Three is not a crowd: A CPU-GPU-FPGA K-means implementation

Marcos Canales¹, Jorge Cáncer¹, Denisa Constantinescu², Carlos Escuín³ and Borja Pérez⁴

¹University of Zaragoza, Spain
²University of Málaga, Spain
³Polytechnic University of Catalonia, Spain
⁴University of Cantabria, Spain

Introduction

- **Clustering** is the task of assigning a set of objects into groups (clusters) so that objects in the same group are more similar to each other than to those in other groups.
- **K-means** is a clustering algorithm that calculates the cluster with the nearest mean for each object. To achieve this, it uses a function like Euclidean or Manhattan distance.

Our objective is to exploit our heterogeneous computing environment, that integrates an Intel Core i7-6700K chip, 2x NVIDIA TITAN X and an Intel Altera Terasic Stratix V DE5-NET FPGA, to run K-means as fast as possible.

Algorithm:

Input: \( K \) (number of clusters), set of \( N \) points with \( D \) dimensions
Output: partition of \( N \) points in \( K \) clusters

1. Place centroids \( c_1, c_2, ... , c_K \) at random locations
2. Iterate until convergence condition is met
3. For each point \( x_i \), \( i = 1..N \):
   4. For each cluster \( c_j, j = 1..K \):
   5. Calculate distance to \( c_j \), given all \( D \) dimensions
   6. Assign the membership of point \( x_i \) to nearest cluster \( j \)
   7. For each cluster \( c_j, j = 1..K \):
   8. Centroid \( \bar{c}_j = \text{mean of all points whose membership is } j \)

\[ O(\text{iterations } \times N \times D \times K) \]

Implementation

We adapted the k-means algorithm to be executed under five possible configurations, so we can then compare them:

- Sequential (single core)
- OpenMP (multi core)
- OpenMP & OpenCL for GPUs
- OpenMP & OpenCL for FPGAs
- OpenMP & OpenCL for GPUs and a FPGA (Fig. 1)

Results

![Figure 2. Speed-up comparison using N=65536 with respect to sequential version](image)

![Figure 3. Speed-up comparison in 2xGPUs version regarding the number of points](image)

Conclusions & future work

- When processing small datasets **OMP** is the best choice, since delegating on other devices is not worth it because of high I/O time (Fig. 1).
- **GPUs** achieve big parallelism, so they are the best choice when it comes to dealing with large datasets (Fig. 1 & Fig. 2).
- When looking for performance, **FPGA** is not suitable for k-means, due to the fact that it’s hard to get advantage of all its resources pipelining the algorithm (Fig. 1).
- An **integrated GPU & FPGA** version would be interesting to be developed in order to further analyze the tradeoff between the overall execution time and the power usage.
- Devices in this kind of environments have many different features (e.g. computing power). A **dynamic load balancer** could be introduced to split the workload depending on specific criteria, like execution time of previous iterations for each device.

References