Introduction

- The design of a multi-dimensional data organization method attempts to solve the conflicting problems of how to store data compactly while enabling it to be 'queried' flexibly and efficiently.
- This requires a strategy for partitioning the space containing data, or partitioning the data itself, and providing means to access it. Geohashing is such a technique which uses space filling curves to achieve such a feat.
- This technique turns out to be especially useful when performing queries as single index queries are much faster than multiple-index queries. This indexing structure can also be used for proximity searches as the closest points are often among the closest geohashes.

Space Filling Curves provide us a way to store multi-dimensional data in a contiguous one dimensional way.

Problems in Geo-proximity computations

- Most space filling curves assign indexes to each grid in space based on the order in which they visit that space and store then in a one dimensional array. So a search for particular point in space would be a binary search on the array of indexes.
- Another useful feature of space filling curves is that we can perform proximity searches by matching the prefixes of the indexes that each point was assigned. This way we can compute bounding boxes enclosing the the two points in space.
- So among the various space filling curves available which one should we pick and are there any disadvantages that these curves have?
- Usefulness of space filling curves depends on how well they preserve locality, if (x,y) are coordinates of grid in 2d space and d is the distance along the curve when it reaches the point then points with d distance between each other will also include point (x,y) but the converse is not always true.
- We can identify points on each curve that are actually close in space but are far apart according to the traversal of the curve.

Proposed Method

- So our method should ideally give better proximity computation results while retaining all the advantages of a space filling curve.
- Our algorithm uses Z-order curve for indexing the points in space even though Hilbert curve is continuous and does a better job of preserving locality in most cases because of a unique property of Z-order curves we want to leverage.
- When we convert indexes to binary format we notice that the odd numbered indexes in the binary string represents top and bottom halves while even numbered indices represent left and right halves.

Algorithm : Given two geohashed binary vectors compute the distance between the two grids containing the points

Function Proposed (A,B)

Input : Two geo hashed binary vectors

\[ A = (a_1, ..., a_n) \] and \[ B = (b_1, ..., b_n) \]

Output : Distance between the two grids containing the geohashed points

\[ A[i] = A[2i] = (a_1, ..., a_i) \]
\[ A[i] = A[2i-1] = (a_1, ..., a_{i-1}) \]
\[ B[i] = A[2i] = (b_1, ..., b_i) \]
\[ B[i] = A[2i-1] = (b_1, ..., b_{i-1}) \]

\[ \text{Max} = \text{Max}(A, B) \]
\[ \text{Min} = \text{Min}(A, B) \]
\[ \text{Max} = \text{Max}(A, B) \]
\[ \text{Min} = \text{Min}(A, B) \]

Return \( \text{Max}(\text{Max}, \text{Min}) > (\text{Max}, \text{Min}) \)

Results

- So to test our algorithm initially we identified a few points on different sized two dimensional grids for which space filling curves showed large variation in actual distance versus distance calculated along the curve. Then we compute an average of these distances calculated using four different space filling curves and our proposed algorithm.
- We perform the same experiment for three dimensional grids.
- Lastly to make sure that the results are not in favour of any particular curve by selecting only a few points we compute distances between every possible combination of points and plot a cumulative distribution function of the all the combinations.

Evaluation of Space Filling Curves for Geohashing
Vineeth Kumar Byreddy (vxb9513@rit.edu)
Advisor : Peizhao Hu (hxpvcs@rit.edu)
B.Thomas Golisano College of Computing and Information Sciences