Hybrid KNN Algorithm using CPU and GPU applied on 3D data

Exequiel Sepúlveda (esepulve@alges.cl)
Felipe Muñoz (fmunoz@alges.cl)
ALGES laboratory, Advanced Mining Technology Center (AMTC), University of Chile
Department of Mining Engineering, University of Chile

The k-th nearest neighbour problem for 3D data has been widely studied, nevertheless, the surge of using GPU (Graphical Processing Unit) as general-purpose computing units opens up the need to design and implement new algorithms, that allow us to get results more rapidly than using conventional algorithms.

The main advantage of GPU is its great computing capacity for parallel computing. This is due to an architecture that contemplates a great number of processing cores originally designed to graphics processing and that can currently be used for other purposes such as high performance cientific calculation.

On the other hand, natural trends of processing everything with the GPU makes the CPU to be idle while it waits for results from the GPU. That is why a hybrid use of CPU as well as GPU is proposed to resolve the problem.

The KNN problem is defined as follows: let R be a set of reference points, of size N, we wish to find for each element of a set of search points Q, of size M, the K points closer to a distance *d*.

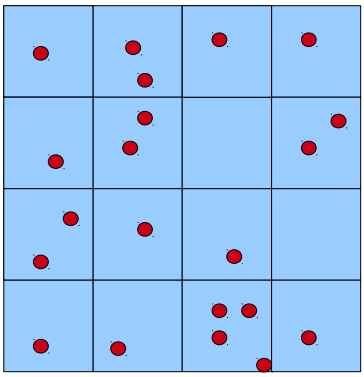
The brute force algorithm that resolves this, has a M*(N + N LOG(N) + LOG(N) + K) complexity. On the other hand, one of the most efficient means to realize this is the use of KDTree, that allows the problem to be resolved with a M*K*LOG(N) complexity.

Our method has three steps, pre-process, Cpu process and Gpu process.

Pre-Process:

1. Generate a grid bounding the data set.
2. Calculate the number of points per sector.

Grid



Index Matrix

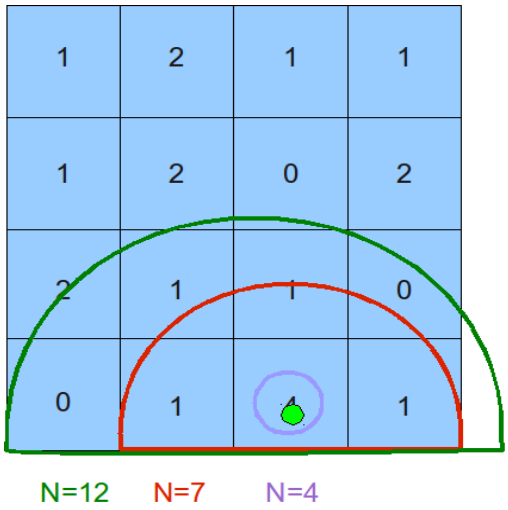
1	2	1	1
1	2	0	2
2	1	1	0
1	1	4	1

This step is executed only one time at the beginning of the algorithm.

Cpu Process:

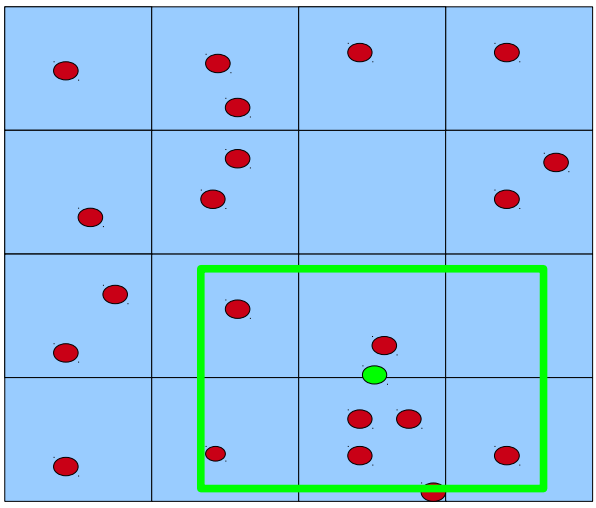
1. For each query, determine a radius *r* that contents at least K points. Using an incremental search.
2. Use the minor value between *r* and *d*
3. Store the array of radius.

Incremental Search



Gpu Process:

1. In Parallel: for each query and radius r, discard points that have at least one coordinate more distant than r to the respective query coordinate.
2. Using the remaining points. Calculate KNN using a brute force technique.



Brute Force:

1. For each query, calculate all distances between query in Q and points in R.
2. Sort by distance and obtain a set of K elements with minimal distance

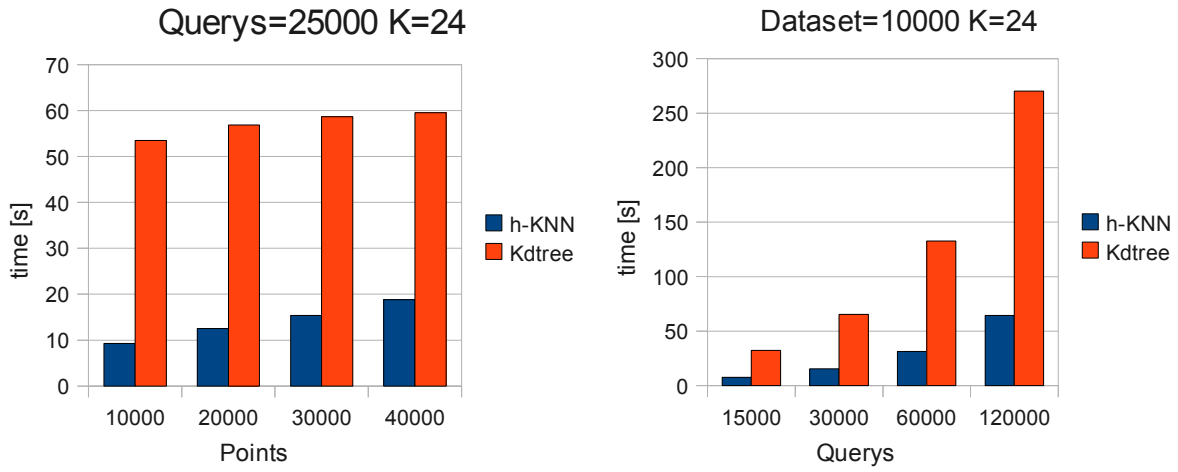
In example , for a given set of four queries, {q1,q2,q3,q4}, the algorithm will be executed in this way:

- Pre-Process
- Step 1
  - Cpu(q1)
- Step 2
  - Cpu(q2)
  - Gpu(q1)
- Step 3
  - Cpu(q3)
  - Gpu(q2)
- Step 4
  - Cpu(q4)
  - Gpu(q3)
- Step 5
  - Gpu(q4)

On every step Cpu and Gpu process are executed concurrently.

Results:

A NVIDIA Quadro Fx 570 GPU with 256MB of dedicated memory with 16 processing cores and an Intel Core 2 duo 1.86 Ghz were used. PyCUDA as programming API of such GPU. The h-KNN algorithm as well as brute force were implemented in Python and Numpy. Results were compared against the scipy KDTree implementation (using Numpy).



Hybrid KNN method is faster than Kdtree for a large set of queries and reference 3D points. In general h-KNN has a 4x speedup.

References

1. D. M. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching, http://www.cs.umd.edu/~mount/ANN/.
2. Sameer A. Nene and Shree K. Nayar. A Simple Algorithm for Nearest Neighbor Search in High Dimensions. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 9, SEPTEMBER 1997.*
3. Vincent Garcia, Eric Debreuve, Michel Barlaud. Fast k Nearest Neighbor Search using GPU.
4. SciPy and NumPy. http://www.scipy.org

Paper Title:
Hybrid KNN Algorithm using CPU and GPU applied on 3D data

Authors information:
Exequiel Sepúlveda (esepulve@alges.cl) – Research Scientist
Felipe Muñoz (fmunoz@alges.cl) – Undergraduate Student
ALGES laboratory, Advanced Mining Technology Center (AMTC), University of Chile
Department of Mining Engineering, University of Chile
Address: Av. Tupper 2069, Santiago, Chile
Phone: +56 (2) 978 4585
Fax: +56 (2) 978 4985