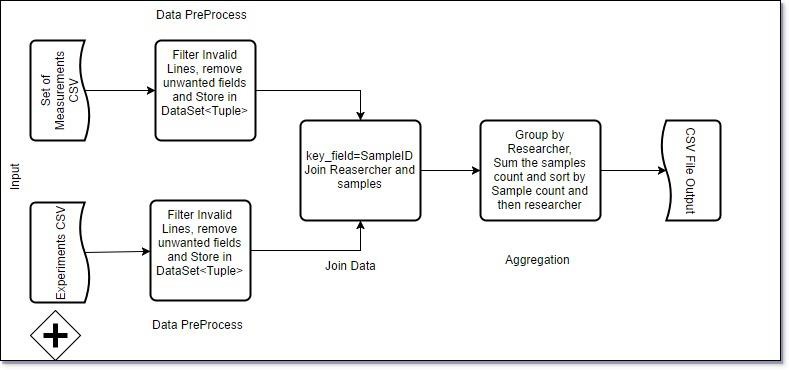
Cloud Computing Assignment 2 using Flink - Report

# Task1: Number of (valid) measurements conducted per researcher:



Steps for Computation:

1. Invoke an ExecutionEnvironment variable to start a flink instance.
2. Data pre-processing: Read files using ReadCsvFile() data files and Filter the header row and unwanted columns. Store the read measurements and researcher data into respective Tuple<fields> DataSet.
3. Call a custom filter() method which checks for valid measurements and filters out invalid records.
4. Using DataSet.join(), both the experiments and the measurements data are joined on a common field SampleId.
5. Using the Aggregation functions available in flink, we groupBy() on Researcher and then count the sum of samples per each researcher. We then sort() the number of samples and researcher and write it to a CSV file using the Write as CSV function.

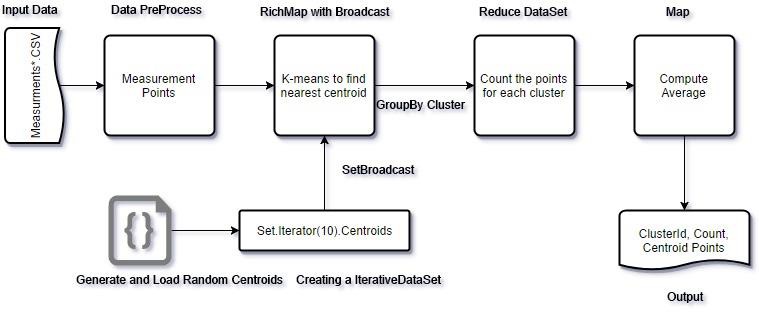
Output:

|  |  |
| --- | --- |
| Darwin | 1622404 |
| Gesner | 1001333 |
| Arber | 977943 |
| Baltimore | 888242 |

# Task2: K-means clustering of the measurements:

Steps for Computation:

1. Read the measurements from the CSV files and remove and also call a custom function to generate 5 random cluster centroids based on MIN and MAX of Measurements.
2. Assign an IterativeDataSet<> to set an Iterator for 10 iterations Iterator(10) for the initial centroids.
3. Implementing a RichMap function, we map() each set of measurments and iterate it with the initial centroids to find the distance and repeat the process for 10 iterations. This will give us a more accurate centroid by finding the euclideanDistance() for each point and the current centroid in the iteration. A new centroid is assigned whenever the there is a measurement with lesser Euclidean distance than the previous point and the centroid.
4. We store the measurements with it’s associated cluster ID. Next, we will take the count of points for each cluster and the divide by the points to find the mean centroid value.
5. Then we get the final output and write it as clusterID, Number of points, Measurements.



Output:

1 509551 1.630410427825682 3.5898112440030148

4.494369884924171

2 380037 1.9739789047408511 0.936892044074653

4.964160759902291

3 754102 0.6365732544218116 2.751327204547926

3.4869687607976423

4 323174 2.3784101305736467 0.5902433020570987

1.632155440029206

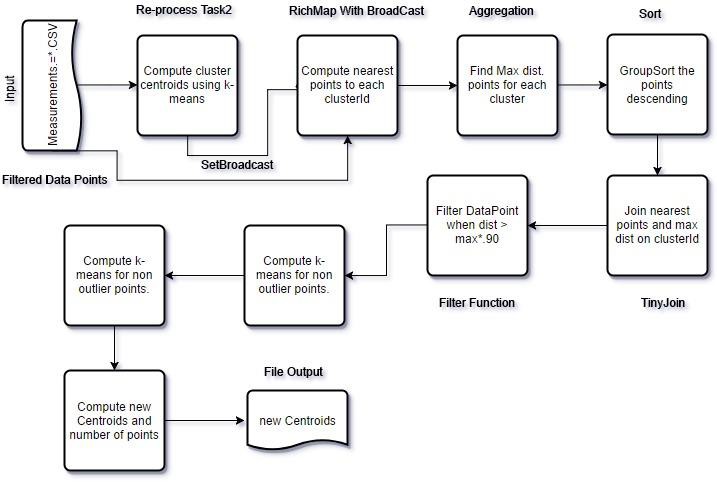
5 810956 0.523327523245414 0.2364343518254988

0.7052266026516888

# Task3: Outlier Removal and Re-clustering:

Steps for computation:

1. For this task, first we compute as same as for Task2. Once we have the value of 5 centroids,
2. We will compute the nearest points associated with each clusterId. We accomplish this by using a RichMap function by sending the Centroids as Broadcast variable and compute the nearest distance for each point.
3. We then use flink aggregation to find the Max () distance for each cluster and then sort it in descending order.
4. We use the tinyJoin() to join the nearest points table and the max distance table with clusterID as key.
5. Then we filter the records when dist > max distance\*0.90.
6. Now we store the measurements with outliers removed and then run K-means to recluster the points and the centroids.



Output:

1 1035897 0.9234657002221272 0.277623506132368 0.8168725105951696

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 297849 2.2708369324926343 | 0.4653182896299803 | 4.322815734828734 |
| 3 | 645105 1.0076281630106705 | 3.662325512919601 | 3.9284602578386627 |
| 4 | 494669 0.7100551494817732 | 2.2042894584297748 | 3.2667541665780684 |
| 5 | 304219 2.036036361653943 | 2.5082221492181653 | 5.32962823162919 |

|  |  |  |
| --- | --- | --- |
| Task | Small Data Runtime | Large Data Runtime |
| 1 | 1078 | 4784 |
| 2 | 3493 | 11983 |
| 3 | 10031 | 28678 |

APPENDIX:

hdfs:///user/lban2974/task1.csv hdfs:///user/lban2974/task2.csv hdfs:///user/lban2974/task3.csv