



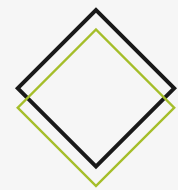
Massively Parallel Machine Learning

Parallel Implementation and Evaluation of K-Means Algorithm

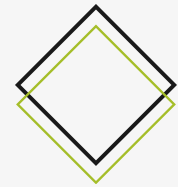
Nicola Cecere | Luca Petracca | Alessandro Rossi



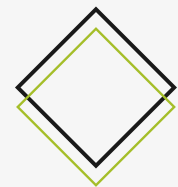
The Goal



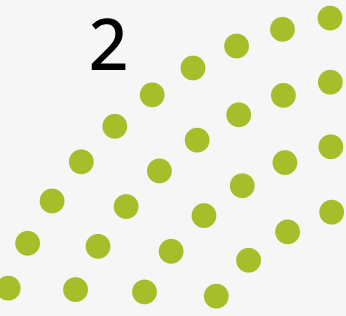
Project Goal: We explore the K-Means clustering algorithm on the MNIST dataset, and its application in a massively parallel processing environment using Spark and Python



Dataset: The MNIST dataset, with 70,000 diverse 28x28 grayscale images of handwritten digits, is widely used for testing clustering algorithms




Process: readFile, assign2cluster, kmeans, testing



Algorithm Implementation



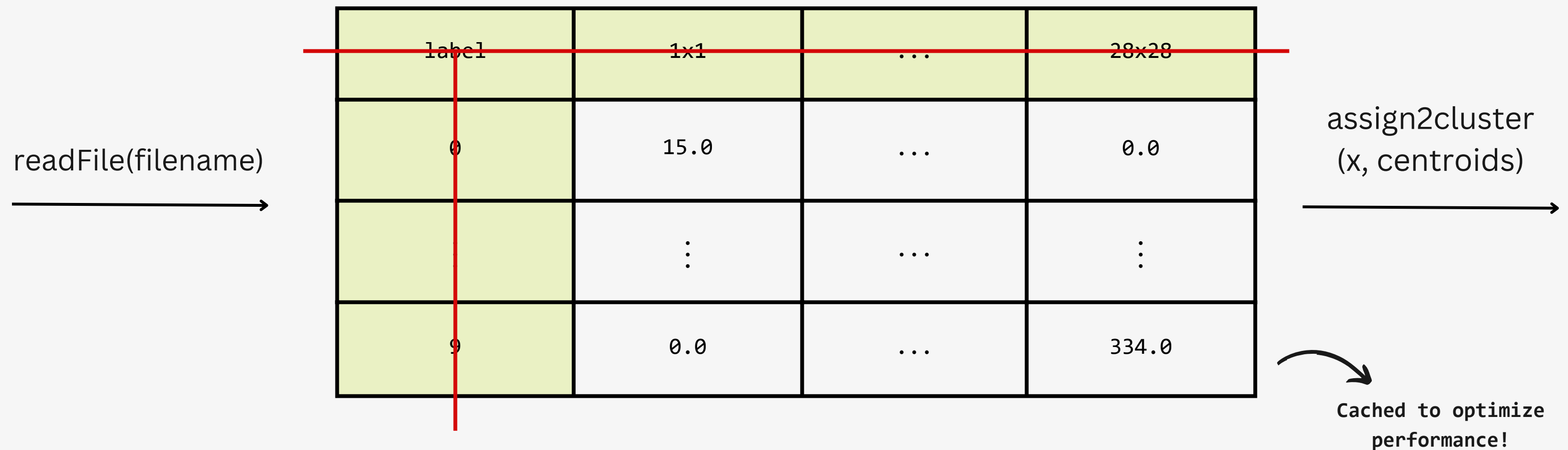
readFile: centralized

1. The serialReadFile function reads data from a CSV file into a Pandas DataFrame, using the 'pd.read csv' method.
 2. It then removes a column named "label" from this DataFrame, modifying it in place.
 3. The function returns the modified DataFrame, now without the "label" column.
- 

readFile: parallelized

1. Reads the specified CSV file into an RDD using the textFile method.
2. Applies two main operations to the RDD:
 - Filter out Header: Utilizes the flatMap operation with the is header function to remove the header row, identified by a known feature such as 'label'.
 - Convert to Floats: Transforms each line into a list of floats, excluding the label column, through the map operation with the convert to float function.
3. Returns the processed RDD, where each data point is now a list of floats representing the features, excluding the label.

readFile: parallelized



Red: rows and columns to be removed

kmeans: centralized

1. Converts the input data X to a NumPy array.
2. Initializes K centroids using the initialize centroids function, which generates centroids from a standard normal distribution based on the number of features in X .
3. Initialize a list of clusters, with each cluster corresponding to one centroid.
4. Iterates over a specified number of iterations (n iter):
 - For each sample in X , identifies the closest centroid using the serialAssign2cluster function and assigns the sample to the respective cluster.
 - Updates each centroid to be the mean of all samples assigned to it. If a cluster is empty, reinitializes its centroid randomly.
5. Returns the final centroids after completing all iterations.

kmeans: parallelized

1. Initializes K centroids using the initialize centroids function. This function generates centroids from a standard normal distribution, considering the number of features in the RDD X.
2. Iterates over the specified number of iterations (n iter), performing the following steps in each iteration:
 - Broadcasts the current centroids to all nodes in the cluster.
 - Assigns each data point in X to the nearest centroid using the parallelAssign2cluster function and a map operation.
 - Aggregates data points assigned to each centroid and updates the centroids' positions. This is done using the reduceByKey and map operations, where the latter applies the averageCentroid function to calculate the new centroid position.
 - Collects the updated centroids back to the driver node.
 - If the number of collected centroids is less than K, additional centroids are initialized to maintain the total number of K centroids.
3. Returns the final centroids after completing all iterations.

kmeans: parallelized

map
→

centroid	sample	count
[c_0,1 ... c_0,784]	[x_1 ... x_784]	1
⋮	⋮	⋮
[c_k,1 ... c_k,784]	[x_1 ... x_784]	1

m rows

reduceByKey
→

centroid	sample	count
[c_0,1 ... c_0,784]	sum([x_1 ... x_784])	7534
⋮	⋮	⋮
[c_k,1 ... c_k,784]	sum([x_1 ... x_784])	4226

k rows

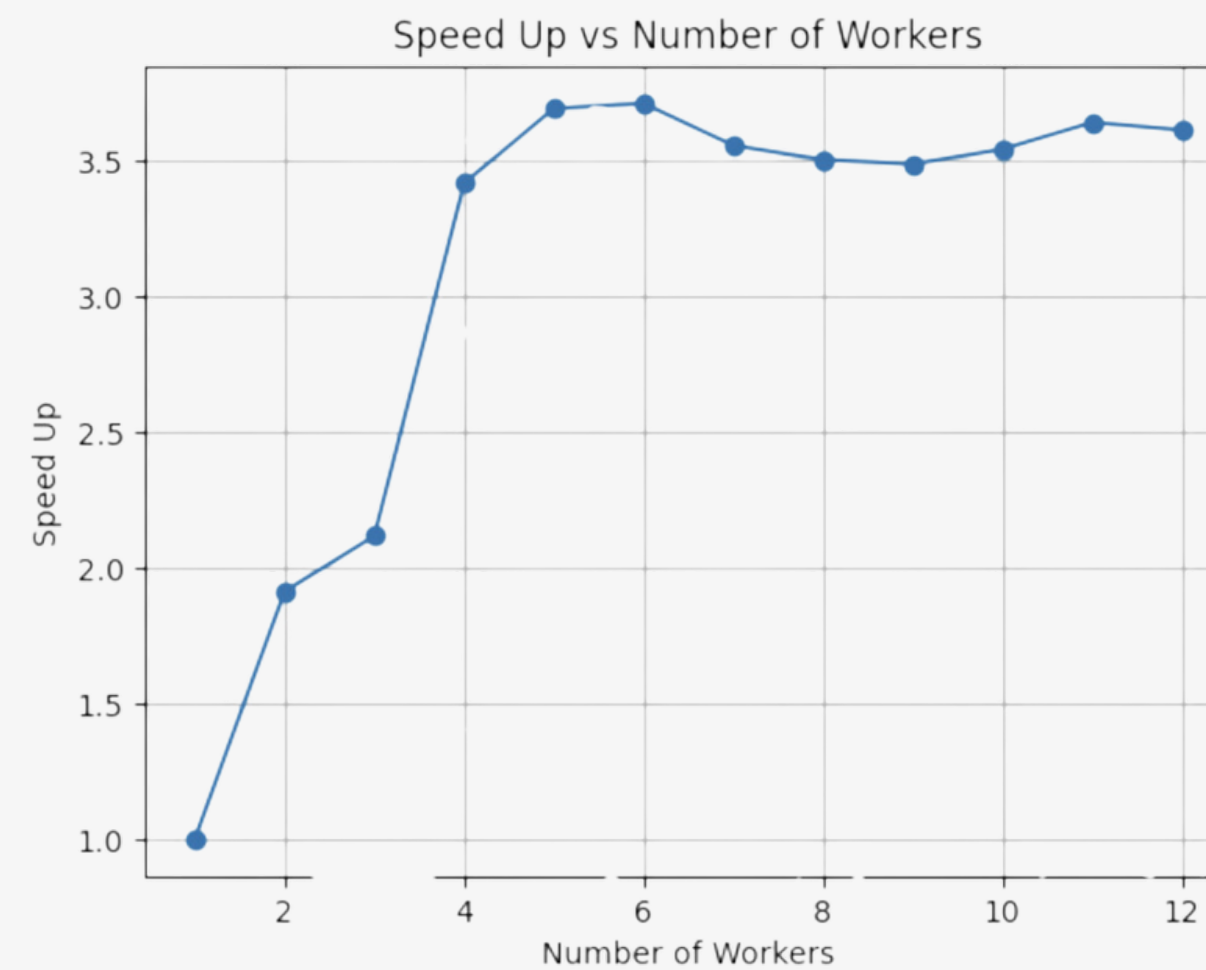
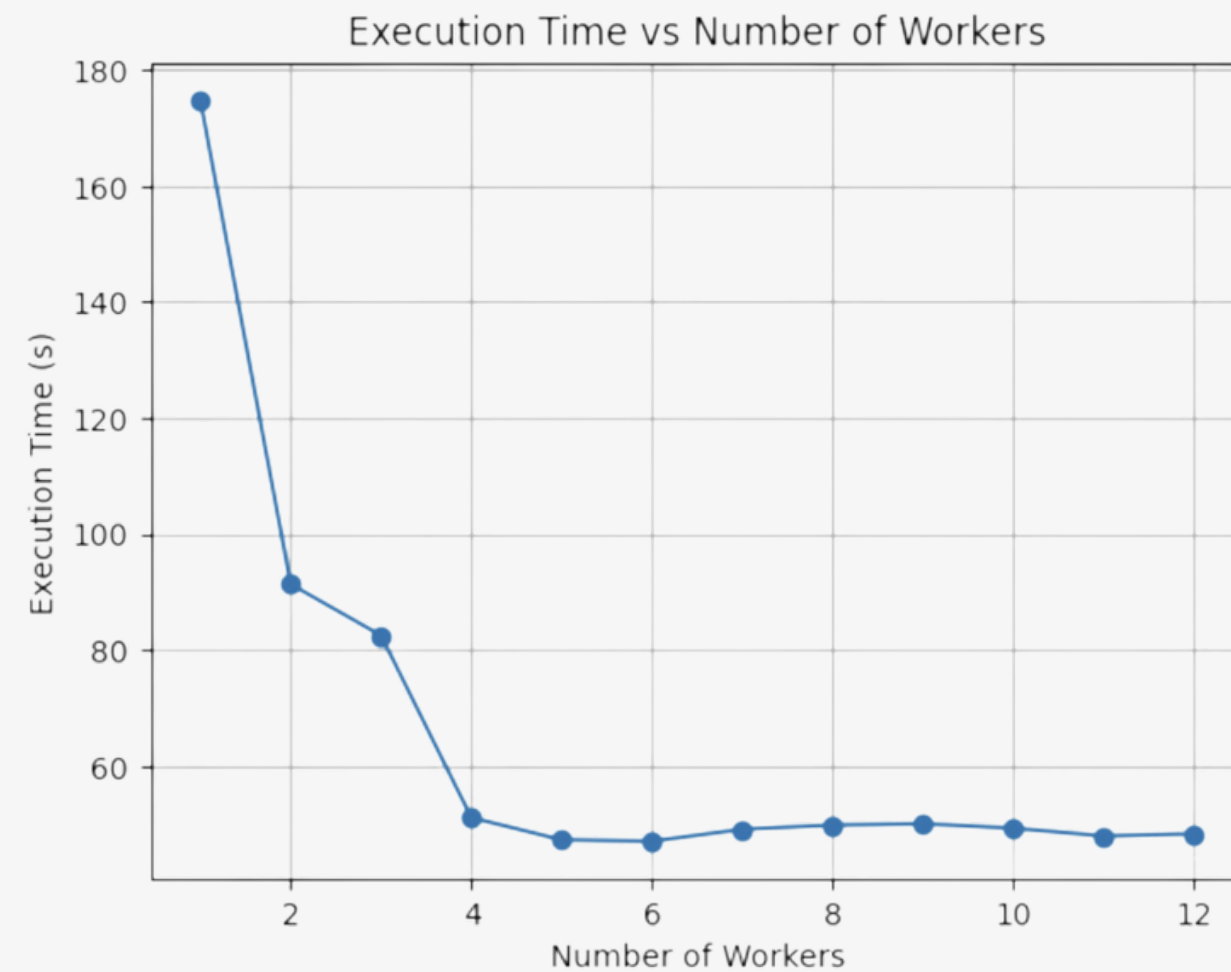
assign2cluster: serial/parallel

1. Initializes the minimum distance as infinity and the index of the closest centroid as -1.
2. Iterates over each centroid in the given list of centroids:
 - Converts the current centroid to a NumPy array for computation.
 - Calculates the Euclidean distance between the data point x and the centroid.
 - Updates the minimum distance and the index of the closest centroid if the current distance is smaller than the recorded minimum.
3. Returns:
 - Centralized: the index of the centroid that is closest to the data point x
 - Parallelized: a tuple consisting of the closest centroid's index and a pair: the data point x (as a NumPy array) and the count.

Experiments

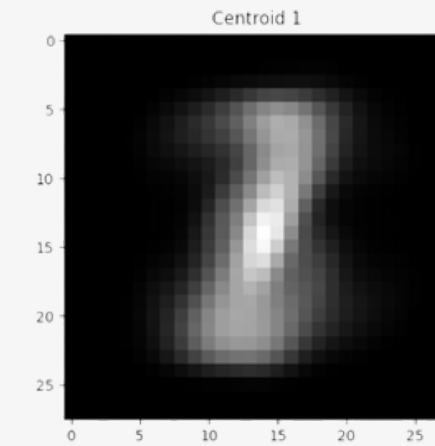
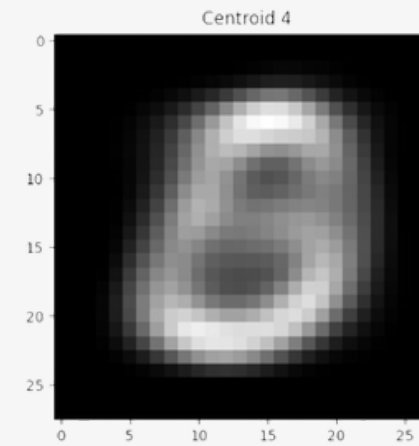
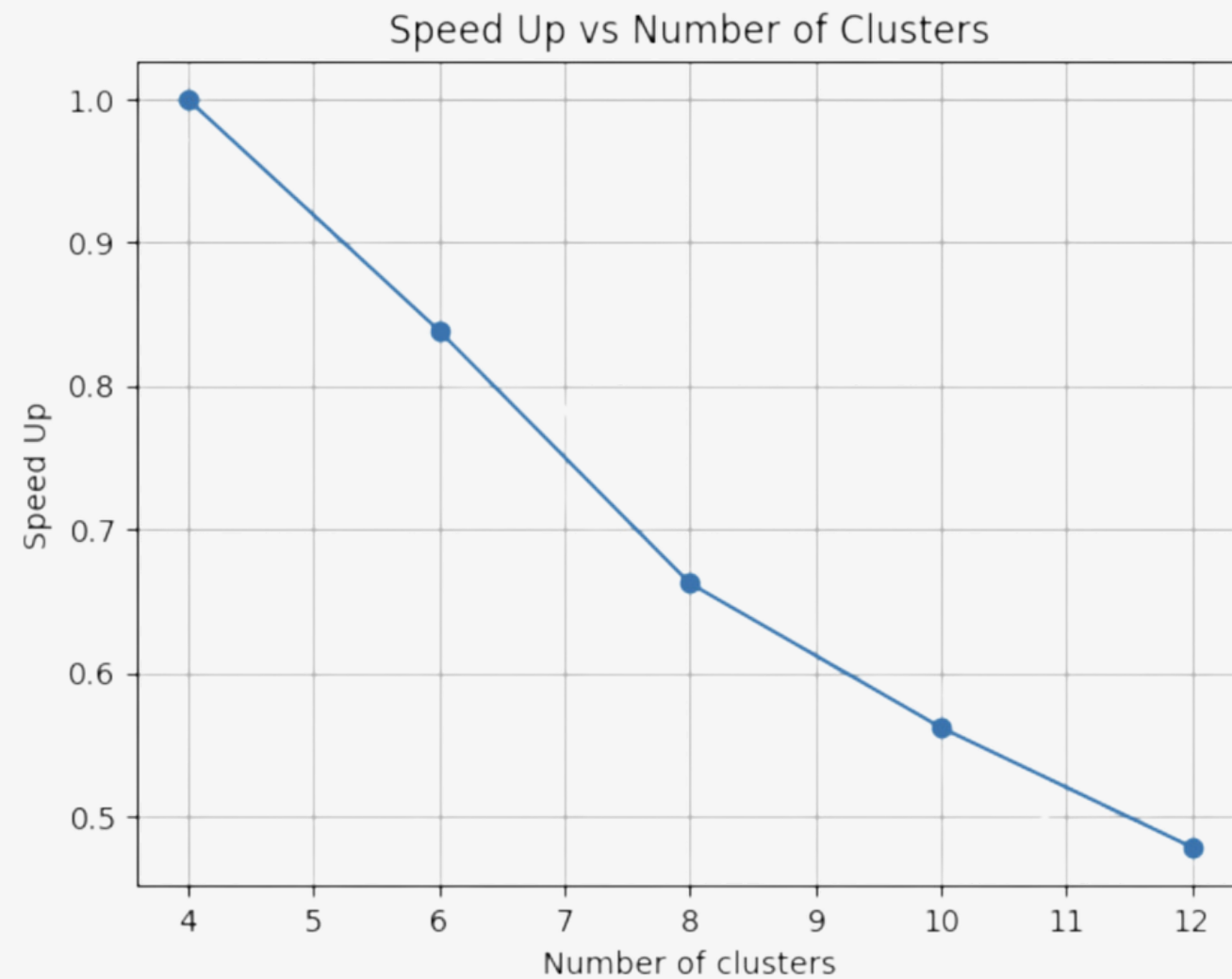
Different workers execution

1. Try the execution of the K-Means with different workers
2. From 1 to 12 workers (executed on a 10-core processor)

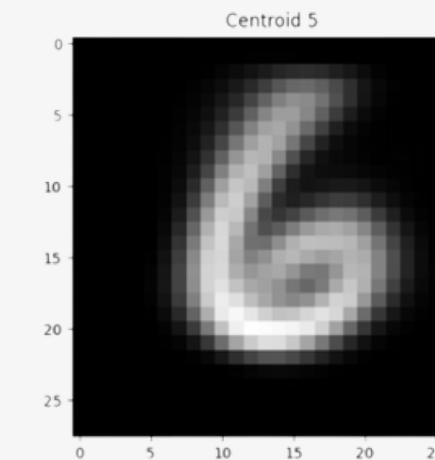
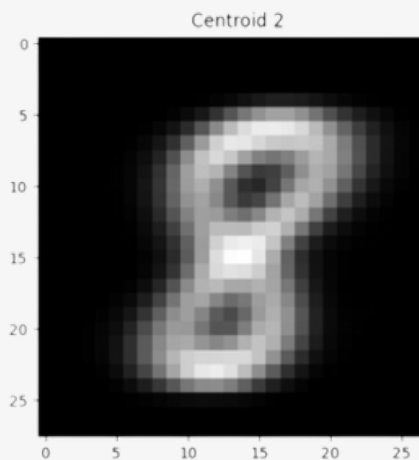


Changing number of cluster

1. Try the execution of the K-Means algorithm with different clusters
2. Execution with 4, 6, 8, 10, 12 clusters



4 clusters



12 clusters

Massively Parallel Machine Learning

Thank you!
Q&A?

Nicola Cecere | Luca Petracca | Alessandro Rossi

