Designing Distributed Geospatial Data-Intensive Applications

Ph.D. Course, **2022**

Instructors:

Prof. **Luca Foschini**, Associate Professor &

Dr. **Isam Mashhour Al Jawarneh**, Postdoctoral Research Fellow

{isam.aljawarneh3, Luca.foschini}@unibo.it

Department of Computer Science and Engineering (DISI), Università di Bologna

Part 4

state-of-art relevant papers discussion

29th July 2022

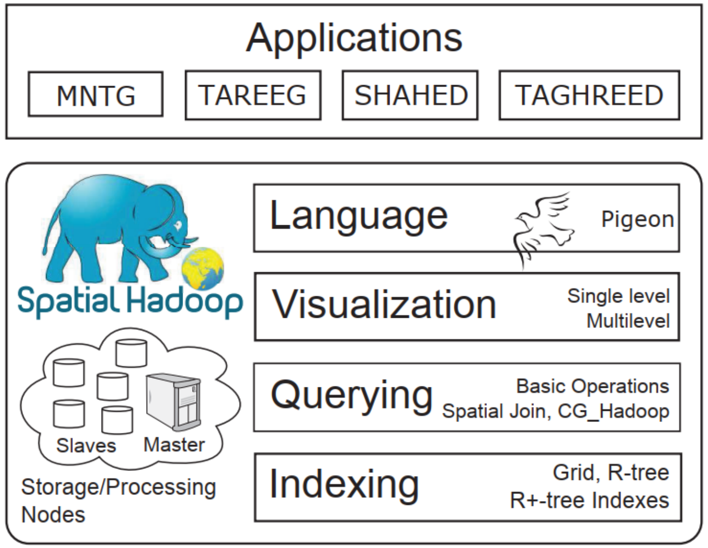
SpatialHadoop & HadoopViz

Related literature papers are available here

SpatialHadoop

[Image source](#)

- **full-fledged** system for spatial data, extends **Hadoop**’s core to support spatial data
- Four main layers: ***language***, ***indexing***, *query* ***processing***, and ***visualization***.
    - ***Language*** layer → standard spatial data types and query processing
    - ***indexing*** layer → grid, R-tree, and R+-tree for organizing spatial data in distributed file system (HDFS)
        - Two-level indexing → one *global* index that partitions the data across machines, and multiple *local* indexes that organize records inside each machine
    - ***query*** *processing* layer → **basic** spatial operations (**range** query and **k-nearest neighbor** query), join operations and computational geometry operations.
    - ***visualization*** layer→ for exploring big spatial data → generating images that provide bird’s-eye view on the data.
        - ***single level* images** → generated at a fixed resolution,
        - and **multilevel images** → generated at multiple resolutions for **zooming-in geo-visualization** effects



Spatial indexing in SpatialHadoop

- **Two-level spatial indexing** structure → one ***global*** index & multiple ***local*** indexes
  - **global** index **partitions** data into HDFS **blocks** and distributes them among cluster nodes,
  - **local** indexes organize records inside each block.
- Index **construction** algorithm:
  - one **MapReduce** job that runs in three phases

  (1)partitioning phase divides the embedding space into rectangles (spatial-aware)
    - partition the spatial objects by attaching each object to MBRs it intersects with
    - partition the space using a uniform **grid** (for **uniformly distributed data**)
    - Collect random sample from input file,
      - bulk load into in-memory **R-tree** (**STR** algorithm) → (for **skewed data**)
      - leaf nodes boundaries partition the file , where random sample is representative for data distribution
    - Spatial **proximity preservation** & load **balancing**

Spatial indexing in SpatialHadoop

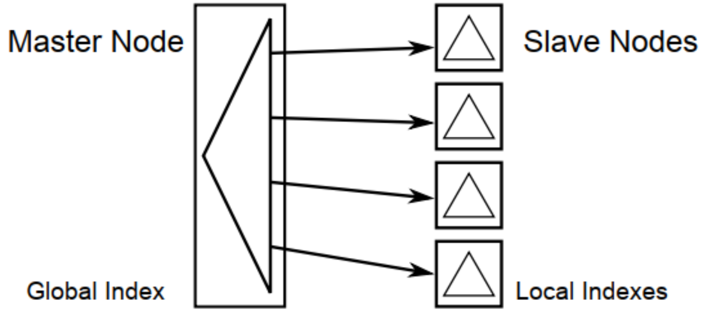
(1) partitioning phase

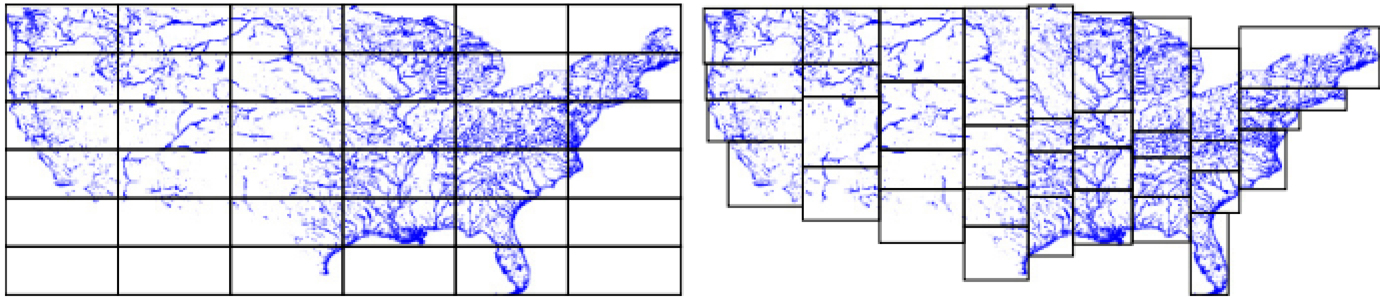
(2) local indexing phase, process each file independently on a single machine and construct an **in-memory local** index

- **reduce** function that accepts objects attached to every partition and stores them in a spatial index
- **R-tree** local index in every partition, local objects are bulk loaded using **Sort-Tile-Recursive** (**STR**) algorithm

(3) global indexing phase.

- Build an in-memory **global** index on the **master** node, **combining** all files in one file
- Indexing all HDFS file blocks using their MBRs as an indexing key
- Concatenate all local index files into one file representing the output indexed file





(a) Grid Partitioning (b) R-tree Partitioning

Image source

Partitioning & Indexing in SpatialHadoop

Image source

R-tree index →400 GB dataset → map objects in the world from OpenStreetMap

R-tree → partitions **overlap** → efficient for **range** queries → **partitions** fully within a query **range** are **copied** to **output** with no required **deduplication**

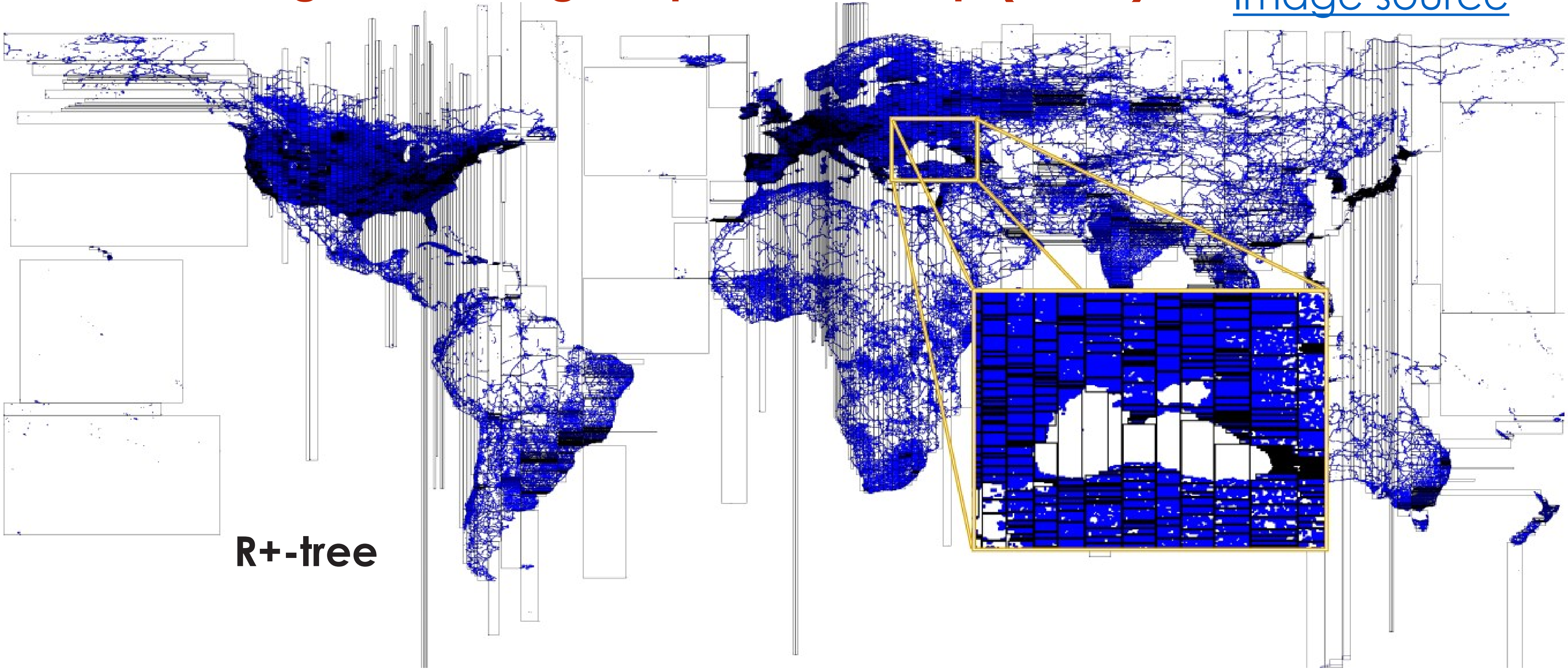
Blue lines → **data**
black rectangles → partition **boundaries** of the global index

adjusts the size of each partition based on data distribution → **load balancing**

Objects in each partition are stored in a single HDFS block in one cluster node

Partitioning & Indexing in SpatialHadoop (cont.)

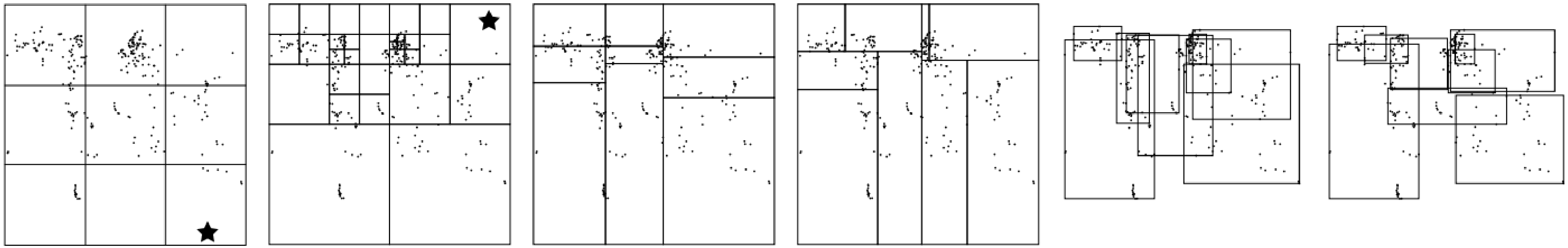
Image source



R+-tree → partitions are **disjoint** → some objects are replicated → efficient with spatial **join** → disjoint partitions allowing each one to be independently processed

Partitioning Techniques in SpatialHadoop

- **Z-curve**
  - sort points by Z-order curve, then partition the curve into partitions
  - Attach objects to cell by mapping the center point of their MBRs to a partition
- **STR**
  - bulk load objects into R-tree using STR algorithm
  - MBRs of R-tree leaf nodes are used as cell boundaries
- **Quad tree**
  - Insert objects into a quadtree
  - MBRs of leaf nodes are used as cell boundaries



(a) Grid (b) Quad Tree (c) STR and STR+ (d) K-d Tree (e) Z-curve (f) Hilbert Curve

Image source

Spatial Range Query in SpatialHadoop

*(1) **global** filter*

- Uses the **global** index with a **range filter** to choose blocks **intersecting** the **query window**
- Blocks fully within the query window are part of the final **output** as all inside objects within
- Partitions fully **outside** the query window are **pruned** as they don't contain points belonging to the output

(2) ***local** filter*

- Uses **local** index to retrieve **objects intersecting** with the **query window**
  - Processing a matching partition with a **range query** to retrieve points **matching** the query window

k Nearest Neighbor (kNN) in SpatialHadoop

*(1) **Initial** answer*

- Find k **closest points** to **query point** within same file block (i.e., **partition**) as query point
  - **filter** function that chooses the **partition** **overlapping** with the query point
  - Apply a **traditional** ***kNN*** algorithm to produce the **initial k set** from that partition

(2) ***Correctness*** *check*

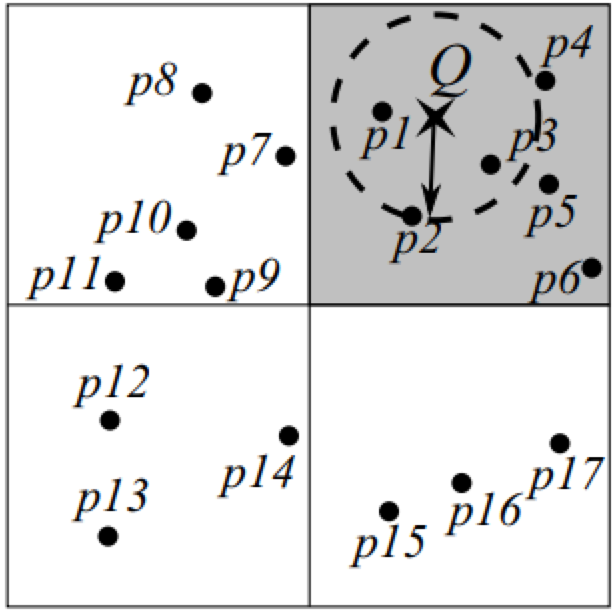
- Verify whether initial answer is final
  - Draw an **enclosing** **circle** centered at the **query point** with a **radius** that is equal to the **distance** from the **query point** to its k^{th} furthest point
  - If **circle** does not **overlap** any other **partition** than the one that is containing the **query point** → initial answer is final

k Nearest Neighbor (kNN) in SpatialHadoop (cont.)

(3) *Answer* ***Refinement***

- run a **range query** to retrieve **points** within the **MBR** of the **enclosing circle** (across overlapping partitions)
- scan range query result to find **k** closes points

Initial *answer*



Refinement

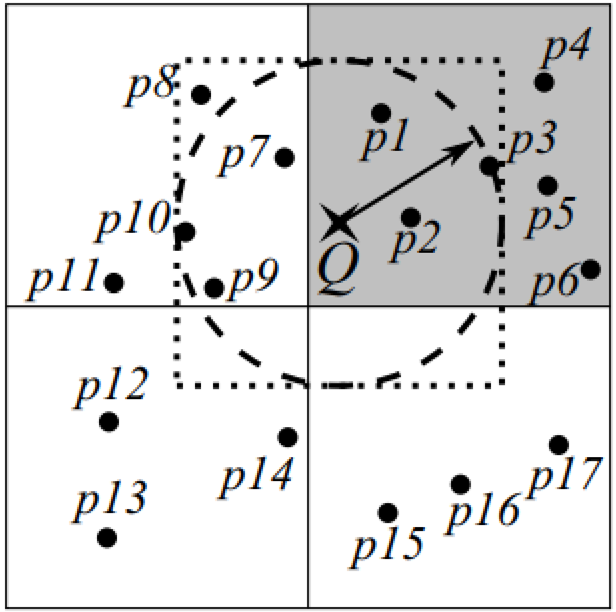
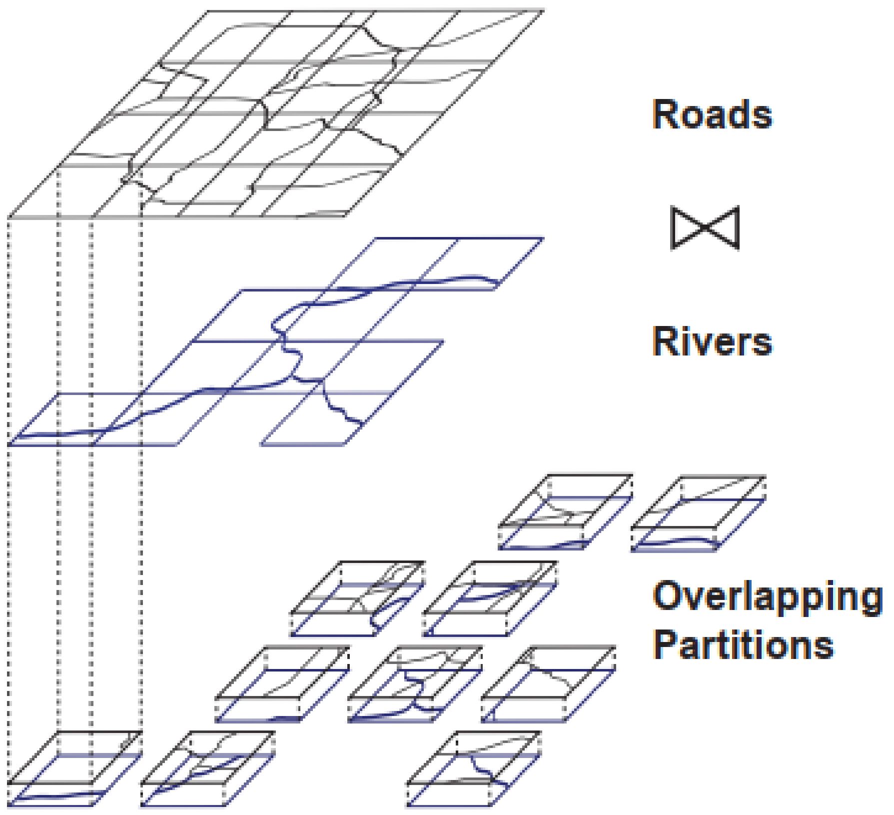


Image source

Distributed Spatial join in SpatialHadoop

- a **MapReduce-based** algorithm
  - Use the **two global indexes** (one **global** index for each table) to find **overlapping** pair of **partitions** (overlapping **MBRs**)
    - only **overlapping block pairs** constitute part of the **final answer** of the **spatial join**
    - Objects in **non-overlapping** block are **disjoint**
    - **Conventional spatial join** algorithm can be applied on the two global indexes to retrieve overlapping partitions pairs
  - Use the two **local** indexes in each partition pair to find matching **spatial objects** → a **map** function

Image source



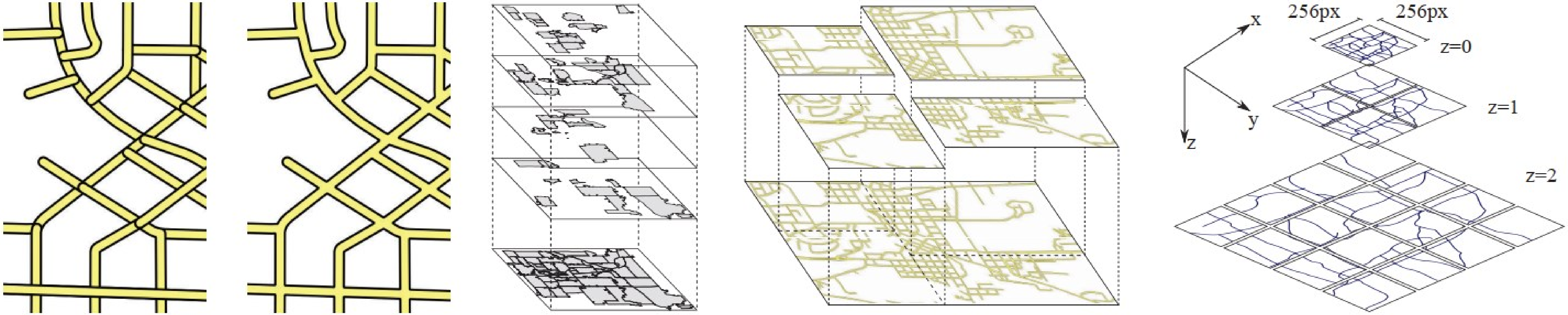
a spatial join between **Roads** and **Rivers**

partitioned using same **4 × 4 grid** structure

Parallel geo-visualization: SpatialHadoop Example

- three phases
  - **(1)Partitioning** phase. Either the default non-spatial Hadoop partitioner or a spatial partitioner
  - **(2)Rasterization (plotting)** phase, the machines in the cluster process the partitions in parallel and generate a partial image for each partition
  - **(3)Merging** phase, **fuse** partial images together to provide the output final image
  - **non-spatial partitioner** → partial images are ***overlaid***, *they all have the same size as the output image*
  - **spatial partitioner** → ***stitch*** partial images together

Image source



(a) Not smoothed (b) Smoothed (c) Overlay (d) Stitch (e) Multilevel Image

HadoopViz

- a framework for *big spatial data geo-visualization*
  - order of magnitude faster than existing techniques, which makes it more plausible for generating giga-pixel images over big data sets

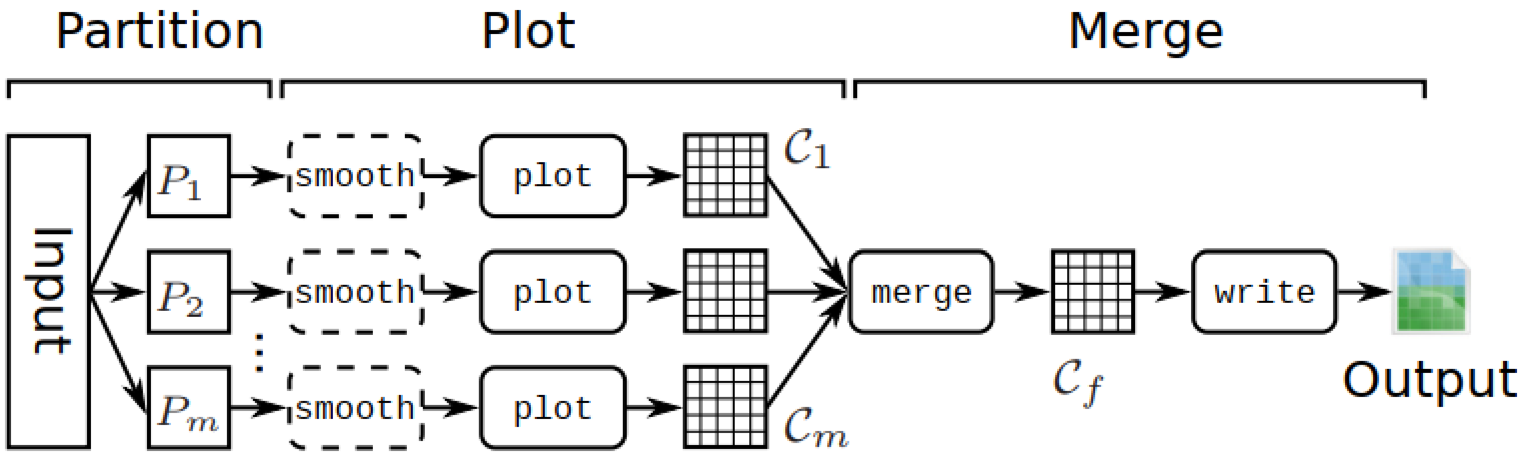


Image source

- ***plotting*** phase
  - *map* function
    - One mapper generates a partial image for one partition
  - 2-D matrix is a **canvas** to plot points contained within a partition
  - the mapper writes matrix contents as an intermediate record to be processed by the next *merging* phase.
- **Merging** phase
  - merge intermediate matrices to a final matrix and output it as a final image

Algorithm 1 Visualization using default Hadoop partitioning

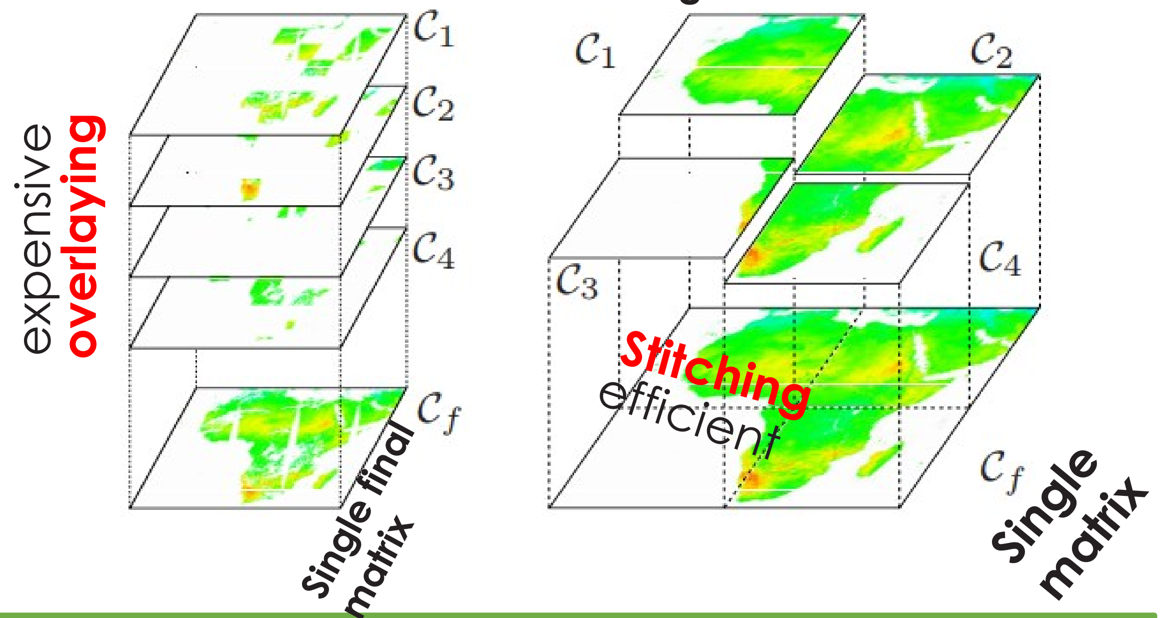
function SINGLELEVELPLOT(InFile, InMBR, ImageSize)
// The input is already partitioned into m partitions P_1 to P_m
// The Plotting Phase
for each partition $\langle P_i, BR_i \rangle$ **do**
    Create a 2D matrix $\mathcal{C}_i$ of size $ImageSize$
    Update $\mathcal{C}_i$ according to each point $p \in P_i$
end for
// The Merging Phase
Create a final matrix $\mathcal{C}_f$ with the desired ImageSize
For each reducer j, calculate $\mathcal{C}_j$ as the sum of all assigned matrices
One machine computes $\mathcal{C}_f$ as the sum of all $\mathcal{C}_j$ matrices
Generate an image by mapping each entry in $\mathcal{C}_f$ to a color
Write the generated image as the final output image

Image source

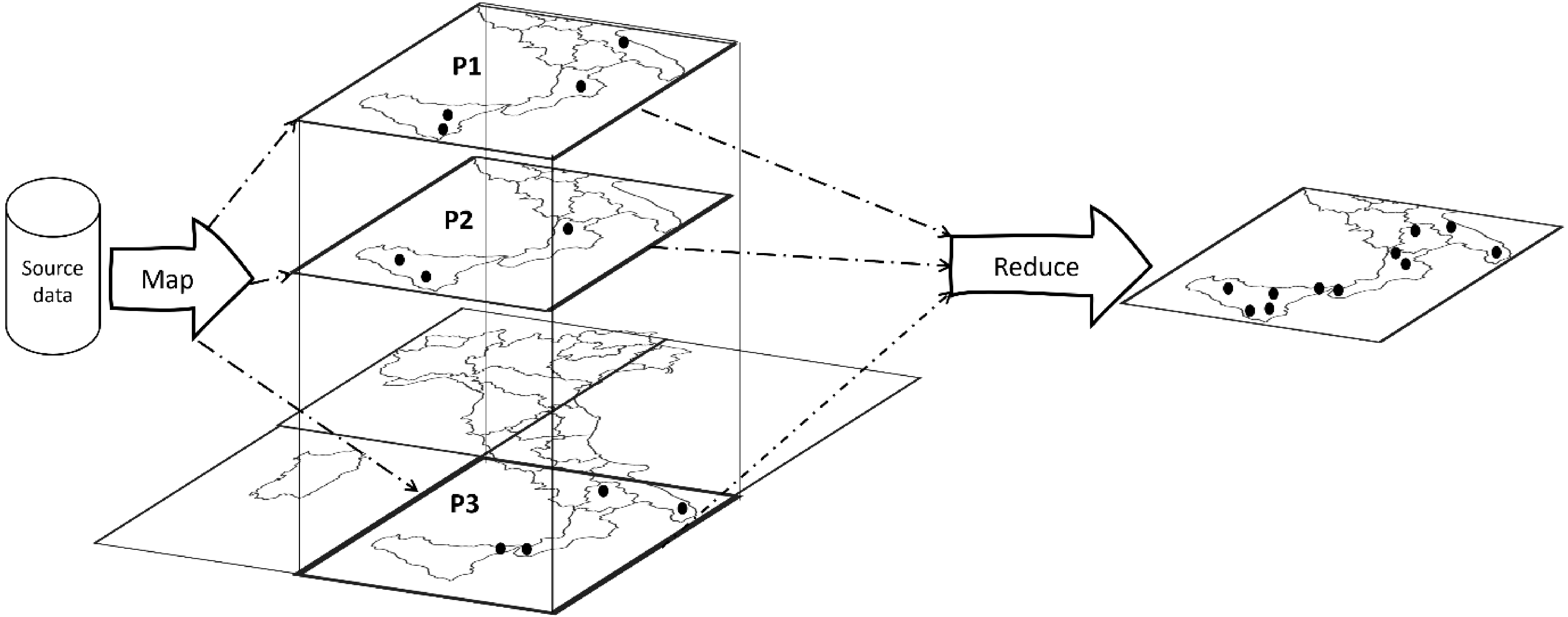
Single-level geo-visualization in SpatialHadoop

- **MapReduce** job
- **Merging** intermediate images
  - ***Partitioning***
    - Utilizes spatial partitioning
    - **map** function that uses a SpatialHadoop partitioner
      - Splits embedding space into *disjoint* cells and attach each point to all overlapping cells
  - ***plotting***
    - ***reduce*** function → points in every partition are grouped together and geo-visualized for producing one ***partial*** image
    - initializes a matrix
    - scans points in a single partition and updates **sum** and **count** statistics of the corresponding **array entries**
  - ***merging***
    - Merges **intermediate matrices** to a **single matrix** by **stitching** matrices in accordance with their locations in the final image

Image source



- ***partial** merge*. **reduce** function
  - runs **locally** in each machine
  - each **reducer** creates a single matrix (canvas), adding all **assigned** intermediate matrices to it
  - Each matrix is added to a position in accordance with the **MBR** of its corresponding partition
- ***final** merge* → read matrices written by all reducers and combine them to a single matrix (C_f) (**stitching** with **spatial** partitioner or **overlaying** **non-spatial** partitioner)


Source data
Map
P1
P2
P3
Reduce

Batch processing

- * ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "Locality-Preserving Spatial Partitioning for Geo Big Data Analytics in Main Memory Frameworks," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6. **(IEEE GLOBECOM 2020)[1] [PDF]. DOI:** 10.1109/GLOBECOM42002.2020.9322544
- [C6] * ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini, R. Montanari and A. Zanotti, "*In-memory spatial-aware framework for processing proximity-alike queries in big spatial data,*" in 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2018, pp. 1-6. **[PDF].** **DOI:** 10.1109/CAMAD.2018.8514950
- [C1] * ***I. M. Aljawarneh***, P. Bellavista, A. Corradi, R. Montanari, L. Foschini and A. Zanotti, "*Efficient spark-based framework for big geospatial data query processing and analysis,*" in 2017 IEEE Symposium on Computers and Communications (ISCC), 2017, pp. 851-856. **[PDF].** **DOI:** 10.1109/ISCC.2017.8024633

Scalable storage of big geo-referenced data in distributed storage NoSQL systems

- [j4] * ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Efficient QoS-Aware Spatial Join Processing for Scalable NoSQL Storage Frameworks," IEEE Transactions on Network and Service Management, 2020. **DOI:** 10.1109/TNSM.2020.3034150 . Journal ***Impact Factor*** (indexed in *ISI Thomson Reuters*): **3.878. [PDF]**
- [C5] * ***I. M. Al Jawarneh***, P. Bellavista, F. Casimiro, A. Corradi and L. Foschini, "*Cost-effective strategies for provisioning NoSQL storage services in support for industry 4.0,*" in 2018 IEEE Symposium on Computers and Communications (ISCC), 2018, pp. 1227. **[PDF].** **DOI:** 10.1109/ISCC.2018.8538616

Spatial Approximate Query Processing

- [j5] *** *I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari. QoS-Aware Approximate Query Processing for Smart Cities Spatial Data Streams. *Sensors* 2021, *21*, 4160. **DOI**: **10.3390/s21124160**.
- [C10] * ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Spatially Representative Online Big Data Sampling for Smart Cities," in 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2020: IEEE, pp. 1-6. **[PDF]. DOI:** 10.1109/CAMAD50429.2020.9209294
- [C9] * ***I. M. Al Jawarneh***, P. Bellavista, L. Foschini and R. Montanari, "*Spatial-aware approximate big data stream processing,*" in 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1-6. **(IEEE GLOBECOM 2019)** [1]. **[PDF].** **DOI:** 10.1109/GLOBECOM38437.2019.9014291

Multidomain geospatial analysis

- * ***I. M. Al Jawarneh***, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Efficiently Integrating Mobility and Environment Data for Climate Change Analytics," in 2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2021: IEEE, pp. 1-5. **[PDF]**. **DOI:** 10.1109/CAMAD52502.2021.9617784

Survey

- * ***<u>I. M. Al Jawarneh</u>***, P. Bellavista, A. Corradi, L. Foschini and R. Montanari, "Big Spatial Data Management for the Internet of Things: A Survey," Journal of Network and Systems Management, pp. 1-46, 2020. **DOI**: 10.1007/s10922-020-09549-6

Geo-visualization in parallel computing frameworks

- GeoSparkViz: A Cluster Computing System for Visualizing Massive-Scale Geospatial Data
  - Related paper is available here

GeoFlink

- Salman Ahmed Shaikh, Komal Mariam, Hiroyuki Kitagawa, and Kyoung-Sook Kim. 2020. **GeoFlink: A Distributed and Scalable Framework for the Real-time Processing of Spatial Streams**. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20). Association for Computing Machinery, New York, NY, USA, 3149–3156. https://doi.org/10.1145/3340531.3412761
- S. A. Shaikh, H. Kitagawa, A. Matono, K. Mariam and K. -S. Kim, **"GeoFlink: An Efficient and Scalable Spatial Data Stream Management System,"** in IEEE Access, vol. 10, pp. 24909-24935, 2022, doi: 10.1109/ACCESS.2022.3154063.