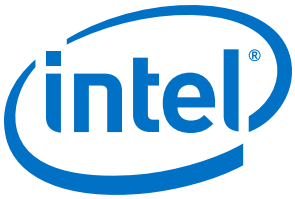


AN 856: K-Mean Clustering with the Intel® FPGA SDK for OpenCL™

AN-856 | 2018.06.12
Latest document on the web: PDF | HTML

Contents

1. AN 856: K-Mean Clustering with the Intel® FPGA SDK for OpenCL™

When you have a set of data where each data point represents a d-dimensional observation, clustering divides that data into groups (or clusters) that are more similar to each other.
Clustering algorithms are used often in many fields such as data mining, machine learning, and image processing. K-mean clustering is a popular clustering algorithm, but it is a very time consuming process.
This algorithm is applied to a data set without any label and is considered an unsupervised learning algorithm

K-mean clustering was developed and optimized for different hardware platforms, and in this application note we propose a design for FPGAs using the Intel® FPGA SDK for OpenCL™[1] [2].

The Intel FPGA SDK for OpenCL can help software engineers to develop and accelerate algorithms on higher efficiency platforms such as FPGAs much faster then coding in RTL.

In the following sections, we explain the k-mean clustering algorithm. Then, we discuss the OpenCL implementation of k-mean clustering. Finally, we provide some performance results comparing how the algorithm can be accelerated on an Intel Arria® 10 FPGA versus an Intel Xeon CPU.

1.1. K-Mean Clustering Algorithm

If you have a set of n observations or data represented with $\{x_1, x_2, \ldots, x_n\}$, and each data includes d features and no labels, k-mean clustering provides a simple method to group the whole data set into k clusters $\{c_1, c_2, \ldots, c_k\}$, where data in the same cluster are more similar to each other.

Each data (x_i) is labeled with the cluster $j\,(1 \ldots k)$, where the distance of the data point i to the centroid of cluster j is minimum compare to all other clusters.

The objective is $\min \sum_{i=1}^{k} \sum_{x \in C} \|x - \mu_i\|^2$, where μ_i is the mean of the data points in cluster i.

---

[1] OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of the Khronos Group™.

[2] The Intel FPGA SDK for OpenCL is based on a published Khronos Specification, and has passed the Khronos Conformance Testing Process. Current conformance status can be found at www.khronos.org/conformance.

Different methods for initializing this unsupervised clustering algorithm exist, and this initialization is important for reducing the complexity of the algorithm (the number of iterations required) and the results of clustering (local optimum).

Typically, if the initial centroids are closer to the final centroid, you can achieve better clustering results and lower complexity. This application note does not cover initialization methods.

In this example, we pick k data as the centroids of the clusters randomly:

1. Pick k data points from the input data set as centroids for k clusters.
2. Find the distance of each data into all centroids and label the data with the cluster that has the minimum distance to its centroid.
3. Calculate the new centroid of all clusters as a mean of the labeled data from the previous step.
4. Check the error.

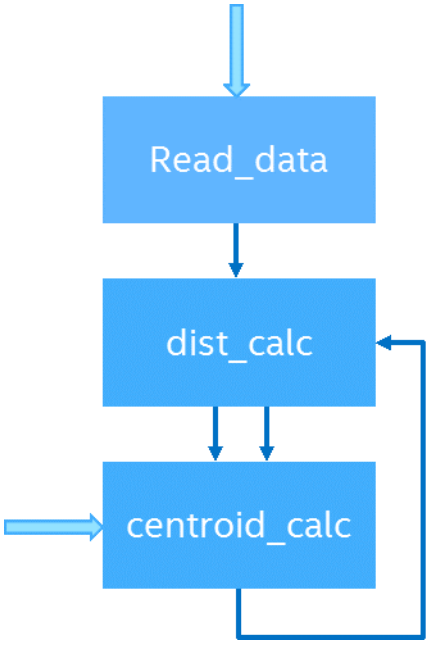
   If the centroids are no longer changing, the algorithm has converged, and you can exit the algorithm. Otherwise, repeat steps 2- 4.

1.2. OpenCL Design

K-mean clustering algorithm is very time consuming algorithm because it goes through all of the data set and finds the distance of each data to centroid of all data. For calculations like these, FPGAs are great target. The Intel FPGA SDK for OpenCL provides a faster solution for you to benefit the flexibility of FPGA platforms and accelerate the clustering process for huge number of unlabeled input data.

In this section, we take the previously described algorithm, and implement it as a series of OpenCL kernels. Initial centroids for k clusters are chosen on the host code and then passed into an OpenCL kernel running on an FPGA. Therefore, you can modify the host code with a method that you like to use to initialize the centroids. All the data is passed to the initial kernel using global memory transfer.

The following diagram shows the data flow between the kernels.



The first kernel (`Read_data`) reads input data from global memory and saves it locally for the iterations of the algorithm that follow. Through an internal channel, this data is passed to next kernel, which is an autorun kernel, in packets of 16 input data.

The next kernel (`dist_calc`) calculates the distance of each data point to all centroids and labels the data in the cluster with the minimum distance. It also calculates the sum of all the data labeled in the same cluster.

The generated data at this level is passed to the next kernel (`centroid_calc`) through an internal channel. In this kernel, the sum of data in each cluster is read from the internal channel and added until all of the data is processed. Then, the new centroids are calculated as an average of all of the data categorized in each cluster.

If the distance of the new centroids to the previous one is less than the acceptable error, the data was successfully clustered and the kernel exits.

If the error is not acceptable, the new centroids are sent to the `dist_calc` kernel again through an internal channel for the next iteration.

1.3. Performance Results

We compared the performance of this implementation on an FPGA with the optimized k-mean implementation on a CPU. For both FPGA and CPU runs, the same data set was used.

The FPGA used for performance comparison was an Intel Arria 10 GX FPGA Development Kit. The FPGA was programmed with Intel FPGA SDK for OpenCL Version 17.1 Update 1. During testing, the FPGA had an f_{MAX} of 320 MHz.

The CPU used for performance comparison was an Intel Xeon® E5-2680 (24 cores, no hyperthreading).

The following table shows the time to converge on acceptable clusters for data with various data sizes.

Data Size (bytes)	FPGA		CPU
	Time with initialization method 1 (ms)	Time with initialization method 2 (ms)	Time (ms)
512	0.028	0.016	0.065
1024	0.042	0.032	0.573
2048	0.051	0.037	0.627
4096	0.089	0.039	0.804
8192	0.105	0.044	0.919

In this experiment, the number of clusters are set to 10.

Each data set includes 2 features of floating type and different numbers of input data sets (512 to 8192) are used to compare the performance of FPGA and CPU.

For the FPGA runs, we tried two initialization methods. In the first method, we used the first k-data as the centroids of the clusters. In the second method, we chose centroids randomly. With randomly-chosen initial centroids, the algorithm required fewer iterations and therefore achieved faster times.

1.4. Document Revision History for AN 856: K-Mean Clustering with the Intel FPGA SDK for OpenCL

Document Version	Changes
2018.06.12	Initial release.