TKSimGPU: A Parallel Top-K Trajectory Similarity Query Processing Algorithm for GPGPUs
Eleazar Leal, Le Gruenwald, Jianting Zhang and Simin You

Abstract
We propose TKSimGPU, an algorithm that incorporates parallelization strategies in order to answer top-K trajectory similarity queries. We ran experiments comparing the throughput of top-K trajectory similarity queries performed on multicore CPUs and GPGPUs against a naïve GPU implementation using a large-scale real world trajectory dataset. The experiments show that TKSimGPU achieves a 3.37x speedup in query processing time over exhaustive search on a GPU, and a 4.9x speedup in query processing time on a 12-core CPU architecture.

Trajectory Similarity Queries
Hausdorff distance:
Given two finite sets of points P and Q, the Hausdorff distance between P and Q is defined as
\[ \text{Hausdorff}(P, Q) = \max \left\{ \max_{p \in P} \min_{q \in Q} d(p, q), \max_{q \in Q} \min_{p \in P} d(p, q) \right\} \]
where \(d(p, q)\) is the Euclidean distance between points \(p\) and \(q\).

Example of the Hausdorff distance between two trajectories

Top-K Trajectory Similarity:
Example of a top-K trajectory similarity query with \(K = 2\)

Possible Applications:
- Understanding disease transmission between birds, and how they make use of space
- Online trajectory sharing services

TKGPU Algorithm

Estimate a Near-join Epsilon
Step 1: Draw random samples from P and Q, and for every \(p\) in the P-sample, calculate the average Hausdorff distance to the K-th nearest trajectory

Perform a Near-join Query
Step 2: Expand the MBRs of the P trajectories by epsilon
Step 3: Find the cells that every expanded MBR intersects
Step 4: Find the pairs \((p, q)\) such that they both go through a same cell

Results
Impact of the query set size on average query execution time
Impact of the database size on average query execution time
Impact of \(K\) on average query execution time

Experimental Environment
Dataset: The GeoLife dataset containing 17,261 trajectories whose lengths add up to 1,251,654 Km, span 48,203 hours, and 23,667,828 points.
Hardware: Workstation with 2 Intel Xeon E5-2620v2 chips @2.1GHz, 64GB RAM and an Nvidia Quadro K5000 GPU with 4GB RAM.
Software: gcc 4.8.2 (with O2 optimizations) and the CUDA 6.5 toolkit.

Conclusions and Future Work
- TKSimGPU’s achieves a 3.37x speedup over a naïve GPGPU algorithm because its pruning strategy avoids unnecessary computations.
- The average query execution time of TKSimGPU scales linearly with the size of the query and the size of the database.
- Our experiments provide empirical evidence that GPGPUs are efficient tools for coping with the inherent scalability problem of trajectory similarity queries.
- For future work: reduce the memory consumption of the near-join (step 4).