Indexing Large Spatial Data using Hadoop, a closer look at spatialhadoop and another complementary QuadTree implementation

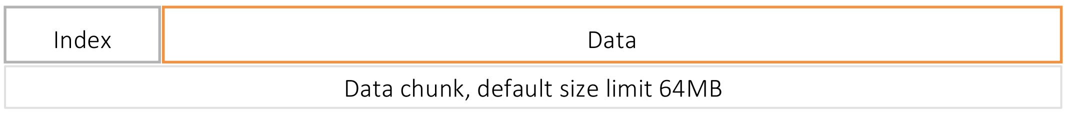
Donghan Miao

This report dissects spatialhadoop implementation details. As a complement, I also implemented a standalone indexing-query system using quadtree on hadoop.

Building Index

In general, database storage consists of persisted index structure and data storage. In big data environment, we leverage the advantage of distribution and parallelism to achieve scalability.

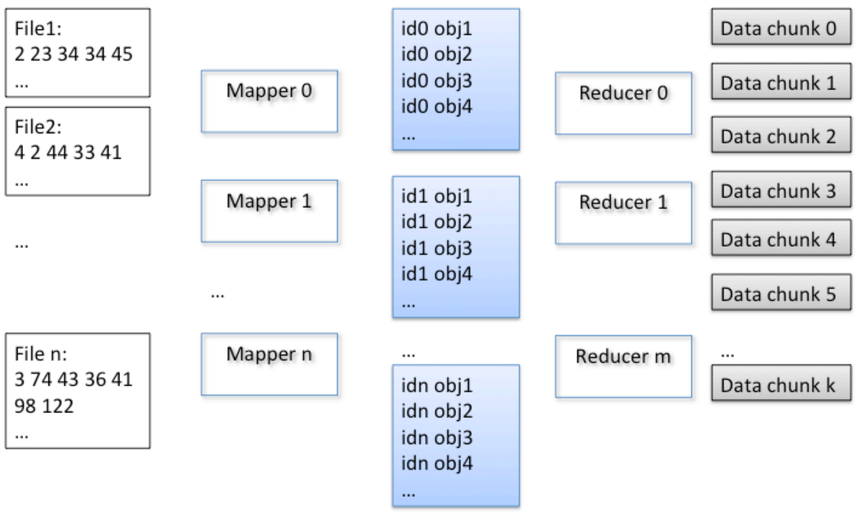
Indexing process partitions large data set into even-sized data chunks (the number of chunks is determined by the volume of spatial dataset). Each data chunk is a small but fully-fledged database storage.



By default, the block size of Hadoop distributed file system (HDFS for short hereafter) is 64MB. Therefore our default chunks should be close to but no larger than that size, in order to guarantee i/o efficiency.

MapReduce Job1 – Index Building and data persistence

In our case, input data are formatted texts where each line represents a polygon object. Mapper input takes <line number, line text string> as key-value pairs. Mapreduce framework will split the files into smaller splits before submitting them to mappers. Multiple mappers will work in parallel to process the input splits. The level of parallelism is determined by the volume of data set or can be customized by the user.



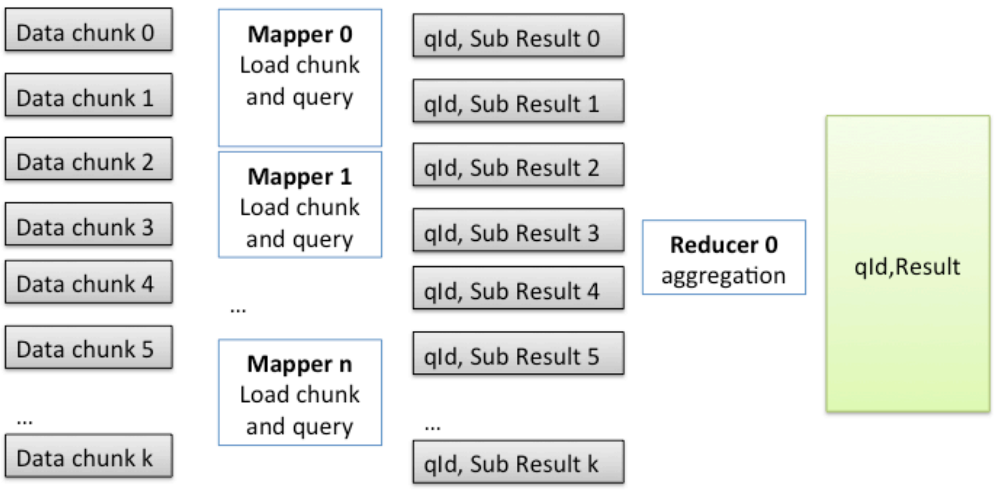
Graph 1, MapReduce process for Tree Building

Mappers convert the text string into a spatial object and emits <mapperInstanceID, object> pairs for reducers. The emitted key-value pairs sharing the same key will be sent to same reducer, and the reducers will collect an appropriate number of objects (a number adjusted to make data chunk close to 64MB) and build spatial index for them. Upon finishing index building, the index and raw data will be combined as a single data chunk and emit as <id, chunk> as a single binary file to the file system.

MapReduce Job 2 - Executing a Query

Executing a range query is also a Mapreduce job, where input is a number of data chunks built from the first Mapreduce job. In each mapper, a customized input format reader was defined to load the data chunks into memory, where their original data structure are restored – the first section of file restored as index structures, and the second section of file are simply loaded into a large byte array.

Query was also executed in the mapper phase. Index structure returns a set of object ids (id is a combination of file position and length) for any given range search. According to the object ids, the actual data of the object will be retrieved from the byte array.



Graph 2, Query In MapReduce

After that, a set of queried objects is sent to reducer with key being the queryID, value being a spatial object. In this way, results belong to the same query will be aggregated in the same reducer.

Data Structure - QuadTree and Modifications

A quadtree is a tree data structure in which each internal node has exactly four children. Quadtrees partition a two-dimensional space by recursively subdividing it into four quadrants or regions.

Some modification was made for efficiency. In one leaf node, more than one objects can be stored if the minimal bounding rectangle of an object has intersection with another object in that node. In this way, objects cluster will not result in over division of the quadtree.