box overlaps query predicate

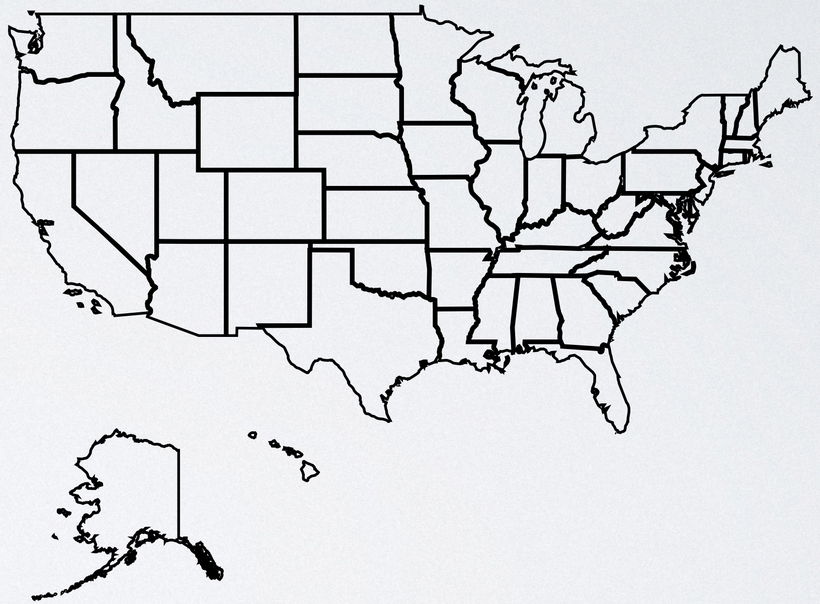


Spatial Join Query

- A set of objects (gas station), a set of polygons (state boundaries)
- Find gas stations in each state



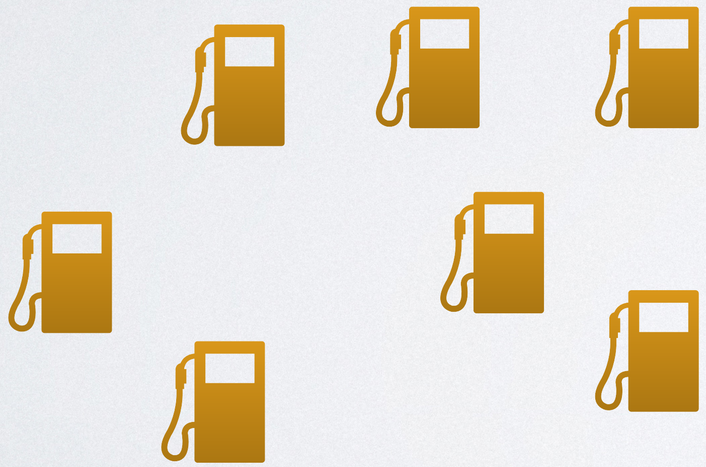
Gas station



State boundary

Spatial Join Query

- Distance join query, similar to spatial join
- Find gas stations within 1mile distance of each grocery
- Add distance buffer to each grocery = spatial join query



Gas station

Grocery

Spatial Join Query

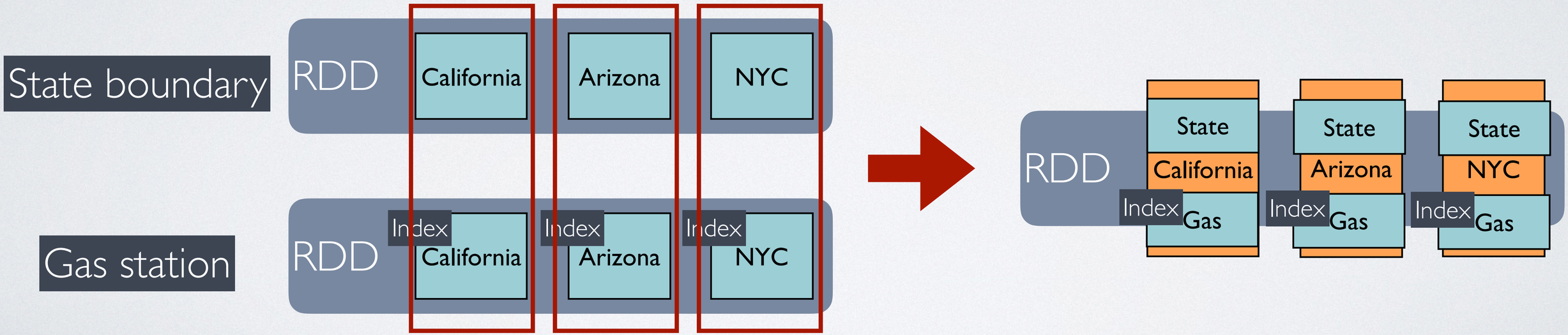
- Algorithm
  - ZipPartition
  - Local index-nested loop join
  - Local de-duplication using the reference point

Yu, Jia, Zongsi Zhang, and Mohamed Sarwat. "Spatial data management in apache spark: the GeoSpark perspective and beyond." *GeoInformatica* (2018): 1-42.

Dittrich, J-P., and Bernhard Seeger. "Data redundancy and duplicate detection in spatial join processing." In *ICDE*, 2000.

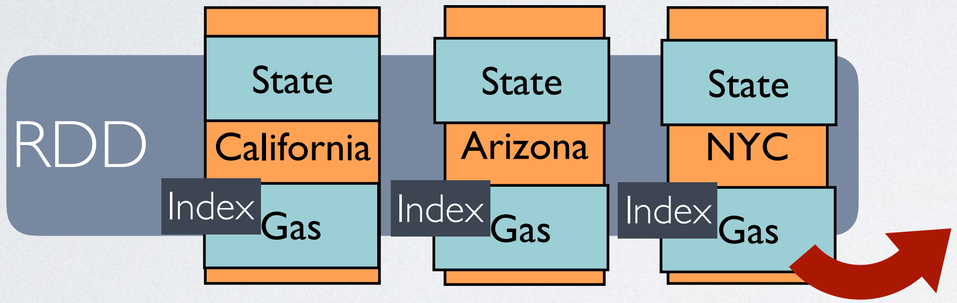
Spatial Join Query Algorithm

- ZipPartition
  - Both RDDs should be partition by the same way
  - One can have local index



Spatial Join Query Algorithm

- Local index-nested loop join

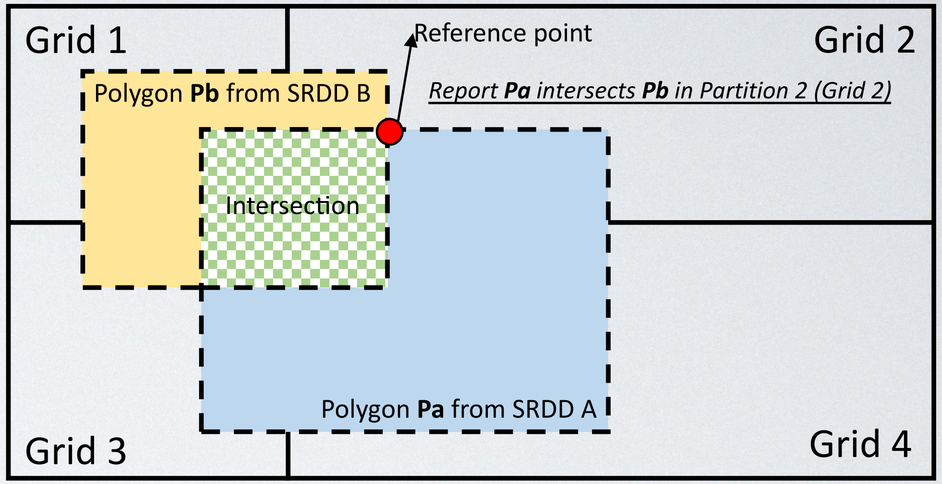


ForEach (state) in local states
Search **local index** on gas station

- Local de-duplication using the reference point
  - Spatial partitioning introduces duplicates
  - Need to remove them without incurring data shuffle!

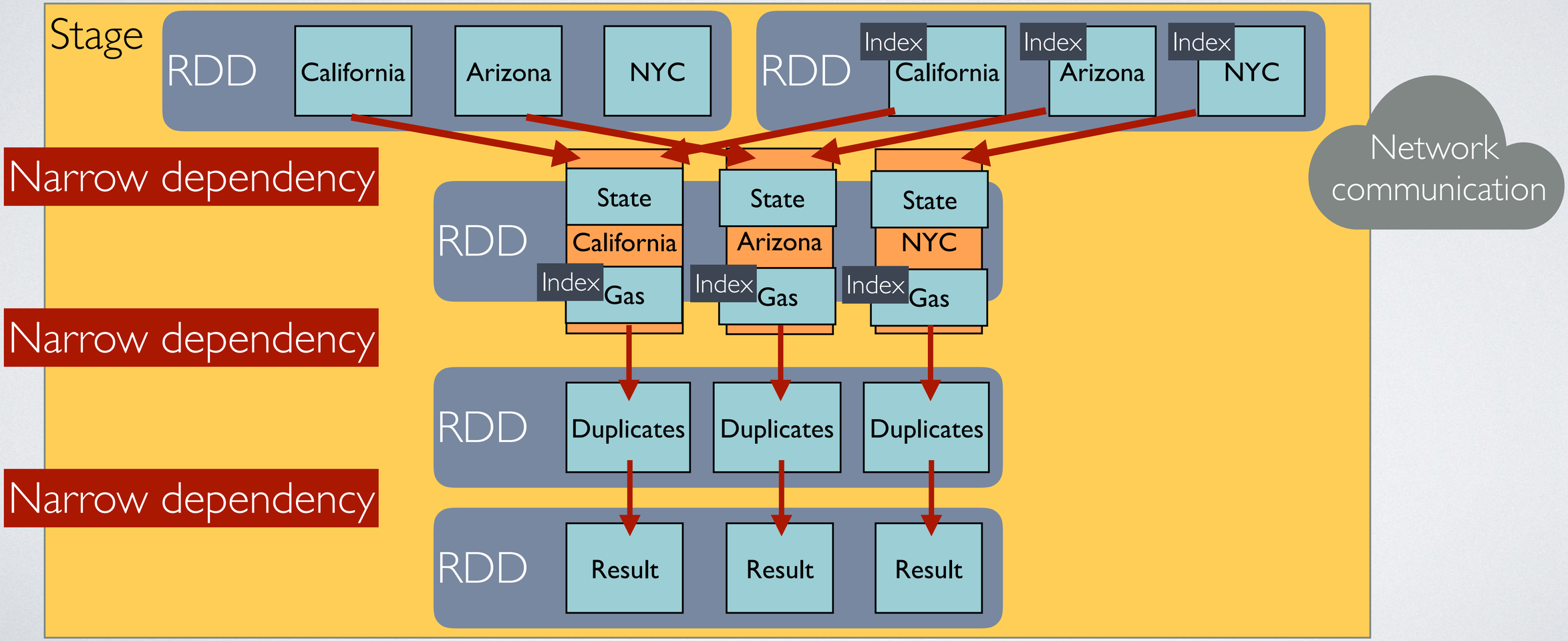
Spatial Join Query Algorithm

- Reference point
  - Query results with duplicates
    - (Pa, Pb) (Pa, Pb) (Pa, Pb) (Pa, Pb)
  - Compute the intersection of Pa and Pb
  - Take Reference Point(maxX, maxY) of intersection
  - Report (Pa, Pb) in a partition only if reference point is within the boundary of this partition



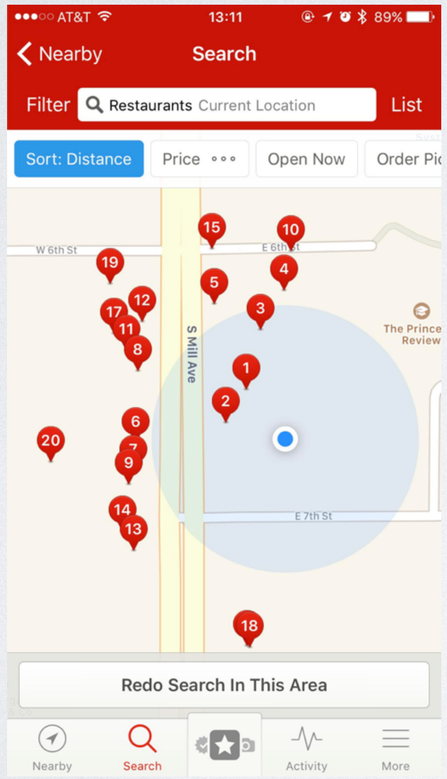
Spatial Join Query

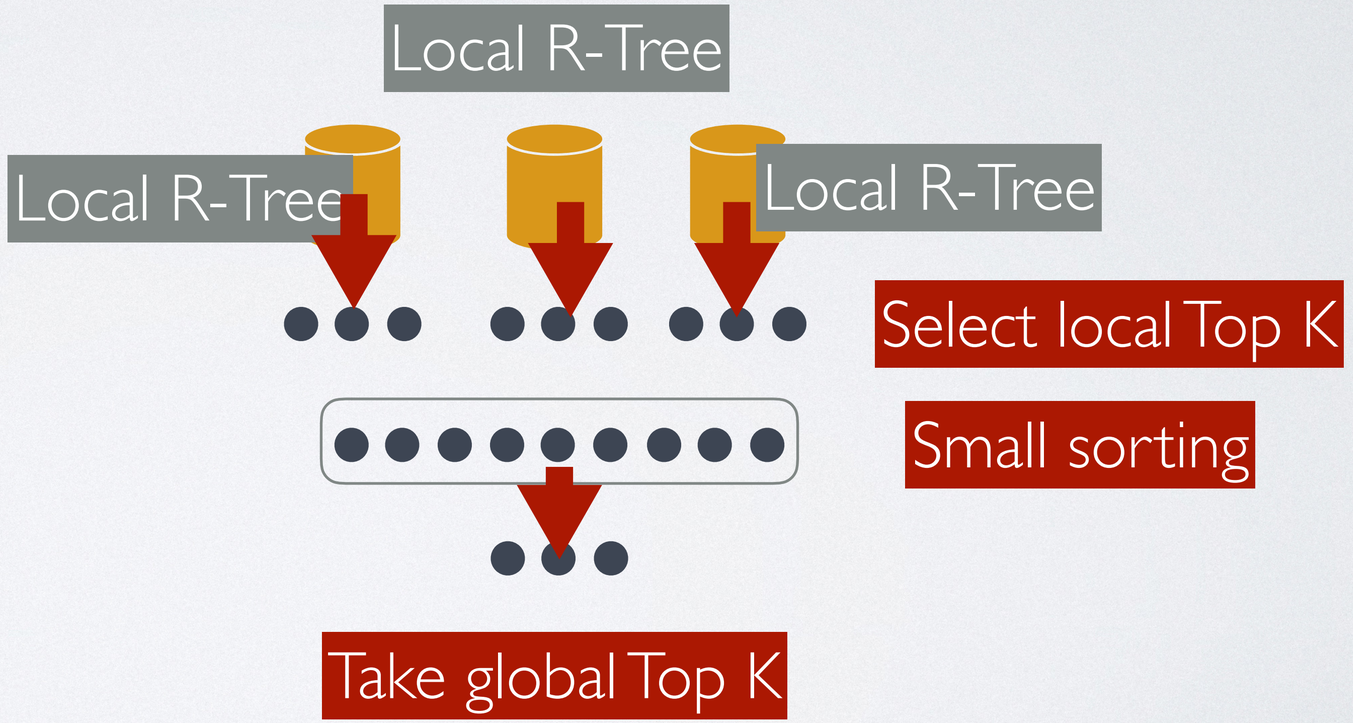
- DAG and data shuffle: 3 RDD transformations



Spatial K-Nearest Neighbor

- Selection + Sorting phase

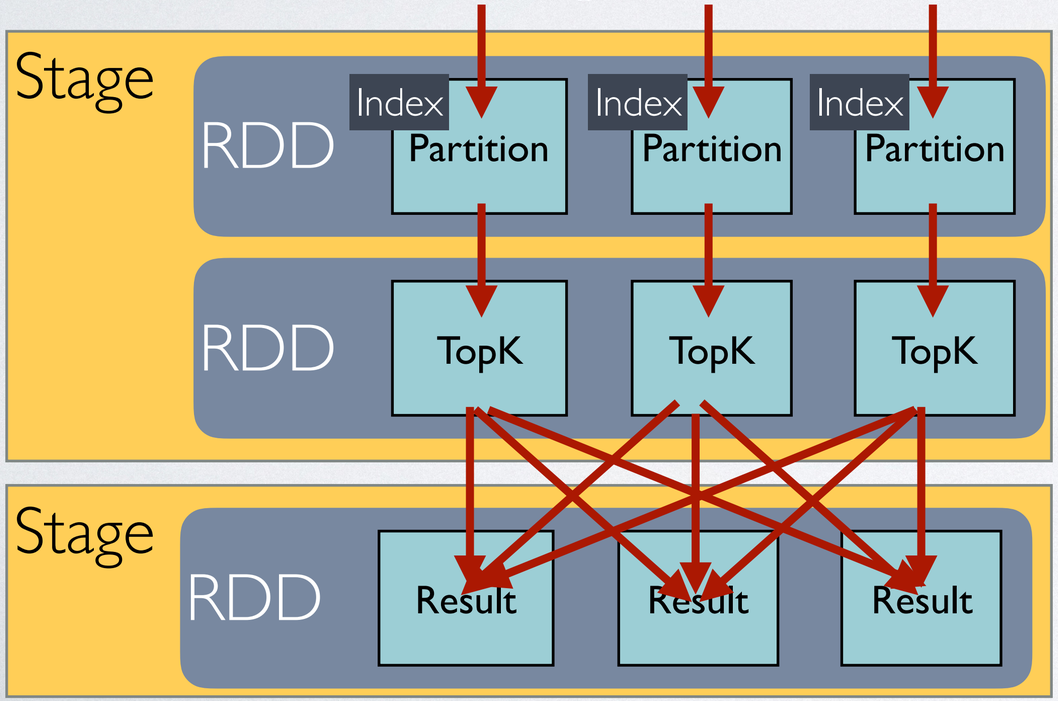




Yu, Jia, Zongsi Zhang, and Mohamed Sarwat. "Spatial data management in apache spark: the GeoSpark perspective and beyond." *GeoInformatica* (2018): 1-42.

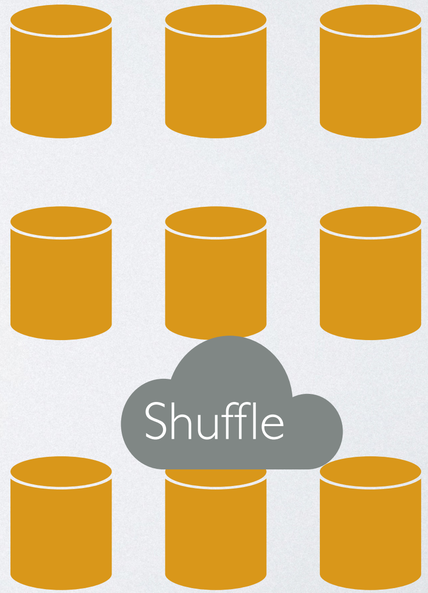
Spatial K-Nearest Neighbor

- DAG and data shuffle: 2 RDD transformations
  - Local Top K selection: Map operation, Narrow dependency
  - Global sorting: Wide dependency



Narrow dependency

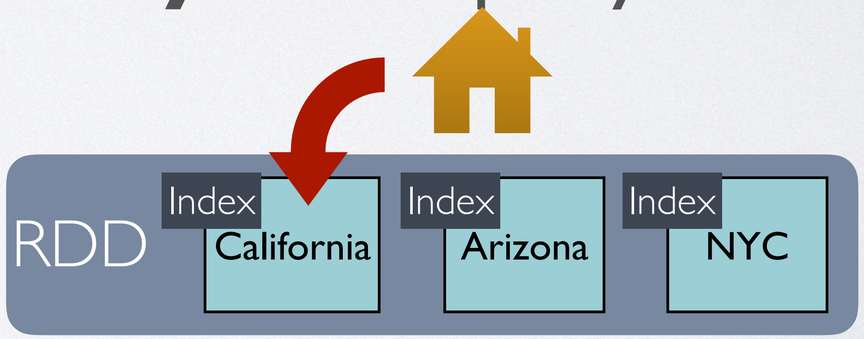
Wide dependency



Spatial K-Nearest Neighbor

- Why not use global index to prune partitions?
  - Query accuracy is not guaranteed
  - KNN might be in other partitions
- The correct spatial partitioning for KNN should have a K-element buffer and **repartition** RDD for **every** KNN query

Too expensive



Some results might be in Arizona partition!

Spatial KNN Join

- Find the nearest 3 gas stations for each grocery

Gas station

Grocery

Spatial KNN Join

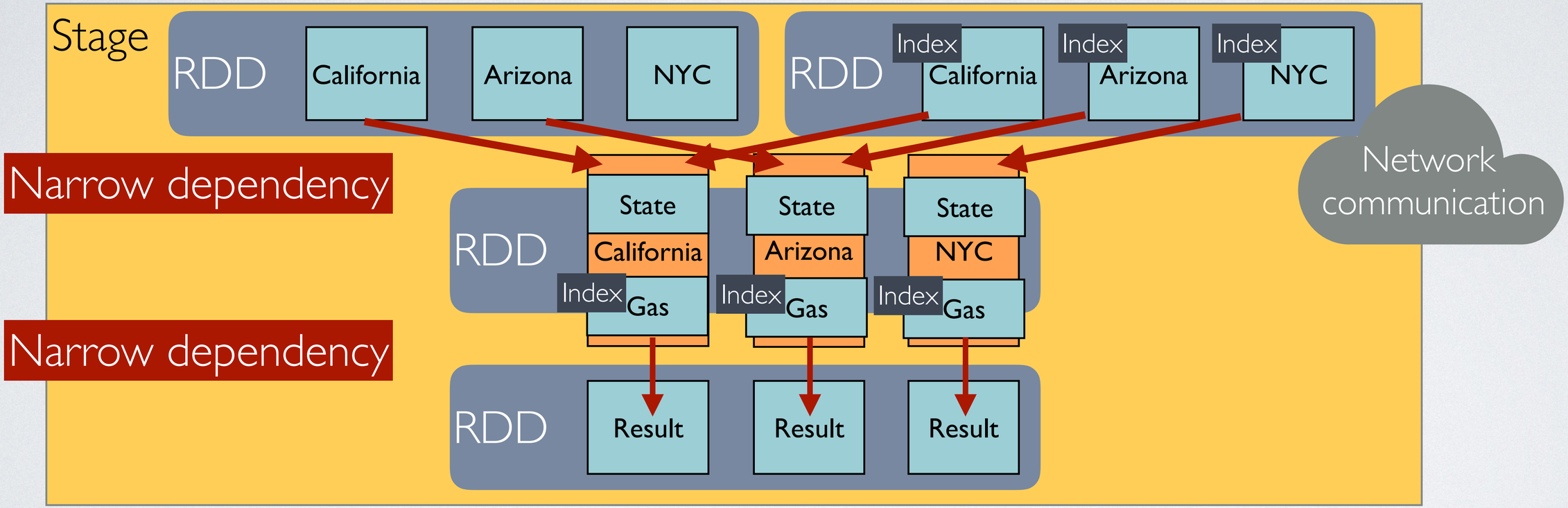
- Spatial partitioning: a distance buffer for each partition such that each query point can find its KNN in one RDD partition.
- Local KNN

Chatzimilioudis, Georgios, Constantinos Costa, Demetrios Zeinalipour-Yazti, Wang-Chien Lee, and Evaggelia Pitoura. "Distributed in-memory processing of all k nearest neighbor queries." *IEEE TKDE* 2016

Xie, Dong, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. "Simba: Efficient in-memory spatial analytics." In *SIGMOD*, 2016.

Spatial KNN Join

- DAG and data shuffle: 2 RDD transformations



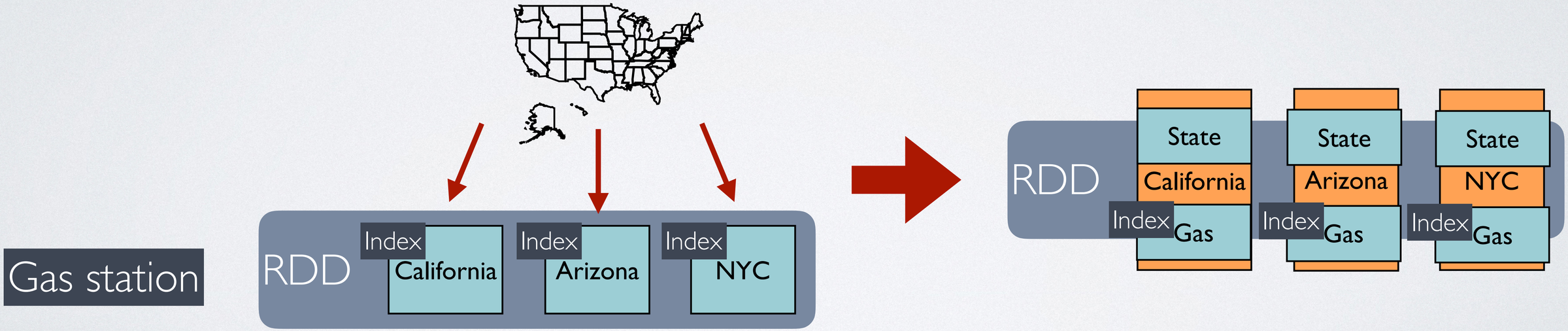
Manage Spatial Data

+ APACHE Spark™

- Spatial data partitioning
- Spatial indexing
- Spatial queries
- **Optimization**
- Language, spatial object support

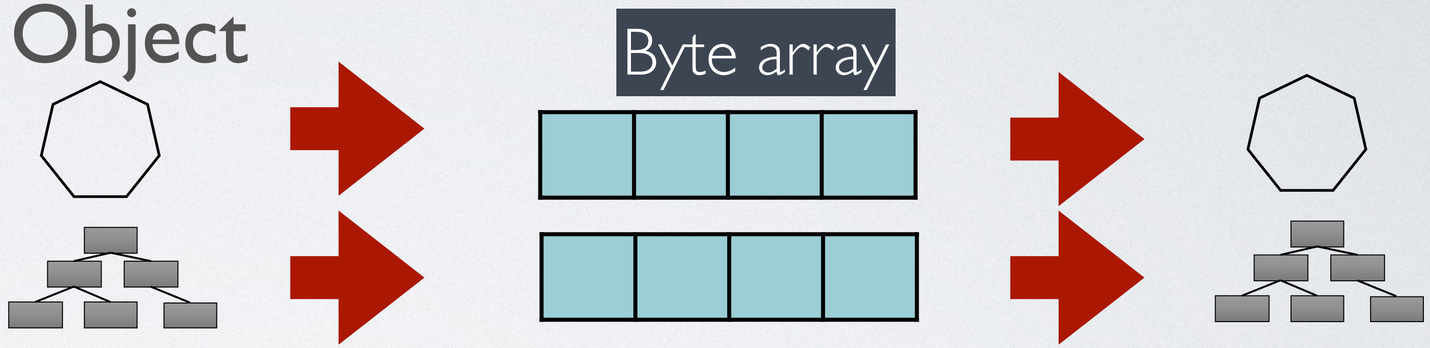
Optimization

- Query optimization
  - Distributed spatial join VS broad-cast spatial join
    - One side of spatial join is smaller, send it to all RDD partitions

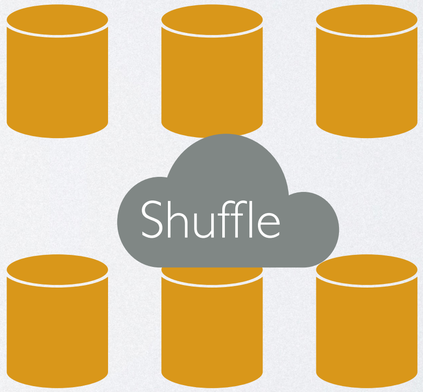


Optimization
Custom Serializer

- What is a serializer?
  - Object -> byte array -> Object



- When is a serializer used?
  - Cache RDD into memory
  - Shuffle objects across the cluster



Optimization
Custom Serializer

- Why do we need a custom serializer for spatial objects?
  - Spatial objects are **very complex, tons of coordinates**
  - Spark default Java and Kryo serializer are not efficient
  - Size according to GeoSpark experiment
    - 3 times smaller than Spark default size
    - 20 times faster serialization
    - 5 times faster deserialization

https://github.com/DataSystemsLab/GeoSpark/tree/master/core/src/main/java/org/datasyslab/geospark/geometryObjects

Optimization
Custom Serializer

- How to write a spatial object serializer?
  - Define a rule to serialize heterogeneous spatial types into a byte array. For example, borrow the definition of Shapefile or WKB
- How to write a spatial index serializer?
  - Use a regular tree traversal algorithm to traverse the tree
  - Note the child node size because an index is not a full tree

Optimization
Custom Serializer

- How to add a serializer to Spark?
  - Write a register via Kryo
  - Register it when creating Spark session

```
var sparkSession = SparkSession.builder()
.appName("myAppName")
// Enable GeoSpark custom Kryo serializer
.config("spark.serializer", classOf[KryoSerializer].getName)
.config("spark.kryo.registrator", classOf[GeoSparkKryoRegistrator].getName)
.getOrCreate()
```

https://github.com/DataSystemsLab/GeoSpark/blob/master/core/src/main/java/org/datasyslab/geospark/serde/GeoSparkKryoRegistrator.java

Manage Spatial Data

+

- Spatial data partitioning
- Spatial indexing
- Spatial queries
- Optimization
- Language, spatial object support

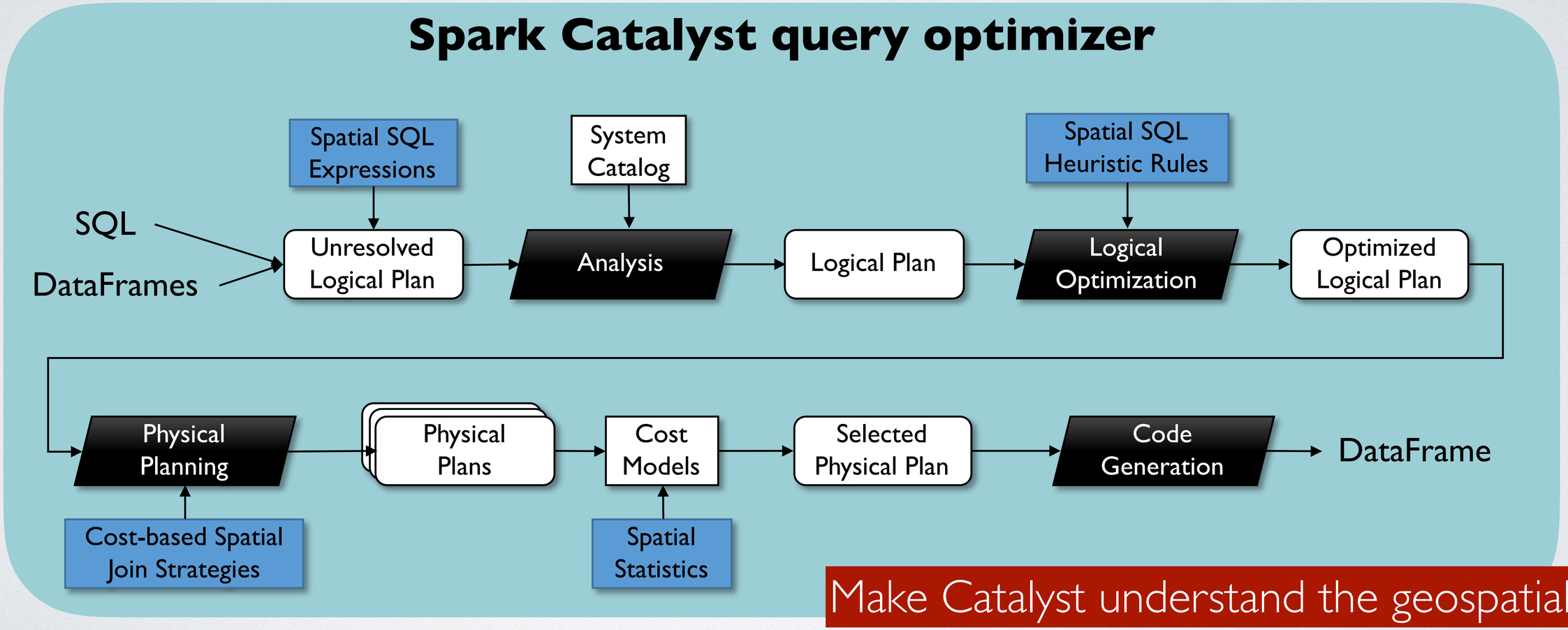
Language

- Implement the system in what language?
  - Scala
    - Spark is written in Scala
    - Functional programming by nature
  - Java
    - No learning curve, Scala/Java functions can call each other
    - Cannot modify Spark kernel
    - Cannot add UserDefinedType and query optimization
  - Python
    - Python code connect to Spark via Py4j
    - Needs Python spatial object handler

Spark Interface

- Spark interface
  - RDD: easy to customize, hard to use
  - DataFrame: easy to use , hard to customize
    - Spatial SQL
    - User Defined Type
    - Indexing and spatial partitioning
    - Optimized join strategy

Integrate With Dataframe



Make Catalyst understand the geospatial!

Integrate With Dataframe Spatial SQL

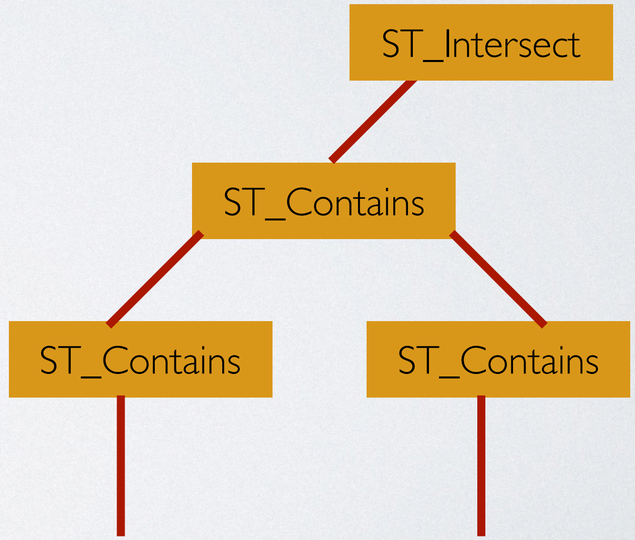
Arizona State University

- Spatial SQL: SQL-MM3, OGC Simple Feature Access
  - SQL-MM3: PostGIS, GeoSpark, GeoMesa…
    - *ST_Contains, ST_Within*
  - OGC Simple Feature Access
    - *Contains, Within*
  - Compatible with each other in most cases
  - Implement these functions in Spark expression (not UDF)

```
SELECT superhero.name
FROM city, superhero
WHERE ST_Contains(city.geom, superhero.geom)
AND city.name = 'Gotham';
```

Integrate With Dataframe Spatial SQL

- Spark expression, the way Spark writes its own functions
  - Unary, binary, ternary
  - Each AST (Abstract Syntax Tree) node is a Spark expression
  - Allow the following features
    - Code generation
    - Output data type
    - Fuse into the Catalyst optimizer



https://github.com/DataSystemsLab/GeoSpark/tree/master/sql/src/main/scala/org/apache/spark/sql/geosparksql/expressions

Integrate With Dataframe User Defined Type

- Each spatial object and index must be a UDT in Dataframe
- Spark provides a developer API
- The spatial object UDT must be based on a primitive type: Array
- Must provide the serialization method

UDT code snippet from GeoSpark

```
private[sql] class GeometryUDT extends UserDefinedType[Geometry] {
  override def sqlType: DataType = ArrayType(ByteType, containsNull = false)

  override def userClass: Class[Geometry] = classOf[Geometry]

  override def serialize(obj: Geometry): GenericArrayData = {
    new GenericArrayData(GeometrySerializer.serialize(obj))
  }

  override def deserialize(datum: Any): Geometry = {
    datum match {
      case values: ArrayData => {
        return GeometrySerializer.deserialize(values)
      }
    }
  }

  case object GeometryUDT extends GeometryUDT

}
```

Integrate With Dataframe Indexing and Spatial Partitioning

Arizona State University

- Indexing
  - Each local index is a big UDT. Each partition has one UDT (row).
  - No way to plug the global index to Catalyst physical plan
- Spatial partitioning
  - Cannot be done via regular DataFrame API
  - Use DataFrame's RDD API to do spatial partitioning

Integrate With Dataframe Optimized Spatial Join Strategy

- Inject spatial join query
  - Overwrite Spark strategy (physical plan)
  - Use pattern-matching to capture spatial join pattern

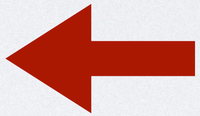
Join pattern-matching snippet from GeoSpark

```
def apply(plan: LogicalPlan): Seq[SparkPlan] = plan match {

  // ST_Contains(a, b) - a contains b
  case Join(left, right, Inner, Some(ST_Contains(Seq(leftShape, rightShape)))) =>
    planSpatialJoin(left, right, Seq(leftShape, rightShape), false)

  // ST_Intersects(a, b) - a intersects b
  case Join(left, right, Inner, Some(ST_Intersects(Seq(leftShape, rightShape)))) =>
    planSpatialJoin(left, right, Seq(leftShape, rightShape), true)

  // ST_WITHIN(a, b) - a is within b
  case Join(left, right, Inner, Some(ST_Within(Seq(leftShape, rightShape)))) =>
    planSpatialJoin(right, left, Seq(rightShape, leftShape), false)

   // ST_Overlaps(a, b) - a overlaps b
  case Join(left, right, Inner, Some(ST_Overlaps(Seq(leftShape, rightShape)))) =>
    planSpatialJoin(right, left, Seq(rightShape, leftShape), false)

  // ST_Touches(a, b) - a touches b
  case Join(left, right, Inner, Some(ST_Touches(Seq(leftShape, rightShape)))) =>
    planSpatialJoin(left, right, Seq(leftShape, rightShape), true)

  // ST_Distance(a, b) <= radius consider boundary intersection
  case Join(left, right, Inner, Some(LessThanOrEqual(ST_Distance(Seq(leftShape, rightShape)), radius))) =>
    planDistanceJoin(left, right, Seq(leftShape, rightShape), radius, true)
```

Integrate With Dataframe Optimized Spatial Join Strategy

- Register the new join strategy using its
  - sparkSession.experimental.extraStrategies = JoinQueryDetector

```
SELECT *
FROM polygondf, pointdf
WHERE ST_Contains(polygondf.polygonshape,pointdf.pointshape)
```



Captured join query plan

```
== Physical Plan ==
RangeJoin polygonshape#20: geometry, pointshape#43: geometry, false
:- Project [st_polygonfromenvelope(XXX) AS polygonshape#20]
:  +- *FileScan csv
+- Project [st_point(XXX) AS pointshape#43]
   +- *FileScan csv
```

Original join query plan

```
== Physical Plan ==
BroadcastJoin polygonshape#20: geometry, pointshape#43: geometry, false
:- Project [st_polygonfromenvelope(XXX), mypolygonid) AS polygonshape#20]
:  +- *FileScan csv
+- Project [st_point(XXX) AS pointshape#43]
   +- *FileScan csv
```

Outline

- Big geospatial data
- Manage spatial data
- Manage Spatio-Temporal Data
- Spatial Data Analytics in Spark

Spatial Visual Analytics

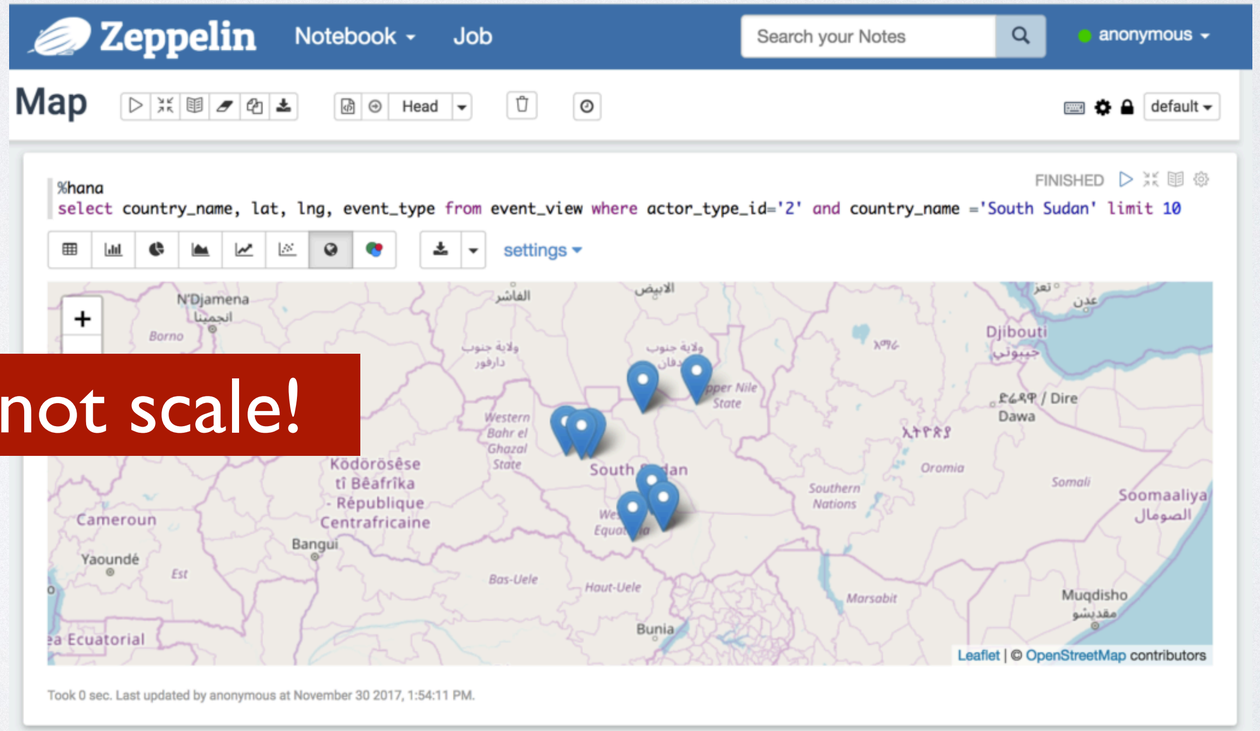
- Spatial visualization is important
- Existing tools can exhibit excellent visual effects but cannot scale

mapbox

Leaflet

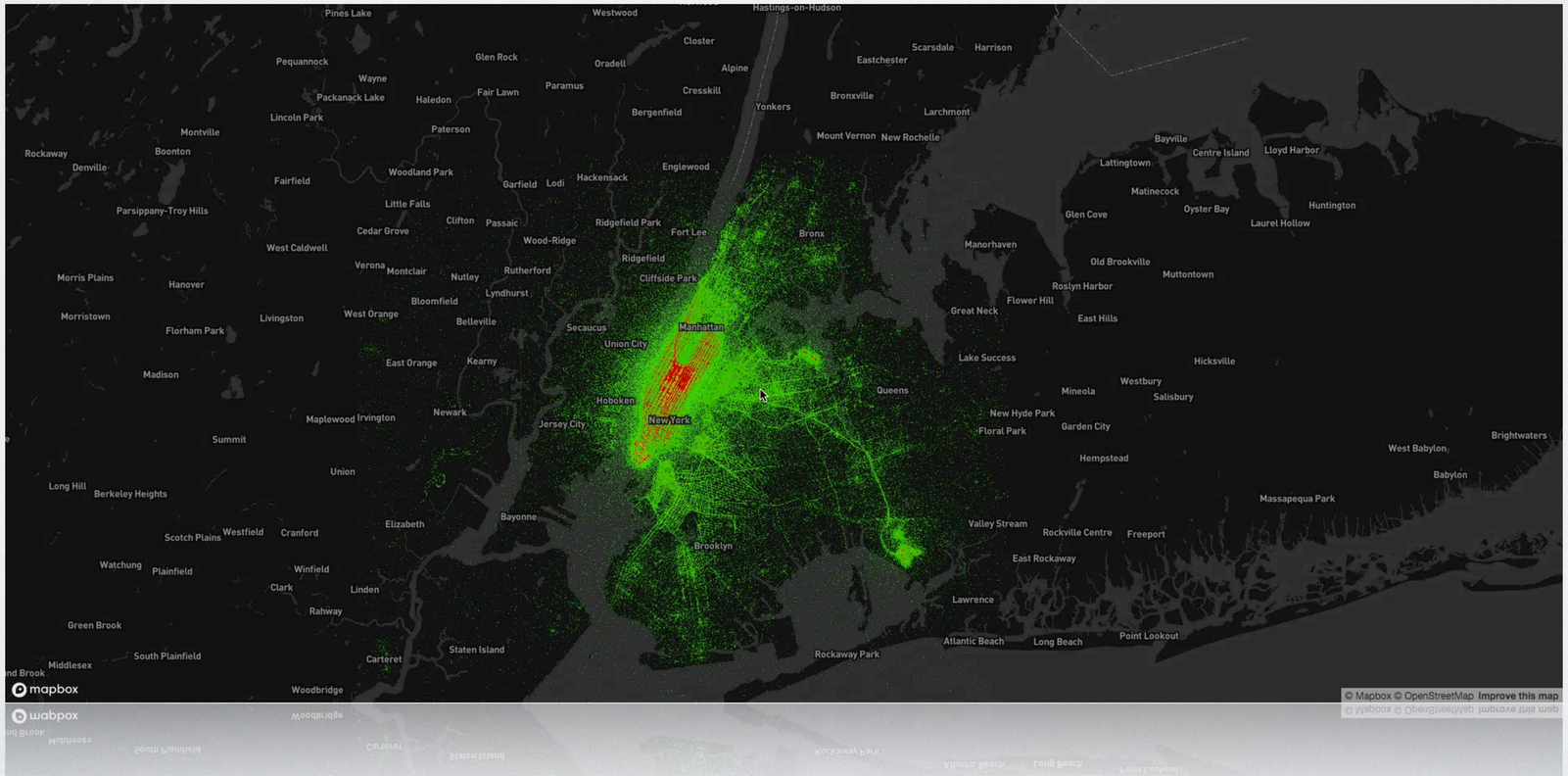
+ableau SOFTWARE

Cannot scale!



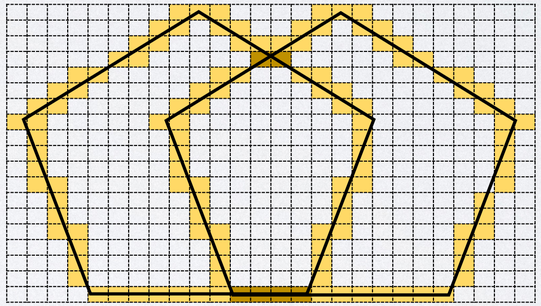
Spatial Visual Analytics

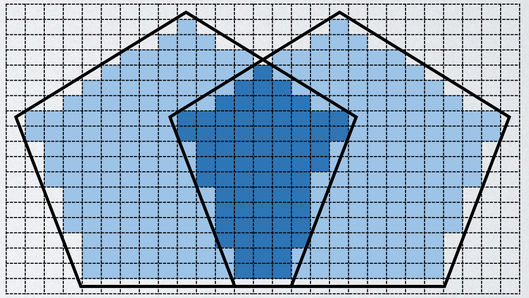
- Scalable visualization: visualize BILLION objects on Gigapixel map
- Customizable visualization: manipulate pixels at scale



Spatial Visual Analytics

- Rasterize vector shapes to pixels (with weights)
  - ST_Pixelize
  - Or, load from GeoTIFF/NetCDF/HDF: array-format spatial observations
- Aggregate pixels (with weights)
- Render color - ST_Render



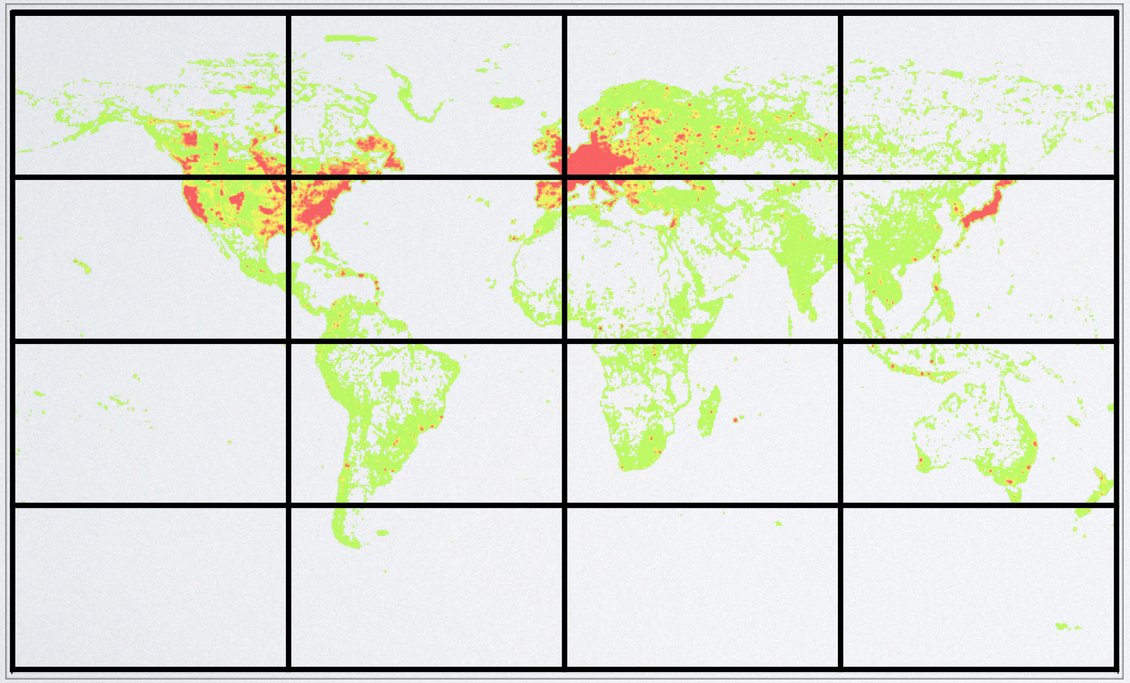


Geotrellis: https://geotrellis.io/

Yu, Jia, Zongsi Zhang, and Mohamed Sarwat. "GeoSparkViz: a scalable geospatial data visualization framework in the apache spark ecosystem." In *SSDBM*, 2018.

Pixel Array Data Partitioning

- Each partition is a map tile
- Each map tile is X*X pixel array

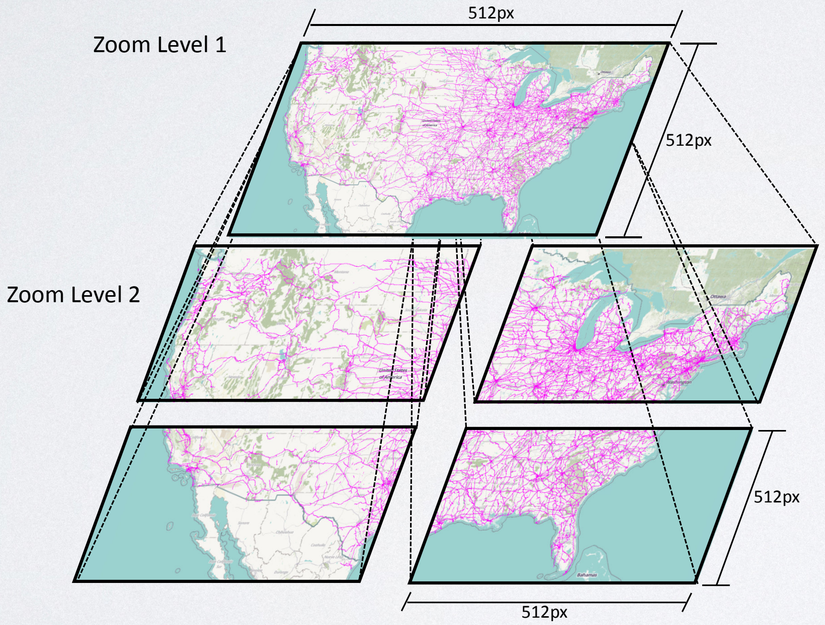


1	2	3	2	1
2	4	5	4	2
3	5	6	5	3
2	4	5	4	2
1	2	3	2	1

A single tile

RDD and Zoom Levels

- Zoom levels: each level consists of a set of map tiles
- Each RDD is a zoom level.



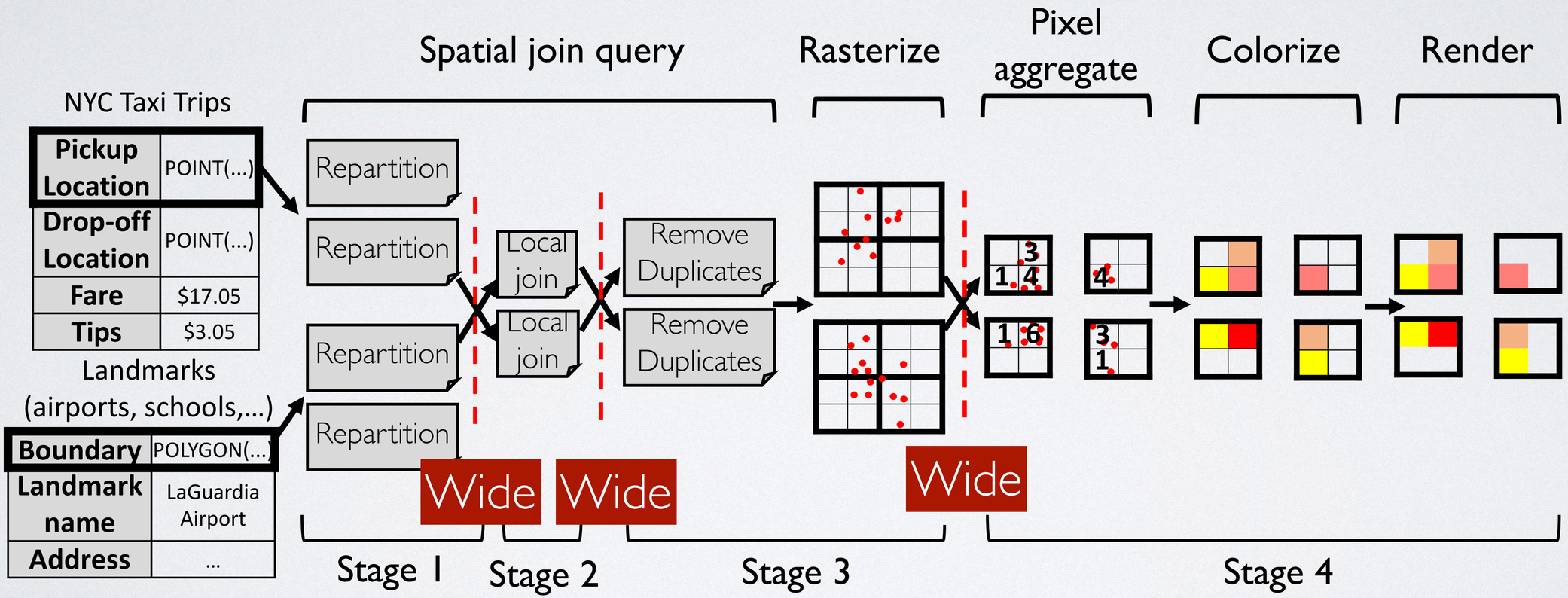
Zoom levels

Level	# Tiles
0	1
1	4
2	16
3	64
…	…

Tiles per level

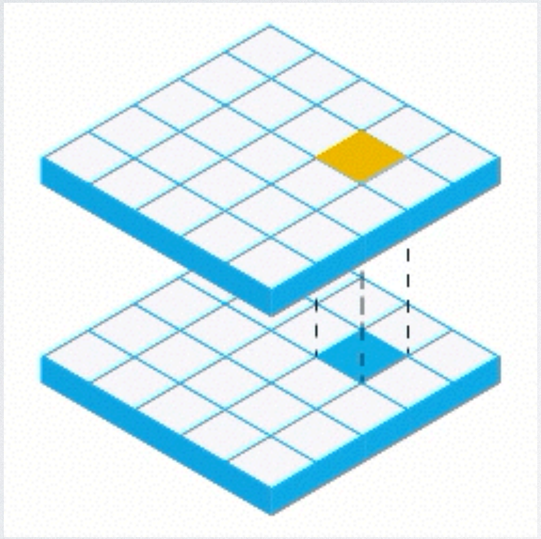
Map Visualization Pipeline

- Visualization pipeline and DAG stages

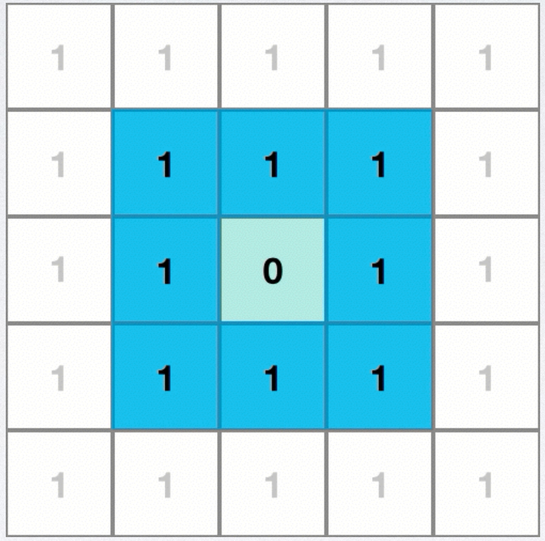


Manipulate Raster Array Data

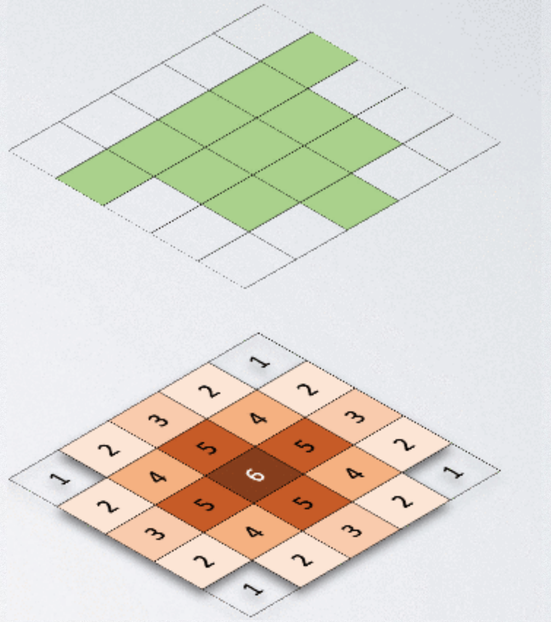
- Map algebra operations



Local operations



Focal operations

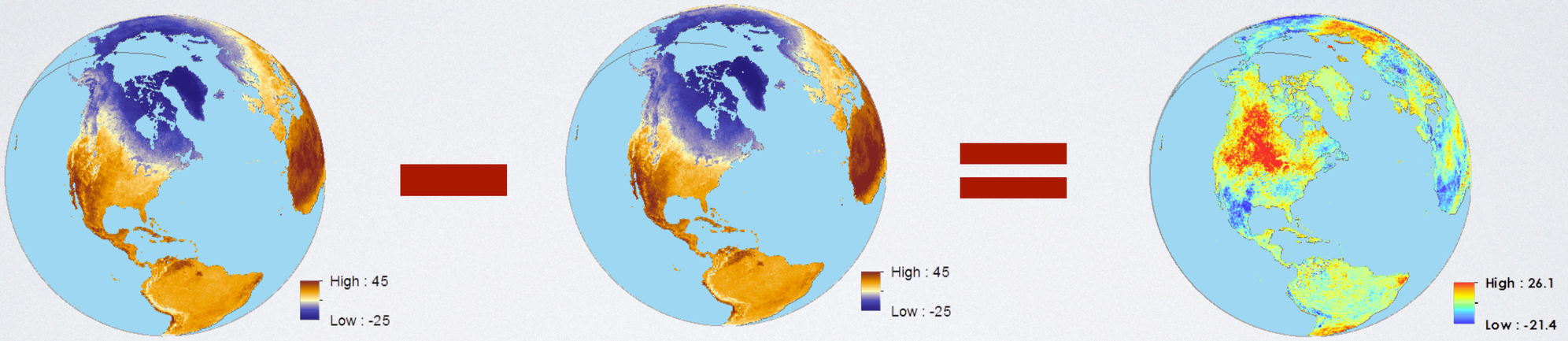


Zonal operations

Local and focal gif from Geotrellis: https://docs.geotrellis.io/en/latest/guide/core-concepts.html#map-algebra

A Map Algebra Example

- Local operation on temperature observations from NASA MODIS

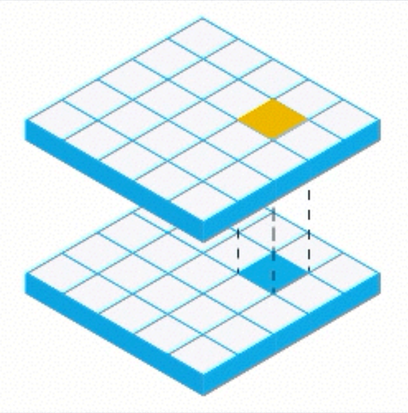


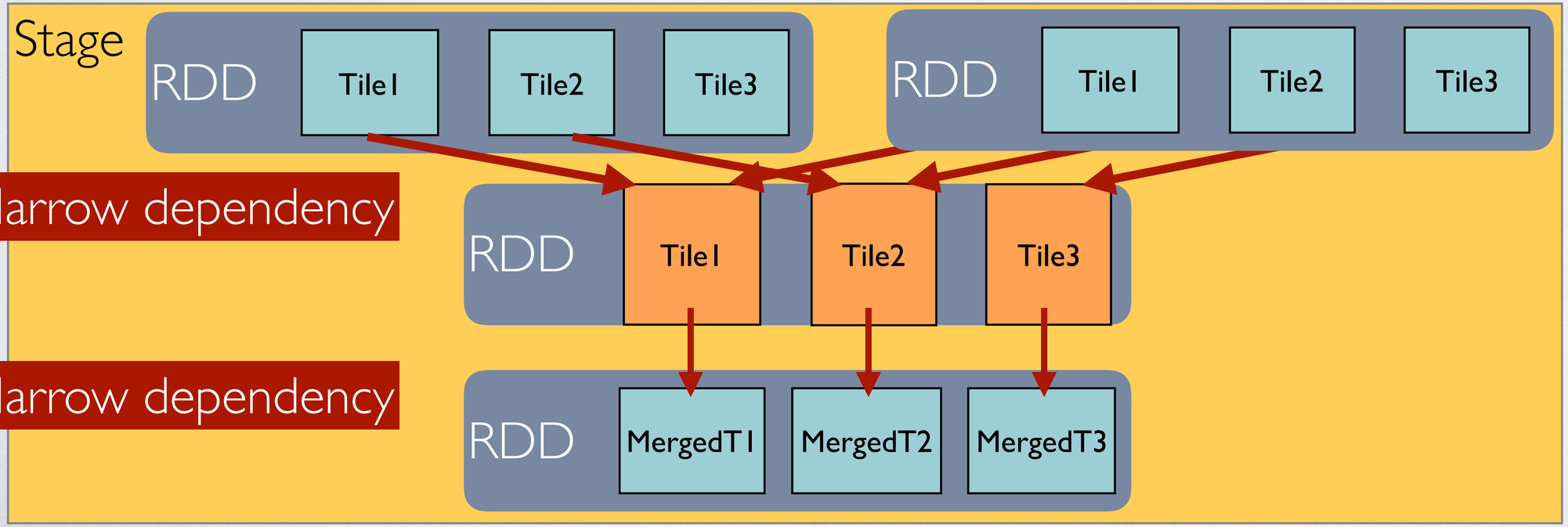
April 2015 | April 2014 | Climate change

Figures and examples from https://gisgeography.com/map-algebra-global-zonal-focal-local/

Local Operation

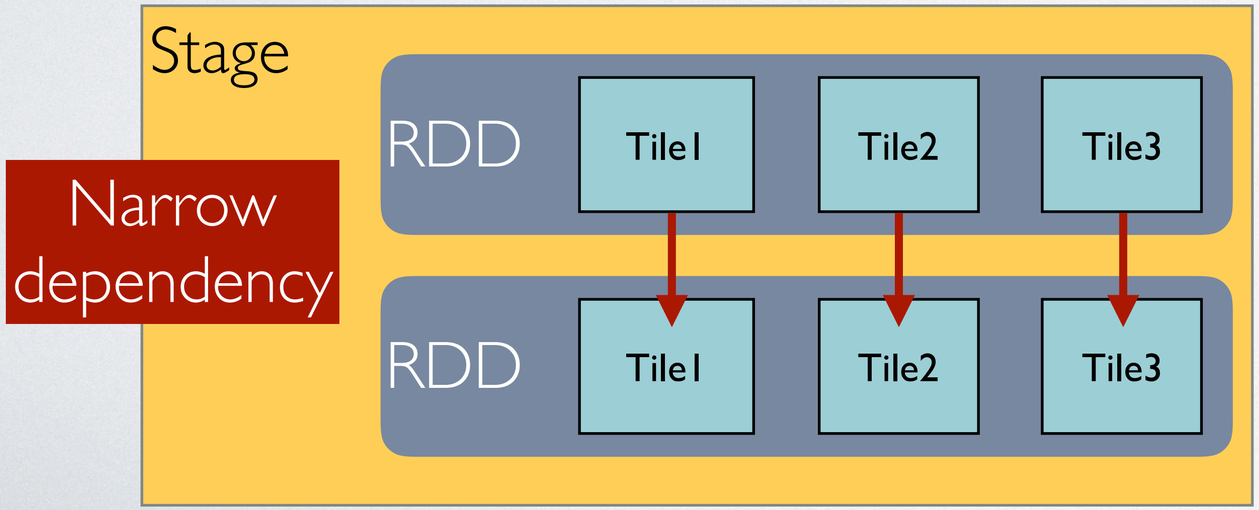
- Algorithm: two pixel RDDs A and B, partitioned in the same way
  - ZipPartitions: Zip A and B
  - MapPartition: Local pixel manipulation

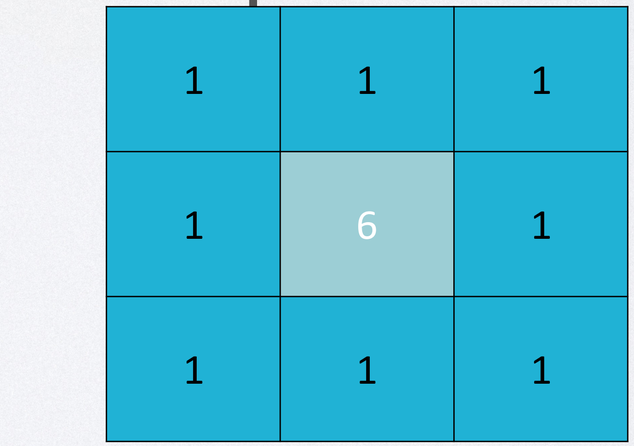




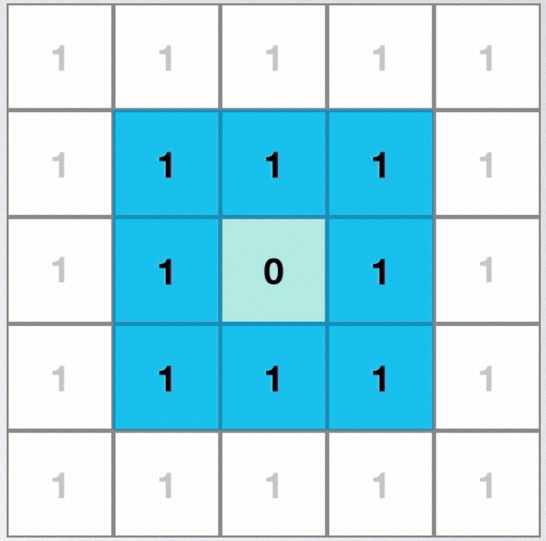
Focal Operation

- Algorithm: each pixel aggregates with its neighbors
  - Spatial partition buffered Pixels
    - Make sure each pixel can find its neighbors
  - MapPartition: Local aggregation in each partition



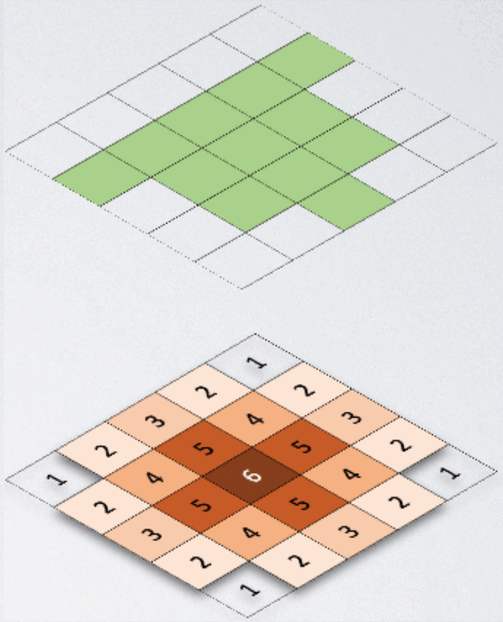


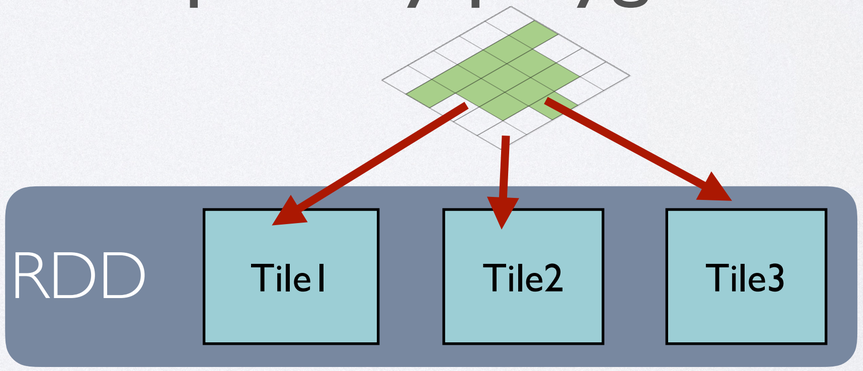
Pixel with 1 pixel buffer



Zonal Operation

- Algorithm 1: Join vector polygons with raster pixels
  - Rasterize each polygon to a mask layer
  - Broadcast it to each pixelRDD partition
  - Find matched pixels on each partition
  - Similar to a range query. Loop every polygon, not scalable

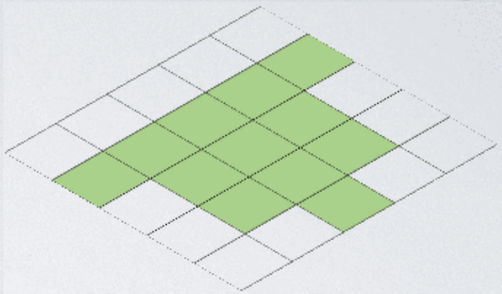


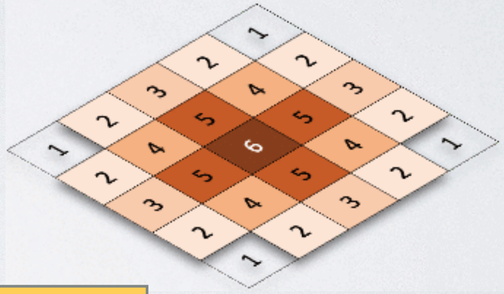


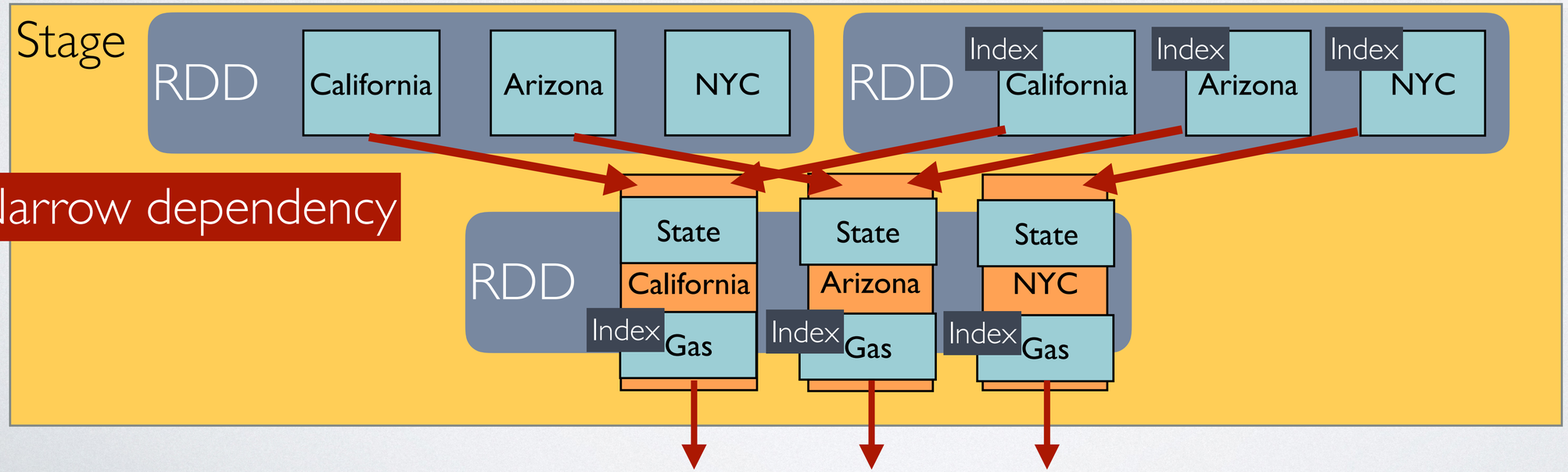
Scidb, https://www.paradigm4.com/

Zonal Operation

- Algorithm 2: Scalable
  - Convert each pixel back to a spatial point
  - Then use spatial join between polygons and points

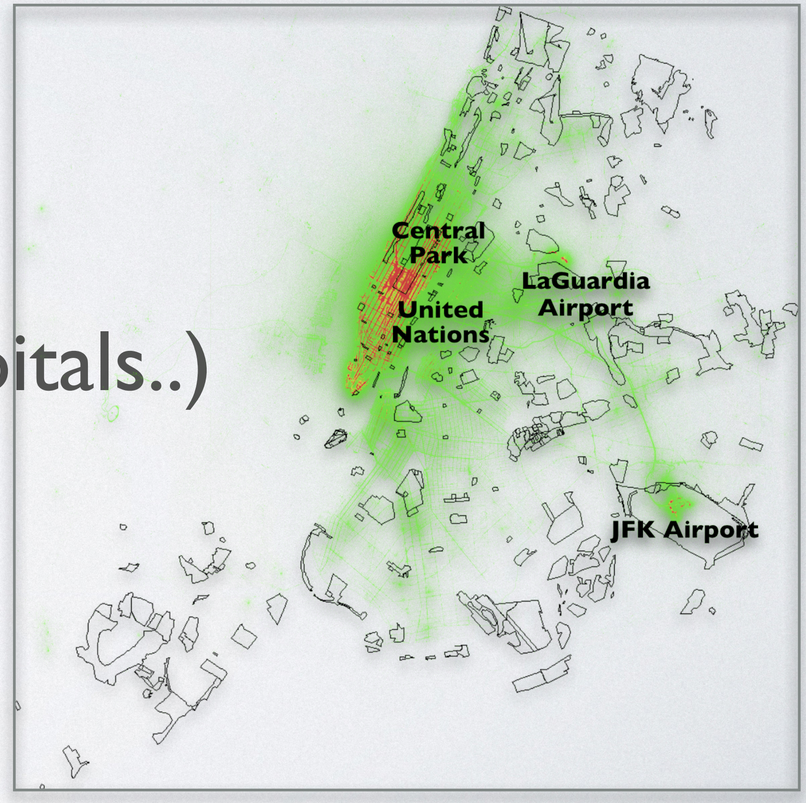






Spatial Data Mining Example

- Spatial co-location pattern mining in Spark
  - Use spatial join to build a whole data mining application
  - Use map algebra to visualize the result
  - Data: Taxi pick ups (1 billion)
  - Data: NYC landmarks (300, airports, hospitals..)



https://github.com/jiayuasu/GeoSparkTemplateProject/tree/master/geospark-analysis

What Is Spatial Co-Location

- Two or more species are often located in a neighborhood relationship. Africa lions co-locates with zebras
- Ripley's K function is often used in judging co-location
  - Executes multiple times
    - Compute adjacent matrix (distance join)
    - Get a K score
  - Form a curve for observation

Ripley's K Function Multivariate Spatial Patterns

1. Set a base distance (say, 1 meter)
2. Perform a distance join to get adjacent matrix between two species
3. Plug the matrix into Ripley's K and compute the K value
4. Repeat Step 2 and 3 until converge
   - Each time increase the distance

Philip M. Dixon, Encyclopedia of Environmetrics https://www3.nd.edu/~mhaenggi/ee87021/Dixon-K-Function.pdf

Write Co-Location Mining in Spark

- Create TripRDD (PointRDD)
  - Spatial partition
  - Build index
  - Cache indexed TripRDD into Spark memory
- Create LandmarkRDD (PointRDD)
  - Do not do spatial partitioning for now

```
tripRDD.spatialPartitioning(GridType.KDBTREE)
tripRDD.buildIndex(IndexType.QUADTREE, true)
tripRDD.indexedRDD = tripRDD.indexedRDD.cache()
```

Write Co-Location Mining in Spark

- Start iterations

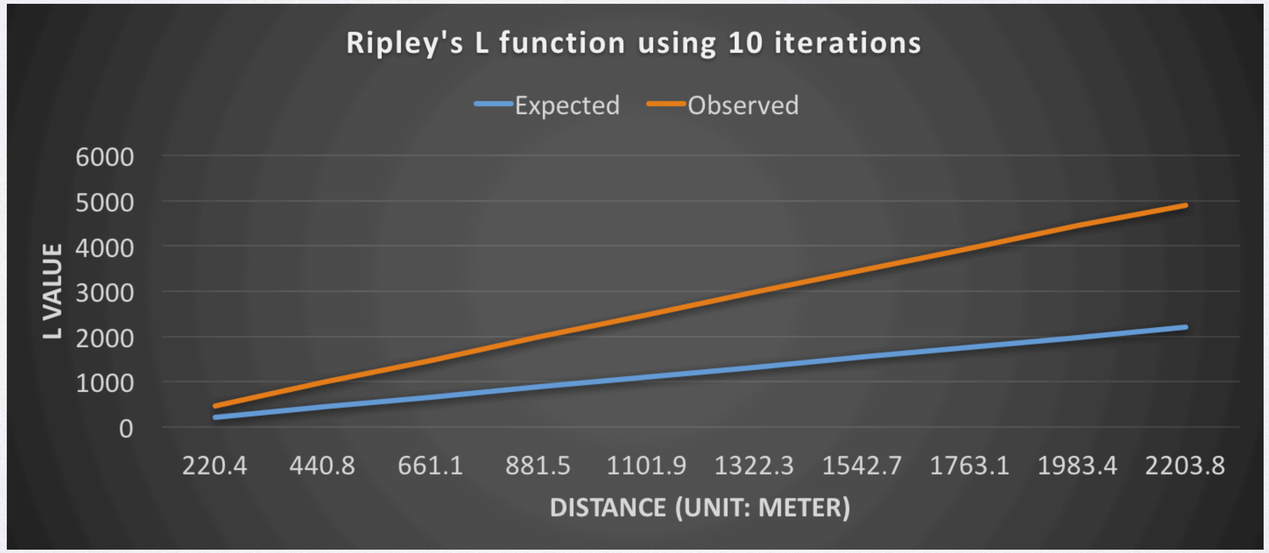
```
var bufferedArealmRDD = new CircleRDD(arealmRDD,currentDistance
bufferedArealmRDD.spatialPartitioning(tripRDD.getPartitioner)
```

  - Create a CircleRDD = LandmarkRDD + distance buffer
  - Spatial partition CircleRDD in the way with TripRDD
  - Perform distance join
  - Compute Ripley's K

```
var adjacentMatrix = JoinQuery.DistanceJoinQueryFlat(tripRDD,
bufferedArealmRDD,true,true)
```

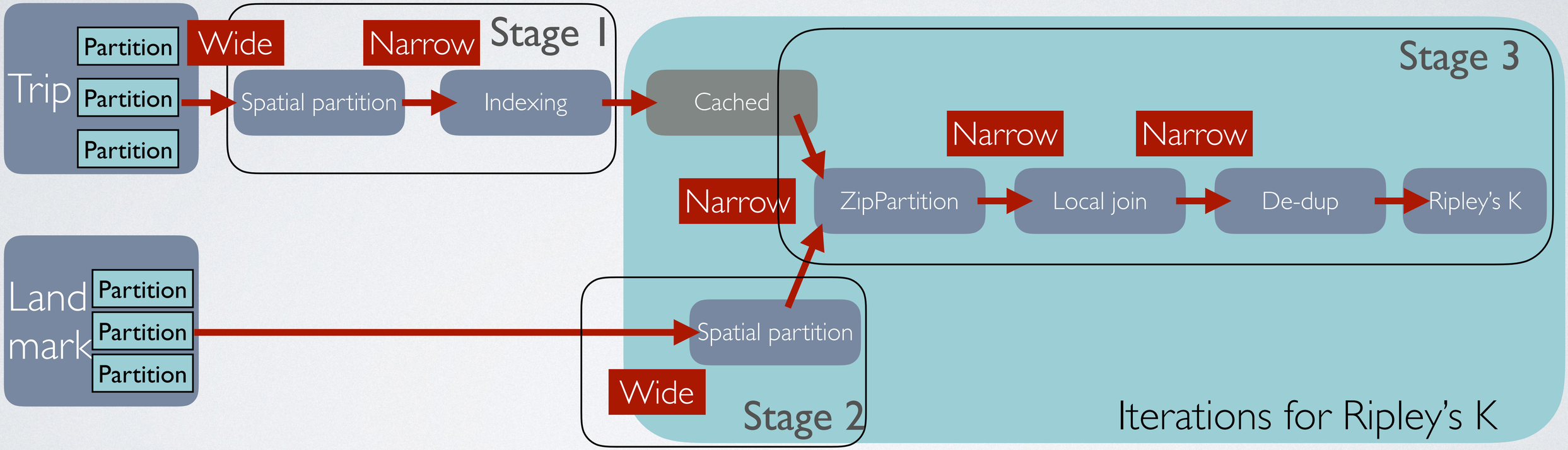
Write Co-Location Mining in Spark

- Mining result
  - Observed K value is always higher than expected K value
  - Conclusion: people call taxis at landmarks such as airport, hospital, library…



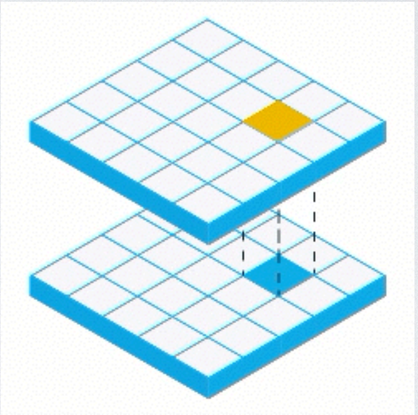
Write Co-Location Mining in Spark

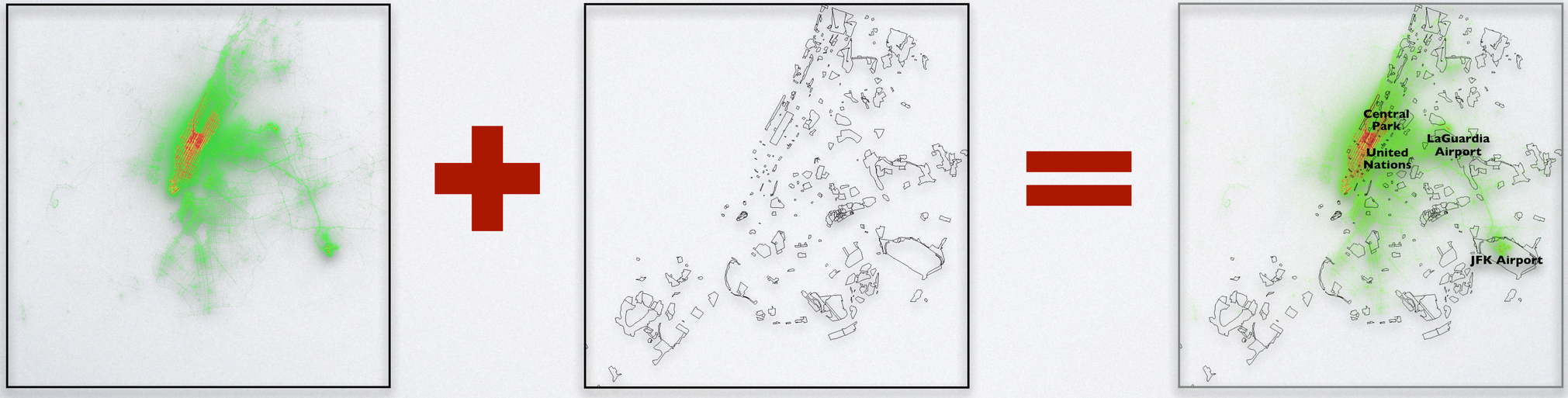
- DAG and data shuffle



Write Co-Location Mining in Spark

- Visual analytics
  - TripRDD
  - LandmarkRDD
  - Map algebra: local operation





Outline

Big geospatial data

Manage spatial data

Manage Spatio-Temporal Data

Spatial Data Analytics in Spark

Manage Spatial-Temporal Data

- What is spatial-temporal data?

Twitter	Location	Timestamp	Content
1	Point(14.315424, 108.339537)	11/11/2017 16:48	"Why would …"
2	…	…	…
3	…	…	…

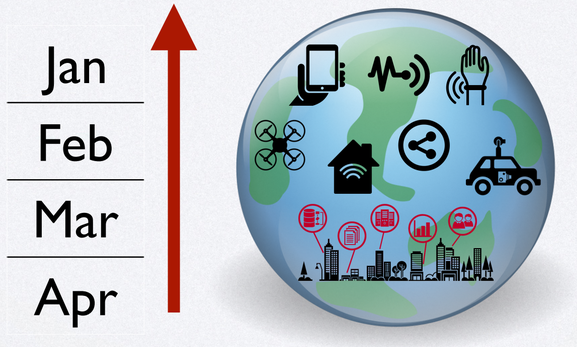
- What is a spatial-temporal query?

```
SELECT *
FROM tweets t
WHERE ST_Contains(t.loc, US) AND timestamp BETWEEN 11/1/2017 AND 11/30/2017
```

- Why do we need to care spatial-temporal data?
  - Temporal filter is done in a table scan. Inefficient!
  - Spatial data distribution / shape changes over time (Trajectories!)

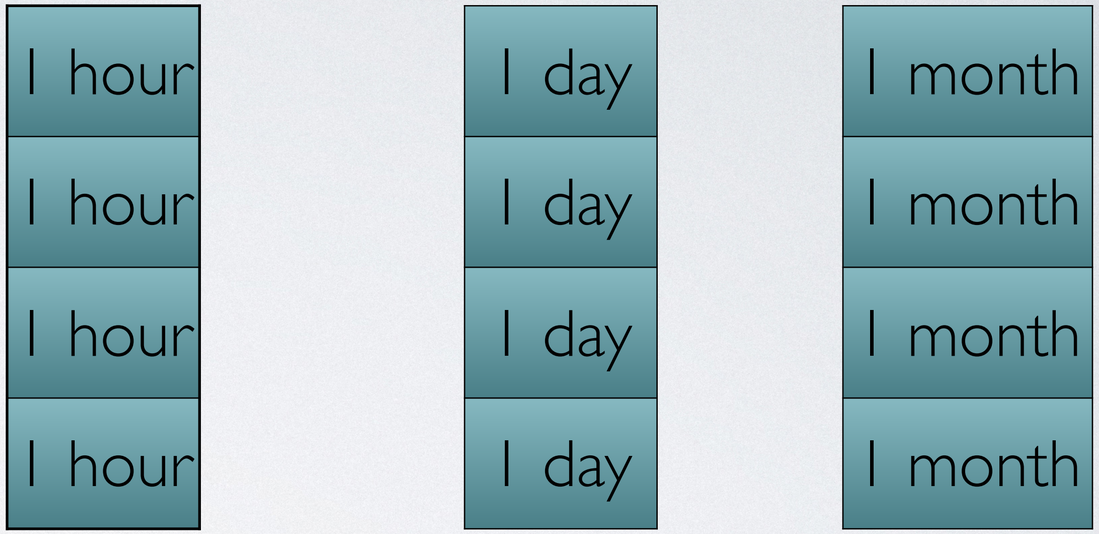
Spatial-Temporal Partitioning

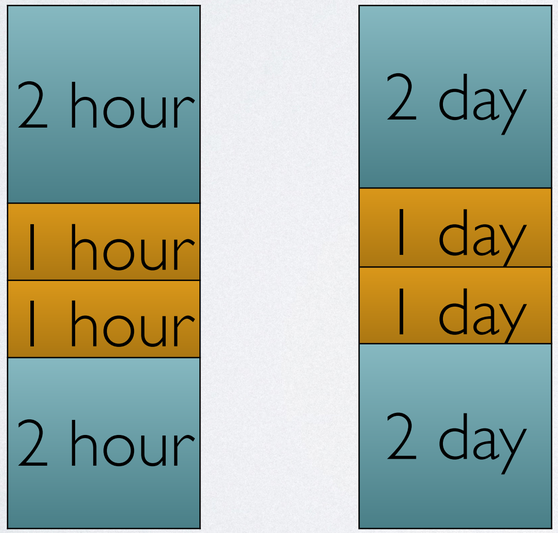
- Partition by spatial and temporal proximity / achieve load balance
  - Randomly sample the RDD and put it on the master
  - Build the global index / partition boundaries on the sample
  - Apply partitions….
- How to partition data by spatial and temporal attributes together?



Spatial-Temporal Partitioning

- Temporal partitioning
  - Uniform granularity
  - Load-balanced



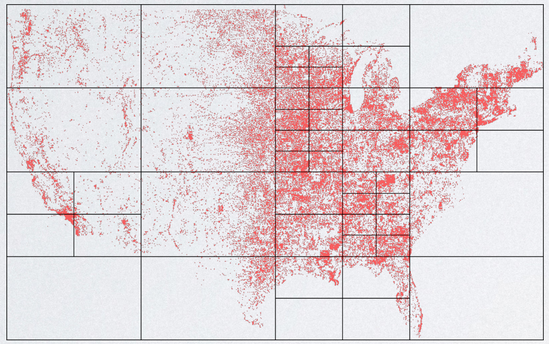


Compute Temporal and Spatial partitions separately

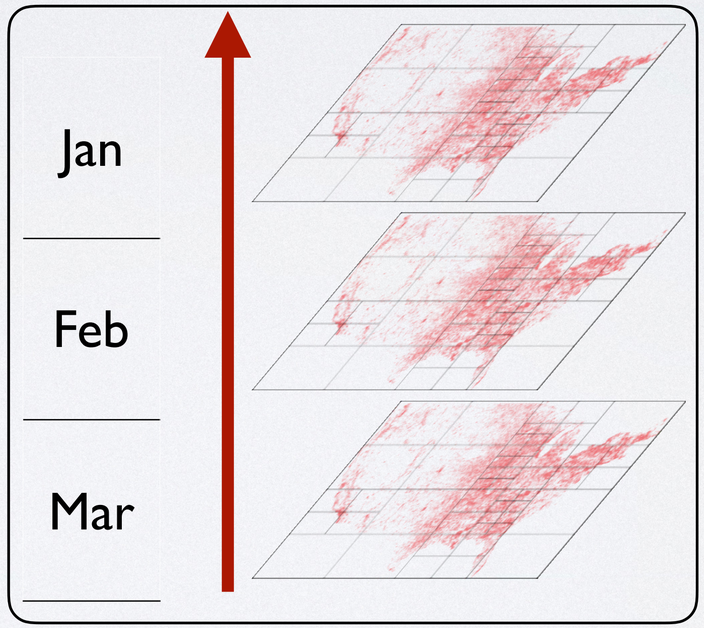
Whitman, Randall T., Michael B. Park, Bryan G. Marsh, and Erik G. Hoel. "Spatio-Temporal Join on Apache Spark." In *SIGSPATIAL* 2017.

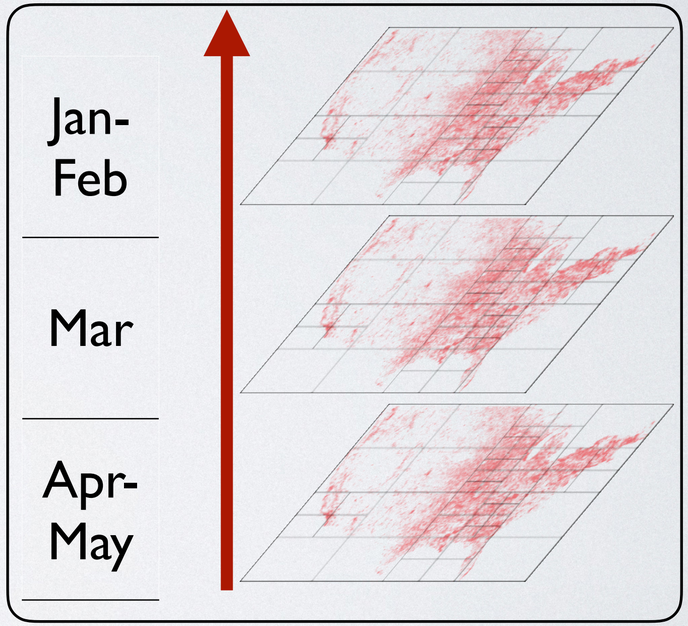
Spatial-Temporal Partitioning

- Spatial partitioning
  - Compute the spatial boundaries using KD-Tree, Quad-Tree,…



Compute Temporal and Spatial partitions separately

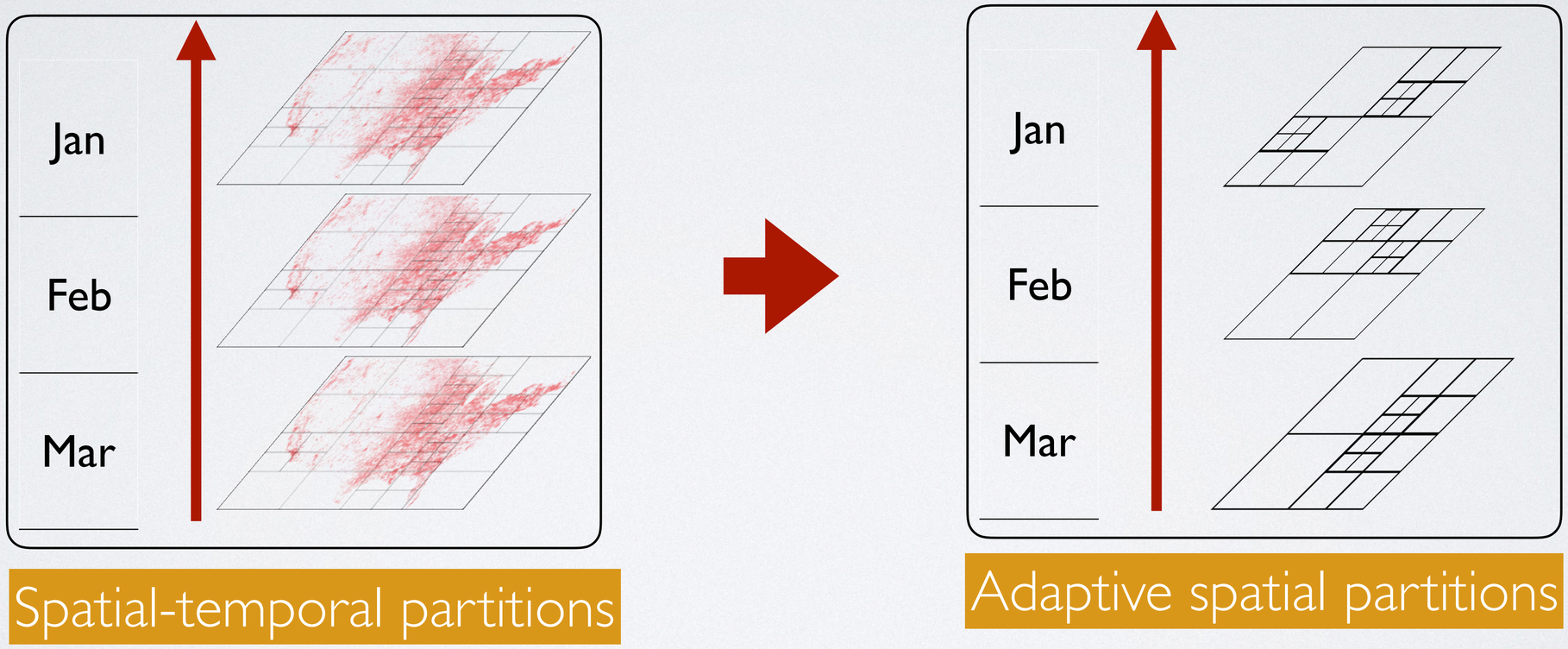




Spatial-temporal partitions

Spatial-Temporal Partitioning

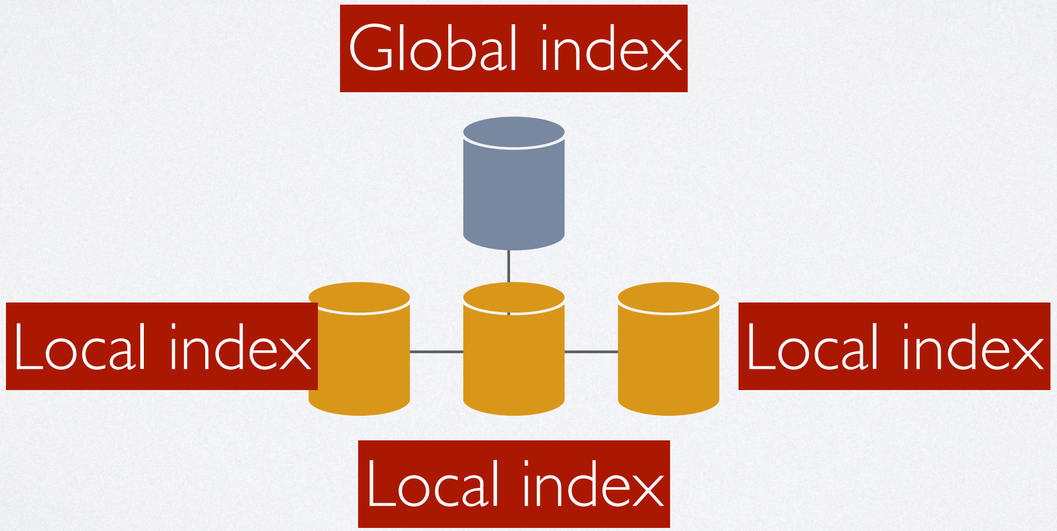
- What if the spatial data distribution changes over time?



Alarabi, Louai, Mohamed F. Mokbel, and Mashaal Musleh. "St-hadoop: A mapreduce framework for spatio-temporal data." *GeoInformatica* 22, no. 4 (2018): 785-813.

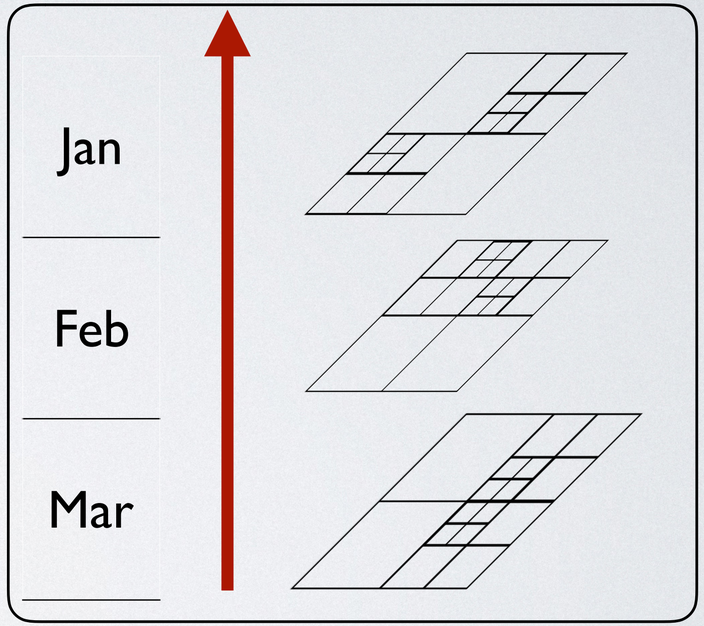
Spatial-Temporal Partitioning

- First, generate temporal partitions on the sample
- Then, create spatial partitions for each temporal partition
- Local spatial index is still built on each spatial-temporal partition



Spatial-Temporal Queries

- Spatial-temporal range query
  - Global index: temporal filter, then spatial filter
  - Prune partitions
  - Query remaining local indexes
- Spatial-temporal join query
  - Partition both datasets in the same way
  - Zip partitions by ID
  - Local join



Adaptive spatial partitions

Global index

Arizona State University

Wrap-Up

APACHE Spark™

Arizona State University

- Manage spatial data
  - Spatial partitioning
  - Query optimization
  - Spatial indexing
  - Object serializer
  - Spatial queries
  - Spark integration
- Manage Spatio-Temporal Data
  - Spatial-temporal partitioning
- Spatial Data Analytics in Spark
  - Distributed map visualization
  - Distributed map algebra
  - Spatial co-location pattern mining