Big Data Management Project 2

Spring 2017

Team 18 Caitlin Kuhlman, Wenlei Cao

Code Organization:

* Project2
  + sjoin
  + json
  + kmeans
* lib
* project2.jar

We have provided a precompiled jar called project2.jar. To compile the code yourself you must include the lib folder on the build path.

**Question 1. Spatial Join**

**To run:**

hadoop jar <jarfile> sjoin.Q1 <point filepath> <rectangle filepath> <output path> <<optional window>>

Example:

hadoop jar project2.jar sjoin.Q1 /data/points.csv /data/rectangles.csv /data/output W(1,3,3,20)

**Explanation:**

The package sjoin contains 6 classes:

DataGenerator

PointMapper

RectangleMapper

Geometry

Q1Reducer

Q1

This DataGenerator class generates two datasets:

* Points : x, y
* Rectangles: id, x, y, h, w

**\*the spatial join program expects the “id” field from the input file, it is used in the output specified in the assignment**

Q1 is the driver. To perform the spatial join in parallel we use a grid index. The domain space to be joined is divided into evenly spaced grid cells. A hash function is then used to map both points and rectangles to grid cells. Each point is mapped to the single cell it falls in, and each rectangle is mapped to every cell it overlaps.

We implemented two custom writable classes PointWritable and RectangleWritable. To map both of these objects to the same reducer we make use of hadoop’s GenericWritable class. We implement this interface in a class called Geometry, which wraps the point and rectangle classes.

Finally in the Q1Reducer each reduce task processes a single cell, checking whether the points fall within rectangles. Each output line is in the format <id, (xy)>.

**Question 2. JsonInputFormat**

**To run:**

hadoop jar <jarfile> json.Q2 <input path> <output path>

Example:

hadoop jar project2.jar json.Q2 /data/airfield.txt /data/output

**Explanation:**

The package json contains 3 classes:

JsonRecordReader

JsonInputFormat

Q2

Q2 is the driver. It utilizes the other two classes to read the input file one json record at a time. The JsonInputFormat extends the FileInputFormat class and contains methods which define the RecordReader and the file split size. We set the max split size to be roughly 1/5 of the airfield file to produce 5 splits as specified in the assignment.

The JsonRecordReader wraps a LineReader object which reads one line at a time from a text file. Extra logic is added to parse json objects, and keep track of where they start and end to process multiple input splits.

In Q2, the MRMapper class, extracts the flag number as a key and counter is included for each record as a value. In the MRReducer class, the counter is aggregated based on flag number.

**Question 3. Kmeans**

**To Run:**

hadoop jar <jarfile> kmeans.Q2 <input path> <centroid path> <output path> <k>

Example:

hadoop jar project2.jar kmeans.Q2 /input/points.csv /data/centroids output 50

**Explanation:**

The package kmeans contains 6 classes:

DataGenerator

KMeansMapper

KMeansCombiner

KMeansReducer

PointWritable

Q3

Q3 is the driver. It first generates k random points and puts them in a file on HDFS specified by the user. This file is sent via the distributed cache to mappers and reducers. It is reused in each iteration for the updated centroids.

The program runs up to 5 MapReduce jobs, stopping earlier only if the centroids have not changed from the previous iteration.

We include the 2 optimizations mentioned in the assignment: the use of a combiner class, and the use of a single reducer. In the reducer we check if the centroids have changed from the previous iteration and emit a message in the cleanup function to pass this information to the driver.