GPU for Scientific Computing

-Ali Saleh

Contents

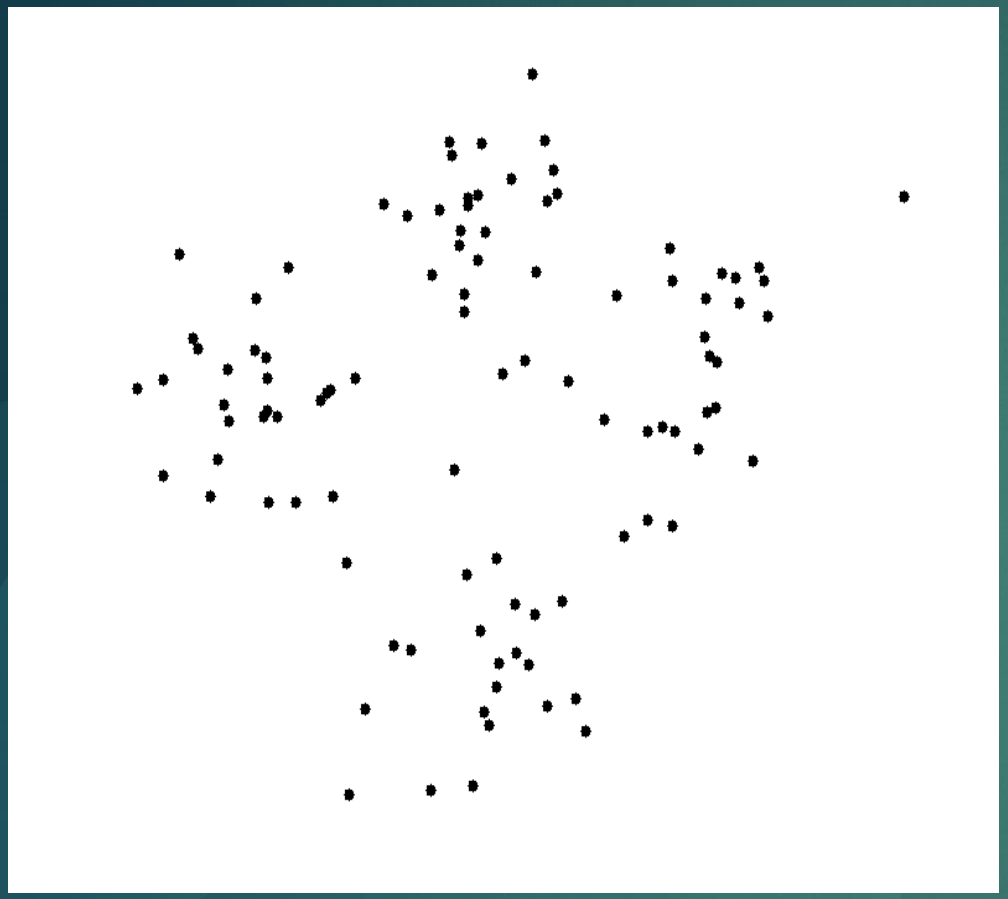
What is GPU

- Graphics Processing Units
- Several Small Cores
- Optimized for memory intensive workload, Geometric Calculations , and fast rendering
- High support for parallelization
- Perfect as Stream Processor

K-Means Clustering

- Heuristics solution to the NP-Hard clustering problem
- The common method work as following:
  - Create an initial seed clusters
  - Iteratively assign each element to a cluster that minimizes your selected criteria (cluster mean for K-means)
  - Repeat till convergence reached (no new assignment or No of iteration reached)

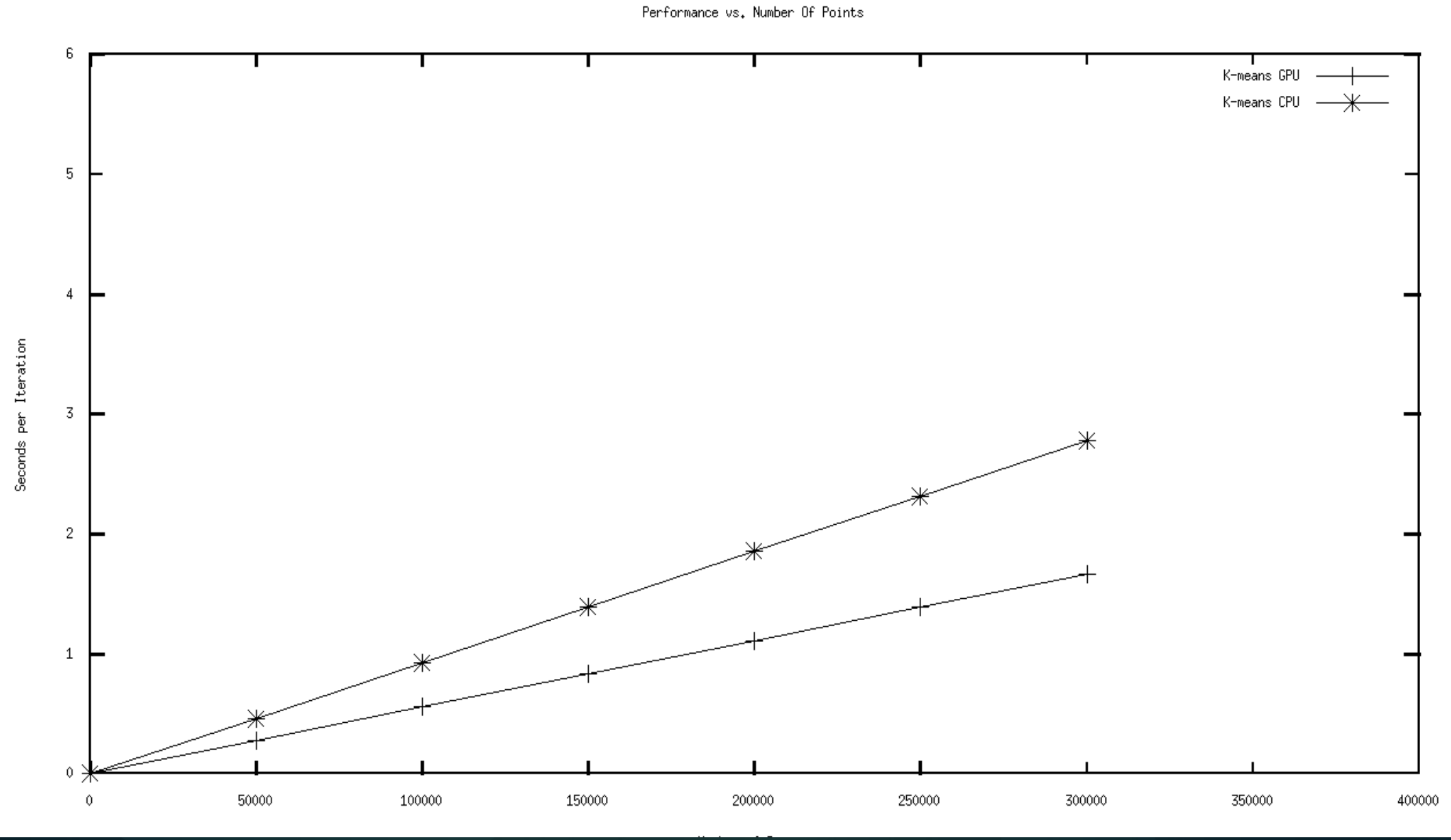
K-Means Clustering



K-Means Clustering

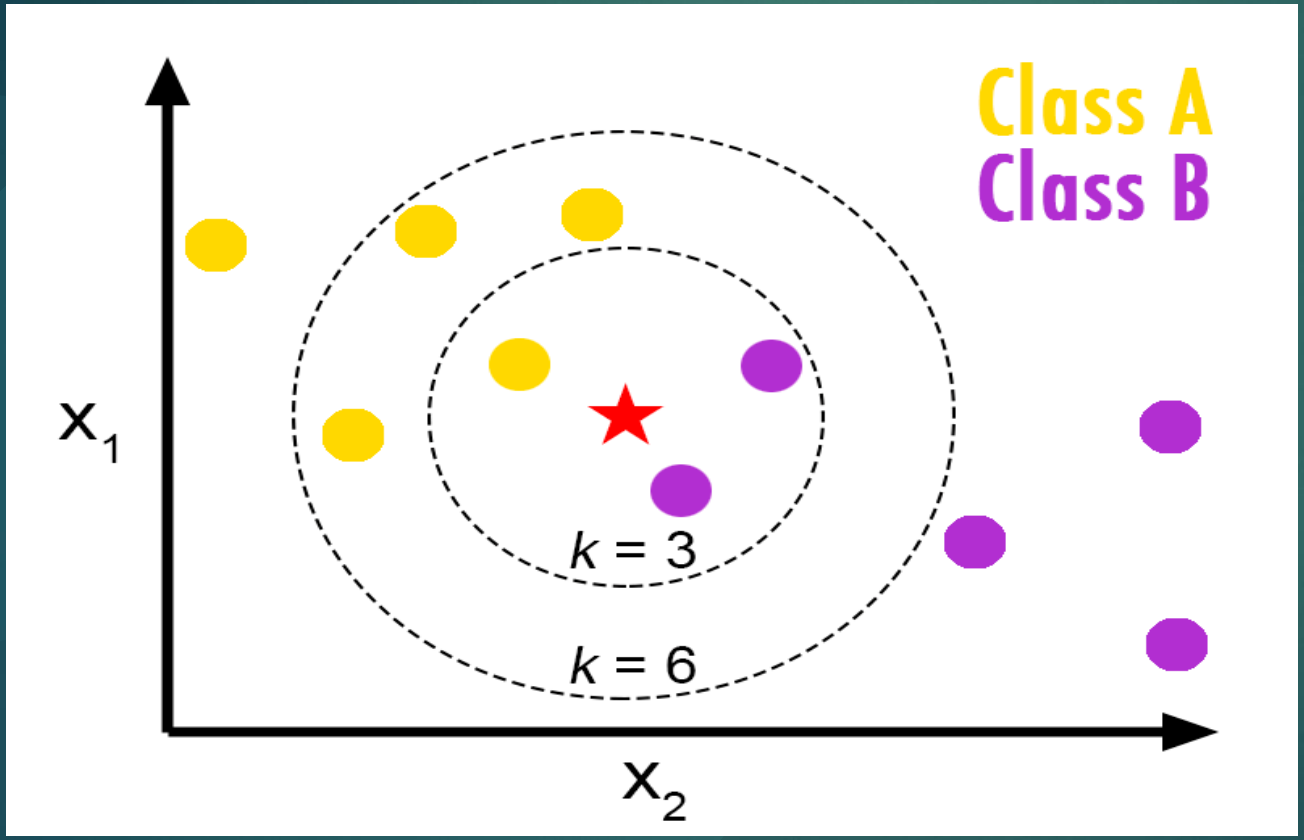
- Metric calculation per point is independent .
- A GPU Implementation involves
  - Copy the whole data for processing once (it never changes)
  - Iterate over points in parallel and calculate the clustering metric per point
  - Copy the data back to the main memory and use CPU to create the new clusters
  - Copy the clusters info back to the GPU memory
  - Repeat till convergence

K-Means Clustering



K-Nearest Neighbours

- k closest training examples in the feature space
- Majority vote for classification, values average of regression
- Distance measurement can be Euclidean distance



K-Nearest Neighbours

- Generate probability image by pixel classification of new mammograms
- Training feature space 200M data point 4 minutes per image
- Clustering the Training feature space reduced it to 50M 1.65 minute per image
- GPU implementation for KNN along with NVIDIA Quadro FX 1700M with 32 GPU Cores
- On Average 12 Seconds only per image

When to use GPU and when not

- Researchers tend to optimize their GPU implementation much more than a CPU one
- Cache bandwidth for a CPU is much better than for a GPU
- This may lead to un-utilized GPU waiting for reusable data to be fetched from memory
- A multipass algorithms will need accumulating results which is not easy in GPU right now.
- Better at calculation intensive tasks

Commercial GPU Programming Platforms

- CUDA
  - Only NVIDIA hardware
  - Comes as one development package from NVIDIA
  - Better Math library built-in
  - Better Marketing strategy
  - NVIDIA hardware is more expensive
  - Multiple higher language wrapper available
  - Better developer support by NVIDIA

- OpenCL
  - The ultimate standard
  - Wider range of processor architectures support
  - Supplied by multiple vendors
  - Harder to debug
  - Multiple higher language wrapper available

References

- {Data Visualization and Mining using the GPU} Sudipto Guha, Shankar Krishnan ,Suresh Venkatasubramanian
- Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU : Victor W Lee et al
- GPU Acceleration of Iterative Clustering : Jesse D. Hall ,John C. Hart
- Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication : K. Fatahalian, J. Sugerman, and P. Hanrahan
- k-nearest neighbour search: fast GPU-based implementations and application to high-dimensional feature matching : Garcia and E. Debreuve and F. Nielsen and M. Barlaud
- Intra-operative Real-Time 3-D Information Display System based on Integral Videography : Hongen Liao et al
- On detecting abnormalities in digital mammography: Waleed et al.

References

- http://en.wikipedia.org/wiki/Graphics_processing_unit
- http://www.nvidia.com/content/global/global.php
- http://wiki.tiker.net/CudaVsOpenCL
- http://streamcomputing.eu/blog/2011-06-22/opencl-vs-cuda-misconceptions/

Thanks

Questions?