cluster (see Section 5.3) and avoiding unnecessary computation on partitions that are impossible to have qualified data. A good spatial partitioning technique keeps all Spatial RDD partitions balanced in terms of memory space and spatial computation, aka. load balancing.

Spatial RDD represents a very large and distributed dataset so that it is extremely time consuming to traverse the entire Spatial RDD for obtaining the spatial distribution and partition it according to its distribution. GeoSpark employs a low overhead spatial partitioning approach to take advantage of global spatial distribution awareness. Hence, GeoSpark swiftly partitions the objects across the cluster. The spatial partitioning technique incorporates three main steps as follows (see Algorithm 1):

Algorithm 1 SRDD spatial partitioning

```
Data: An original SRDD
Result: A repartitioned SRDD
/* Step 1: Build a global grid file at master node */
Take samples from the original SRDD A partitions in parallel;
Construct the selected spatial structure on the collected sample at master node;
Retrieve the grids from built spatial structures;
/* Step 2: Assign grid ID to each object in parallel */
foreach spatial object in SRDD A do
    foreach grid do
        if the grid intersects the object then
            Add (grid ID, object) pair into SRDD B;
    // Only needed for R-Tree partitioning
    if no grid intersects the object then
        Add (overflow grid ID, object) pair into SRDD B;
/* Step 3: Repartition SRDD across the cluster */
Partition SRDD B by ID and get SRDD C;
Cache the new SRDD C in memory and return it;
```

Step 1: **Building a global spatial grid file:** In this step, the system takes samples from each Spatial RDD partition and collects the samples to Spark master node to generate a small subset of the Spatial RDD. This subset follows the Spatial RDD's data distribution. Hence, if we split the subset into several load balanced partitions that contain a similar number of spatial objects and apply the boundaries of partitions to the entire Spatial RDD, the new Spatial RDD partitions should still be load-balanced. Furthermore, the spatial locations of these records should also be of a close spatial proximity to each other. Therefore, after sampling the SRDD, GeoSpark constructs one of the following spatial data structures that splits the sampled data into partitions at the Spark master node (see Fig. 2). As suggested by [16], GeoSpark takes 1% percent data of the entire Spatial RDD as the sample:

- **Uniform Grid:** GeoSpark partitions the entire two-dimensional space into equal sized grid cells which have the same length, width and area. The boundaries of these grid cells are applied to the entire Spatial RDD. The partitioning approach generates non-balanced grids which are suitable for uniform data.
- **R-Tree | Quad-Tree | KDB-Tree:** This approach exploits the definition of capacity (fanout) in the classical spatial tree index structures, R-Tree [14],