CS561: Advanced Database Systems
Spring-2014

Project 3

Total Points: **100**

Release Date: **04/20/2014**

Due Date: **05/01/2014 (11:59PM)**

Short Description

In this project, you will write java map-reduce jobs that implement spatial join operation as well as learn more details about Hadoop's Input Formats.

Problem 1 (Spatial Join) [50 points]

Spatial join is a common type of joins in many applications that manage multi-dimensional data. A typical example of spatial join is to have two datasets: ***Dataset P*** (set of points in two dimensional space) as shown in Figure 1a, and ***Dataset R*** (set of rectangles in two dimensional space) as shown in Figure 1b. The spatial join operation is to join these two datasets and report any pair (rectangle *r*, point *p*) where *p* is contained in *r* (or even in the border of *r*).

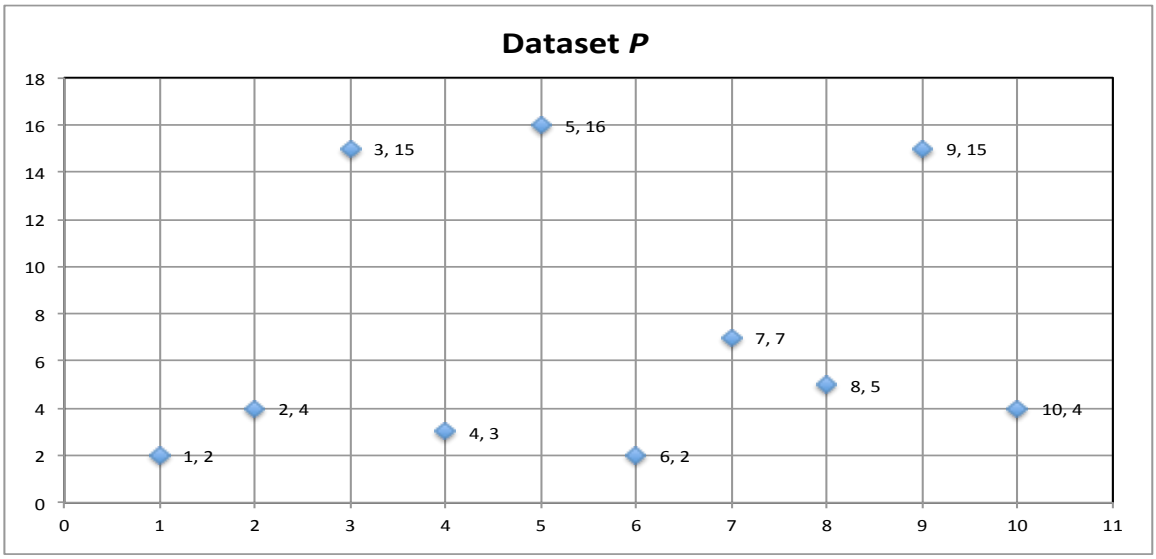

Figure 1a: Set of 2D Points

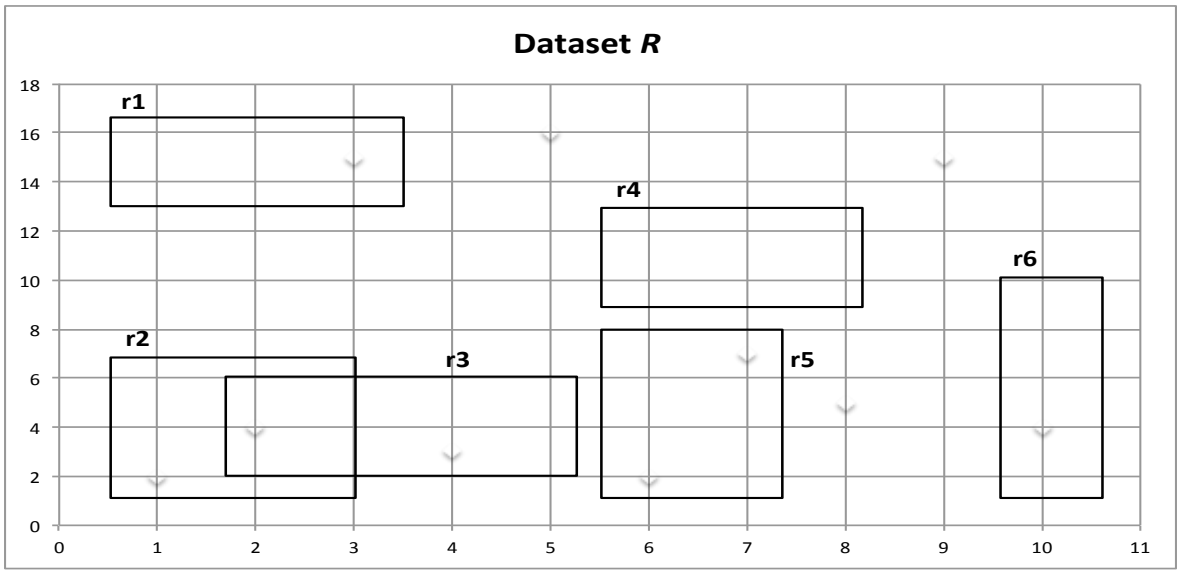

Figure 1b: Set of 2D Rectangles

For example, the join between the two datasets shown in Figure 1, will result in.

<r1, (3,15)>
<r2, (1,2)>
<r2, (2,4)>
<r3, (2,4)>
<r3, (4,3)>
<r5, (6,2)>
<r5, (7,7)>
<r6, (10,4)>

Step 1 (Create the Datasets)[10 Points]

- Your task in this step is to create the two datasets *P* (set of 2D points) and *R* (set of 2D rectangles). Assume the space extends from 1…10,000 in the both the X and Y axis. Each line will contain one object.
- Scale each dataset *P* or *R* to be at least 100MB.
- Choose the appropriate random function (of your choice) to create the points. For the rectangles, you will need to also select a point at random (say the top-left corner), and then select two random variables that define the *height* and *width* of the rectangle. For example, the height random variable can be uniform between [1,20] and the width is also uniform between [1,5].

Step 2 (MapReduce job for Spatial Join)[40 Points]

In this step, you will write a java map-reduce job that implement the spatial join operation between the two datasets *P* and *R* based on the following requirements:

- The program takes an optional input parameter W(x1, y1, x2, y2) that indicate a spatial window (rectangle) of interest within which we want to report the joined objects. If W is omitted, then the entire two sets should be joined.
    - Example, referring to Figure 1, if the window parameter is W(1, 3, 3, 20), then the reported joined objects should be:

      <r1, (3,15)>
      <r2, (2,4)>
      <r3, (2,4)>

- You should have a single map-reduce job to implement the spatial join operation.
    - ***Hint:*** *Make use of the prior knowledge of the space boundaries, i.e., 1…10,000 in each dimension. Your algorithm should make use of that.* Think of how this space can be divided in order to do the spatial join operation in parallel.

Problem 2 (Custom Input Format) [50 points]

So far, all of the given assignments use text files as input, and hence you use 'TextInputFormat()' to read the files. In this problem, you will learn more about Hadoop input formats and you will write your custom one to read the input data.

Step 1 (Create the Datasets)[10 Points]

Assume we have a customer dataset, where each record is stored as follows:

{ Customer ID: *<id>,*
  Name: *<name>*,
  Address: *<addr>*,
  Salary: *<salary>*,
  Gender: *<gender>*
},
{ Customer ID: *<id>,*
  Name: *<name>*,
  Address: *<addr>*,
  Salary: *<salary>*,
  Gender: *<gender>*
},
….

- In this step, you are required to create a customer dataset with the above format.
- Scale it to be at least 100MBs
- Make sure within a single record, there is no "{" or "}". These brackets should identify the start and end of each record.

Step 2 (Map Job with a Custom Input Format)[30 Points]

- Now, to do any job on the above dataset using the standard "TextInputFormat()", the map function must be complex as it needs to collect many lines to form a single record. This complexity will repeat with each written job over the above dataset.
- A better way is to write a custom input format, call it "***CustomerInputFormat***" that reads many lines from the input file until it gets a complete record, and then coverts them to a list of comma separated values in a single line, and then pass it to the map function.
    - E.g., each input to the map function should be: *<id>, <name>, <addr>, <salary>, <gender>*

- Your task is to write this new "CustomerInputFormat" and use it in a simple job to report the information of customers whose salary greater than 1,000.

- ***Hint:*** You need to understand first the "***FileInputFormat***", "***TextInputFormat***", and "***LineRecordReader***" classes.
    - Your "CustomerInputFormat" should call "CustomerRecordReader", and the latter is the one responsible for reading many lines and forming a single comma-separated line for the mapper.
    - These classes exist under: home/Workspace/Hadoop-1.1.0/src/mapred/org/apache/Hadoop/mapred/
    - You can add your new classes under the same directory (inside Hadoop) and re-compile it, if that is easier for you.

Step 3 (File Splits)[10 Points]

Input formats not only form the records that are passed from the input file to the mapper code, but they also decide on how to divide the file. By default, each HDFS block is assigned to a single mapper. However, input format have the logical concept of "splits", and each split (by default corresponds to an HDFS block) is assigned to a single mapper. Developers can do many tricks with this middle concept of "splits".

- Your task in this step is to modify the input format to read the entire customer file as one split. Hence, there should be only one mapper to process the file regardless of its size.
  - It is up to you to either create a new input format for Step 3, e.g., call it "SplitCustomerInputFormat", or you allow parameters to be sent to the "CustomerInputFormat" in Step 2. This parameter---which should be in the job context---will tell the input format whether to divide the file or not.

- ***Hint:*** Again, your focus will be on these three files "***FileInputFormat***", "***TextInputFormat***", and "***LineRecordReader***".

What to Submit

You will submit a single zip file containing the java code needed to answer the queries above. Also include a .doc or .pdf report file containing any required documentation.

How to Submit

Use blackboard system to submit your files.

Demonstrating Your Code

Each team will schedule an appointment with the instructor to demonstrate the project. Demonstration should be within the week after the due date.