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

Marcos Canales[1], Jorge Cáncer[1], Denisa Constantinescu[2], Carlos Escuín[3] and Borja Pérez[4]

[1]University of Zaragoza, Spain
[2]University of Malaga, Spain
[3]Polytechnic University of Catalonia, Spain
[4]University 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:

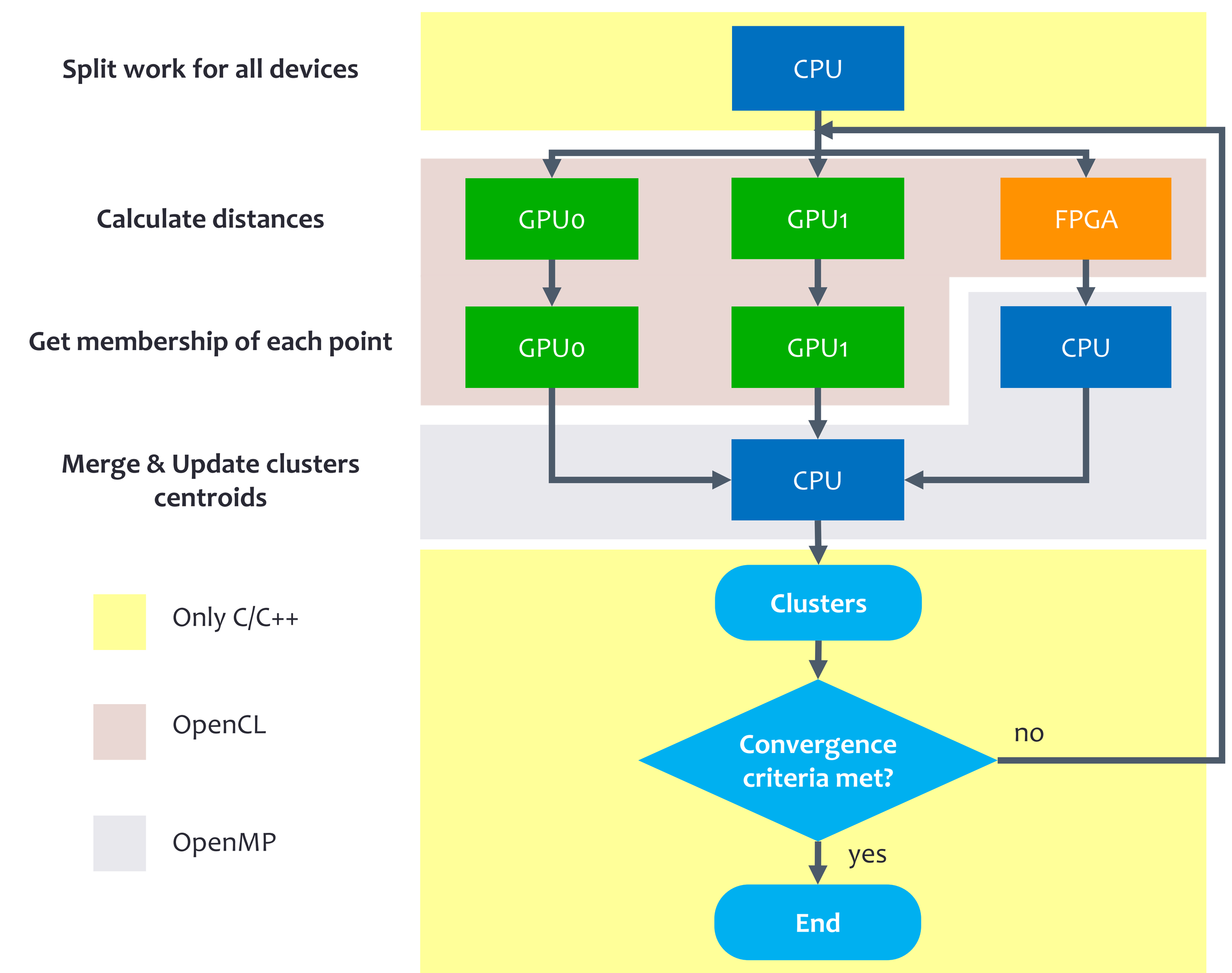
```
Input: K (number of clusters), set of N points with D dimensions
Output: partition of N points in K clusters
1. Place centroids c1, c2, …, cK at random locations
2. Iterate until convergence condition is met
3.   For each point xi, i=1..N:
4.     For each cluster cj, j=1..K:
5.       Calculate distance to cj, given all D dimensions
6.       Assign the membership of point xi to nearest cluster j
7.   For each cluster cj, j=1..K:
8.       Centroid cj = mean of all points whose membership is j
```

$O(\#iterations \times N \times D \times K)$

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 FPGA
- OpenMP & OpenCL for GPUs and a FPGA (*Fig. 1*)



Results

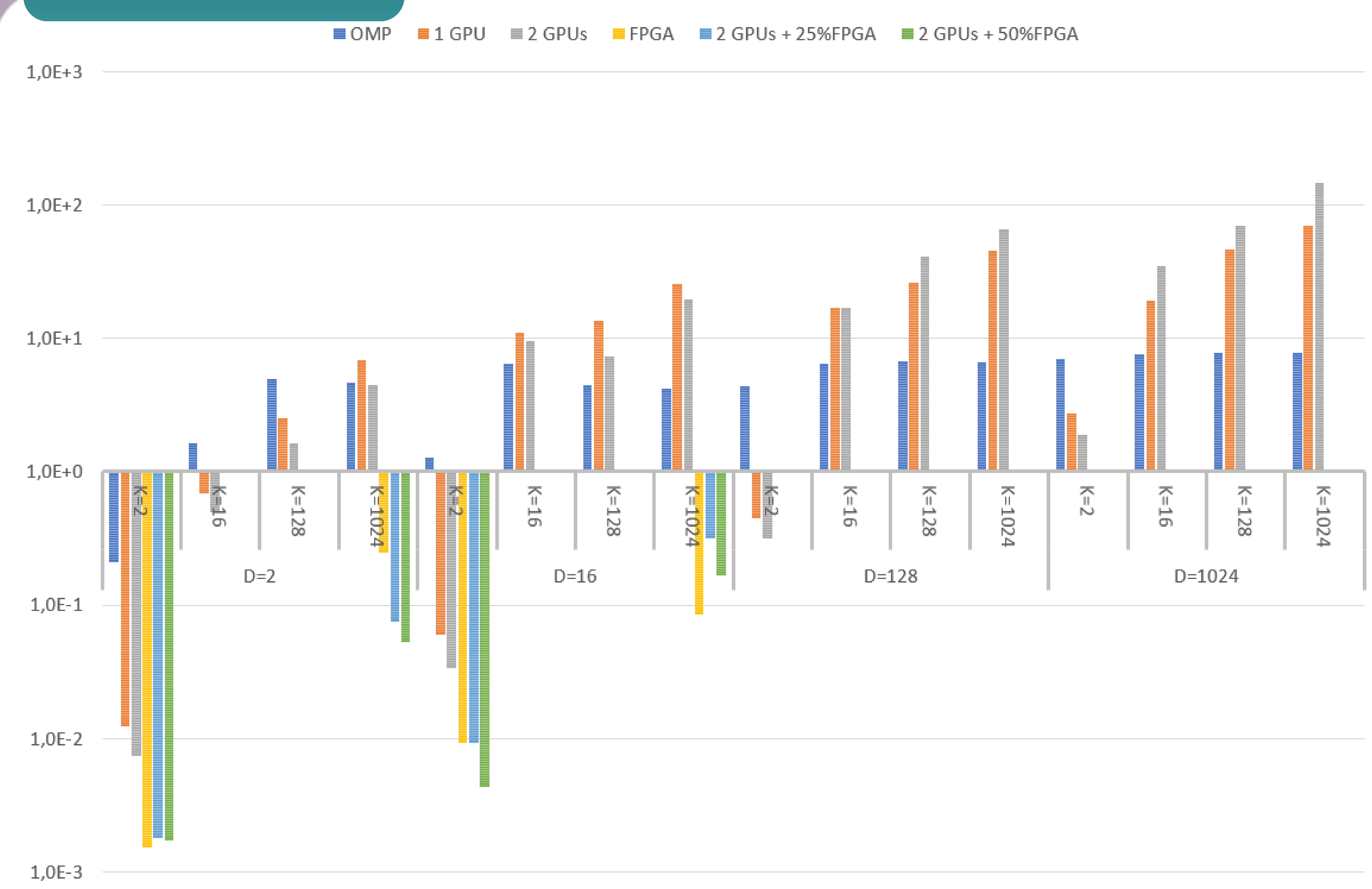


Figure 2. Speed-up comparison using N=65536 with respect to sequential version

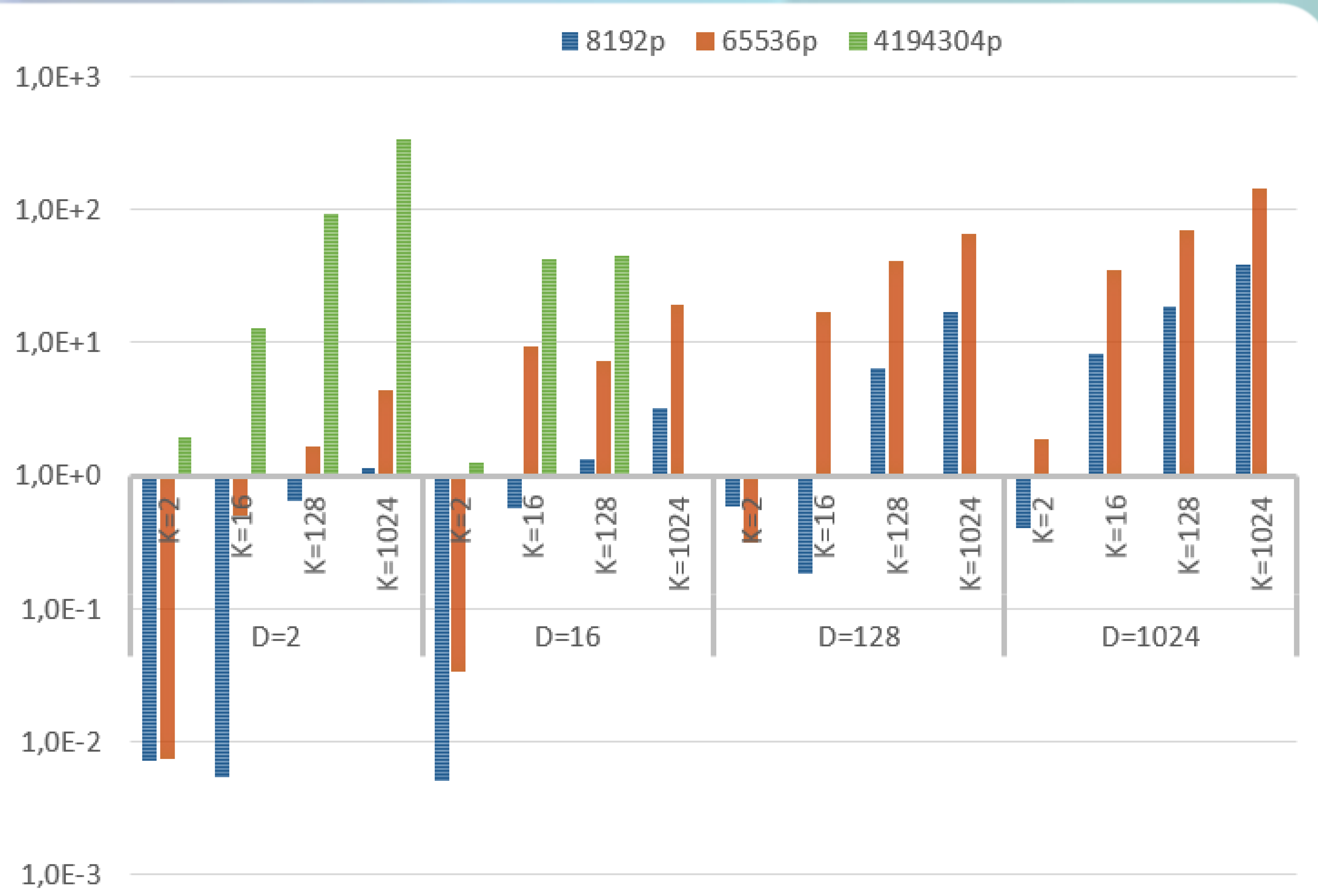


Figure 3. Speed-up comparison in 2xGPUs version regarding the number of points

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

1. Tang, Q. Y., & Khalid, M. A. (2016). Acceleration of K-Means Algorithm Using Altera SDK for OpenCL. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 10(1), 6.
2. K-Means. Rodinia. Retrieved April 23, 2017, from www.cs.virginia.edu/~skadron/wiki/rodinia/index.php/K-Means
3. OpenMP. Retrieved April 23, 2017, from www.openmp.org/
4. Khronos Group. OpenCL - The open standard for parallel programming of heterogeneous systems. Retrieved April 23, 2017, from www.khronos.org/opencl/
5. Intel. Intel FPGA SDK for OpenCL. Retrieved April 23, 2017, from www.altera.com/en_US/pdfs/literature/hb/opencl-sdk/aocl-best-practices-guide.pdf

HiPEAC CSW 2017 | 27 – 28 April | Zagreb, Croatia