**Indexing Larger Spatial Data using Hadoop, a closer look at spatialhadoop and its complementary QuadTree implementation**

Donghan Miao

**Building Index**

Generically, database storage consists of persisted index structure and data storage, but neither part should infinitely scale up. In big data environment, we leverage the advantage of distribution and parallelism to achieve scalability.

Indexing process is 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.

Data

Index

Data chunk, size limit 64MB

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, to guarantee i/o efficiency.

**Partitioning**

In the beginning, Mapreduce framework will split the files into smaller chunks before submitting them to mappers. Multiple mappers will work in parallel to process the input splits. The level of parallelism is largely determined by the volume of data set.

In our case, input data are formatted texts where each line indicates a polygon object. Mapper input takes <line number, line text string> as key, value. Each mapper instance has its own unique id. Mappers convert the text string into a spatial object and emits <mapperID, object> pairs for reducers.

Emitted key, value pairs share the same key will be sent to same reducer, Reducers receiving the spatial objects will collect an appropriate number of objects 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> to the filesystem.

**Executing a Query**

Executing a query is also a mapreduce job, where input is the output from the first mapreduce job. A customized inputformat reader was defined to load the data chunks into a set of key value pairs. In each mapper, the data chunk was loaded into memory as index structures and raw data block, query was executed on the index structure and return result was sent to reducer with key being the queryID, value being the spatial object. In this way, results belong to the same query will be aggregated in the same reducer.