Hadoop GIS –

RESQUE query engine

Parallel spatial query execution

Global partition indexing

Local on demand spatial indexing

Spatial DBMS -  
SDBMS stores vectors and Rasters

SDBMS is Object relational DBMS

Why not conventional DBMS

Table joint are time consuming

Solution -> Dual architecture - Relational DBMS + File storage systems

Columns store polygons, lines and points in such architecture (Object DBMS)

ORDBMS (Object oriented DBMS)

POSTGIS – POSTGRE + Spatial database

This is similar to OOPs in SQL queries

Spatial Join operations

Very heavy -> Based on spatial operations

Petabytes of data generated for bio-analytics. Hence, its important to have efficient solutions.

2 main problems –

1. Spatial skew data problem (Lack of symmetry) – Leads to load imbalance and long response times
2. Boundary object problem – Incorrect results

Query cases –

1. Feature aggregation (on spatial) – Mean values or finding distribution
2. Spatial queries – point based, spatial joins
3. Complex spatial queries
4. Integrated spatial + feature queries
5. Global spatial queries (Finding some regions, etc)

Steps –

1. Space partitioning to form ‘tiles’
2. The ‘tiles’ are assigned UIDs, merged and stored over HDFS
3. Pre-processing: Global indexing, remove data that doesn’t need tiles based processing
4. Parallel processing of tiles
5. Border handling
6. Post-processing: Joining results
7. Data aggregation and output to HDFS

How to handle skewness in data (Tiles)-

Break down complex and high density tiles recursively.

RESQUE ->  
Global region index for tiles – Stored in binary format and shared over the cluster using Hadoop distributed cache mechanism

Queries – Map/Reduce/Combination jobs

Boundary object handling -   
1. Multiple alignment – Border objects appear in multiple tiles (Better, lesser IO bottlenecks)

2.Multiple matching

The system basically adds Spatial Queries support to Hive

SDBMS uses Spatial indices for Querying (Pages based random access)  
This cant be used in Hadoop/Spark as the inherent architecture uses big blocks of data (Big blocks of data for batch processing)

Pre-generated indices not good idea – Dynamic queries cant be run on them!

Hence the approach used -   
1. Global spatial indices based on tiles and regions

2/ On demand index

Global region indeces are small – Cann be shared using Hadoop’s caching - Quickly lookup tiles relevant to the query