QuadTree Index for Spatial Objects on Apache MapReduce

Brief Documentation

Donghan Miao, Feb. 6th 2014

1. Source and Compile

File List

```
├── hadoop.core
│   └── hadoop-core-1.2.1.jar
├── libs
│   ├── commons-lang-2.4.jar
│   ├── commons-logging-1.1.3.jar
│   ├── commons-logging-adapters-1.1.3.jar
│   └── commons-logging-api-1.1.3.jar
└── src
    ├── io
    │   ├── QuadTreeFileOutputFormat.java
    │   └── QuadTreeInputFormat.java
    ├── mapReduceTasks
    │   ├── QuadTreeIndexer.java
    │   └── QuadTreeQuery.java
    └── quadIndex
        ├── FileLoc.java
        ├── InputParser.java
        ├── Point.java
        ├── Polygon.java
        ├── QuadTree.java
        ├── QuadTreeTest.java
        ├── Rectangle.java
        └── SpatialObj.java
```

Packages

- io

  This package includes implementation for customized input format and output format interface of Apache hadoop MapReduce.

- mapReduceTasks

  This package defines two MapReduce jobs, namely, index building job, query search job.

- quadIndex

  this package is an overall wrap-up for different spatial objects and quadtree structure.

Additional libraries

- libs
- hadoop.core

Minimum subset of Apache hadoop standard library are included for compilation.

Latest source code is available on Github.

```
>> git clone https://github.com/miaodonghan/QuadTreeHadoop.git
```

2. **Deploy hadoop**

   The hadoop version we are using is 1.2.1 stable release. Please see Apache Hadoop official website for deployment instructions.

   Single node deployment

   http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html

   Cluster deployment

   http://hadoop.apache.org/docs/r1.2.1/cluster_setup.html

3. **Prepare the data source**

   Location

   The source data should be placed under **/user/username/src** directory on hdfs.

   Format

   Raw data should be in plain text format. Each line indicates a spatial Object. Points, Rectangles and Polygons are supported.

   Point:

   1, x, y

   Rectangle:

   2, x, y, width, height

   Polygon:

   numberOfpoints, x1, y1, x2, y2, …, xn, yn

4. **Indexing the spatial data**

   Command

   ```
   hadoop jar tb.jar mapReduceTasks.QuadTreeIndexer
   ```

   Output

   Output Path is /user/username/out, part-000XX are index files, XX.rawdata are formatted raw data for spatial objects.

5. **Run a query on mapreduce**

   Query Files

   All queries should be placed in /user/username/query/query.txt, the format is as follows:

   queryID, x, y, width, height

   where queryID is a unique integer

Command

```
hadoop jar tb.jar mapReduceTasks.QuadTreeQuery
```

Output

All output results are produced into /user/username/result

6. MapReduce Process

QuadTree Builder

Mapper

Spatial objects are evenly distributed across mapreduce clusters, each working node should received same number of spatial objects. Quadtree is built in this phase, each quadtree will be assigned with an id. The key, value emitted to reducer is <TreeId, QuadTree>

Reducer

Reducer is responsible for writing the binary representation of the structure received from mapper. Customized output format and customized record writer is used in this phase.

Query Executer

Mapper

We use serialized Quadtree data from the first mapReduce job as input, in the mean time, query is loaded from user provided text file. Each query has a unique id, and the query is sent individually to each working node. rangeQuery() method is called in the mapper, which returns a set of index ids of queried spatial objects.

With index ids queried from quadtree, the raw data will be retrieved from formatted raw data file using those ids.

After serialization of the spatial objects queried, they are delivered to reducers.

Reducer

Reducer aggregate all results from different nodes, noting that query result with same queryID will be aggregated on only one reducer designated by MapReduce system.