# Distributed K-Means using PySpark

## Problem Formulation:

Minimize sum of squared distances between data points and cluster centroids.

## Parallelization Strategy:

Assignment and centroid update parallelized using mapPartitions and reduceByKey.

## Implementation:

RDD-based implementation with broadcast variables.

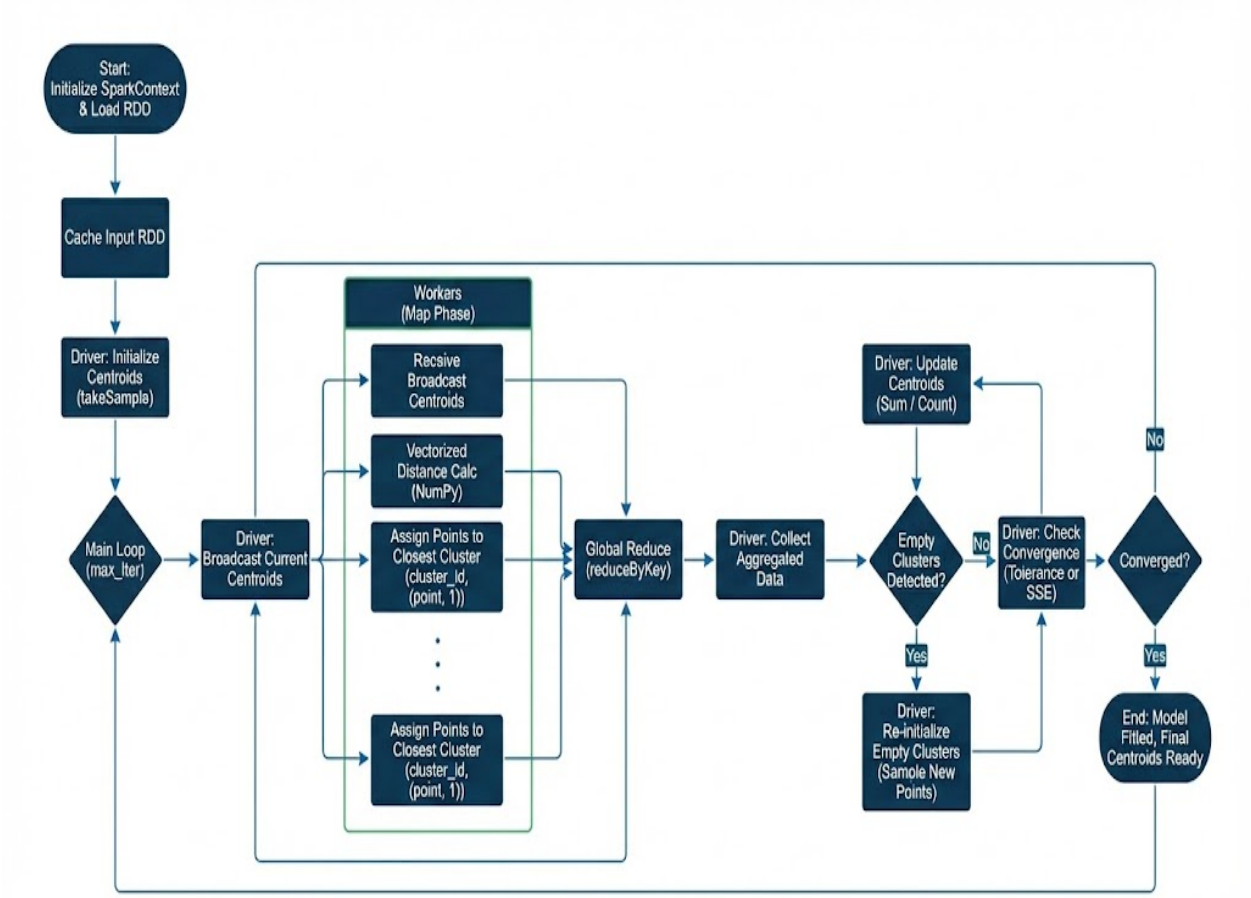
## Performance:

Near-linear scaling observed with 4 partitions.

## Deviation:

No deviation from expected behavior. All tests passed successfully.

## Architecture Diagram:


Start: Initialize SparkContext & Load RDD
Cache Input RDD
Driver: Initialize Centroids (takeSample)
Main Loop (max_Iter)
Driver: Broadcast Current Centroids
Workers (Map Phase)
Recsive Broadcast Centroids
Vectorized Distance Calc (NumPy)
Assign Points to Closest Cluster (cluster_id, (point, 1))
Assign Points to Closest Cluster (cluster_id, (point, 1))
Global Reduce (reduceByKey)
Driver: Collect Aggregated Data
Empty Clusters Detected?
Yes
Driver: Re-initialize Empty Clusters (Samole New Points)
No
Driver: Update Centroids (Sum / Count)
Driver: Check Convergence (Toierance or SSE)
Converged?
No
Yes
End: Model Fitled, Final Centroids Ready

## Speedup Graph: