CSE 512 Project Phase 3 Requirement

1. Experiments on operations developed in Phase 2

1.1 Metrics:

1. CPU utilization per node when an operation is running on your cluster
2. Memory utilization per node when an operation is running on your cluster
3. Run time when an operation costs.

1.2 Strategy:

1. Operations: You have to test each of the operations in Phase 2
2. Dataset: You can use datasets from previous phase or whatever dataset you have as long as it can show your program is good and efficient.
3. Cluster size: You have to test them on different cluster sizes (1 node, 2 nodes, 4 nodes. You can try more if you want) to show your program's performance on distributed system.

1.3 Tool and hints:

1. For the run time, you have to find it from Spark Web UI. More statistics of one Spark application can be found in Spark History Server if you can make the server run.
2. You have to use Ganglia which is a cluster performance monitoring tool for measuring CPU utilization and memory utilization in Linux. For each metric in each operation, you have to provide necessary screenshots of Ganglia Web UI (CPU/Memory) or Spark Web UI (RunTime).

2. Application: Spatial aggregation (Example: Heat map)

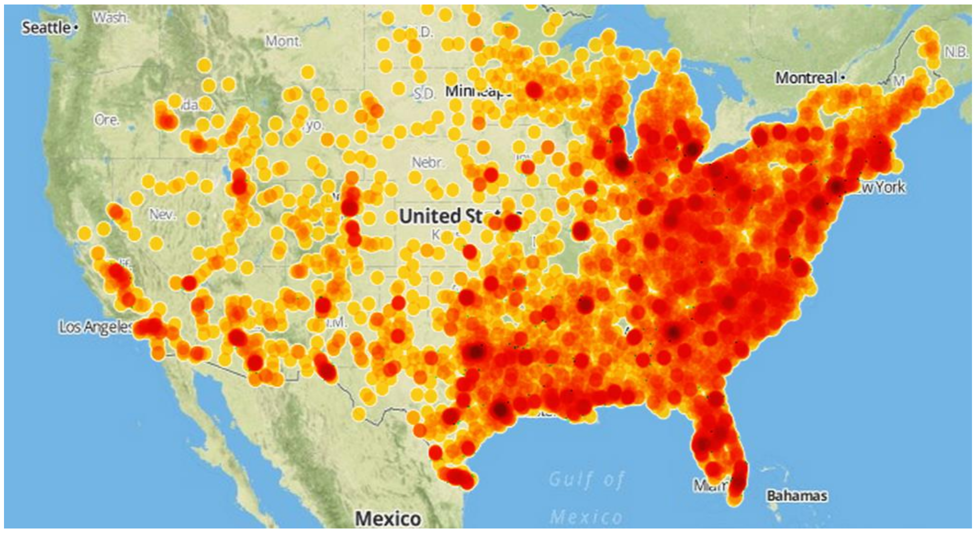
2.1 Introduction

2.1.1 What is spatial aggregation?

Objects which are distributed in a spatial plane may aggregate together in some particular areas because of some non-spatial attributes. In other words, in some particular areas, the density (or amount) of these objects are higher than that in others.

2.1.2 One example: Heat map

a. Definition: A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. (From Wikipedia)
b. What it looks like?



(Image from: https://sigacts.files.wordpress.com/2014/07/heatmapoffamilydollarstores.png)

As the image shows, the area which is red has more density of these objects.

2.1.3 What spatial queries can help?

Spatial aggregation is a kind of join query which has a set of points as the target set and a grid file as the query area sets. And this query should be able to count the number of objects which lie in each of the girds.

2.2 Requirements

1. Functionality: Implement a Java application which can take the same inputs as the spatial join query (one target point dataset and one query rectangle set). The output should follow the format <Rectangle, Count> and the count represent the number of points which lie in this polygon.
2. Datasets: We provide two datasets: zcta510 (Query rectangle set) and arealm (Target point set) in BlackBoard. You have to use them for testing.

3. Experiments on the application developed in Step 2

1. Use the same metrics and strategy mentioned in step 1 to run your experiments. The only difference is that you have to use the datasets we provide in BlackBoard.
2. Cluster capacity: Some of your own clusters may not be powerful enough to test datasets provided in BlackBoard. One alternative is that try Amazon Web Service EC2 free tier. Please arrange your Amazon EC2 usage reasonably and keep it in the free tier. We are not responsible for your fee on Amazon EC2.

4. Final submission

1. Your report which contains the performance on each metrics per operation. Please use charts and screenshots and attach necessary analysis.
2. Your java Spatial Aggregation source code which should leverage the spatial join API written by you in Phase 2. (The submission code should be like this: aggregation.java).
3. Please put the report and the source code to such a folder "GroupName_phase3" (e.g. Group1_phase) and compress this folder.