Term Paper: 2D hierarchical structure, Geohash and Quadtree

Song Yang (sy540)

April 3, 2018

Abstract

This paper will give readers a brief understanding of geological location method and will have practical experiments comparing among several geological searching methods. I will mainly concentrate on geolocation calculating method GeoHash in details and implement some of its functions. The paper will also introduce other technique related to this including quadtree, base32, and.

1 Introduction

Geohash is a latitude/longitude geocode system invented by Gustavo Niemeyer for encoding or decoding (latitude, longitude) pairs in a compact form. Encoding the position takes only (lat, lng) pairs neglecting the third coordinate (altitude) and as such it is unaware of the real 3D position of the object in space. It was made to have a better and more efficient way to store and search 2D locations.

As a consequence of the gradual precision degradation, nearby places will often (not in some edge cases, equator or a meridian) present similar prefixes. The longer a shared prefix is, the closer the two places are. Which is friendly to the database to search and build the index.

A quadtree is a tree data structure in which each internal node has exactly four children. Quadtrees are the two-dimensional analog of octrees and are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. The data associated with a leaf cell varies by application, but the leaf cell represents a "unit of interesting spatial information".[YS83]

The linear Quadtree (or Quadtree for short) computes tile approximations for geometries and uses existing B- tree indexes for performing spatial search and other DML operations. This approach results in simpler index creation, faster updates and inheriting of built-in B-tree concurrency control protocols. [KRA02]

The standard quadtree encoding technique allows the convenient representation of two-dimensional objects in an integer tree-a tree that can be efficiently manipulated using the simplest of operations.Again, three-dimensional objects can be represented in a similar manner using octree encoding.

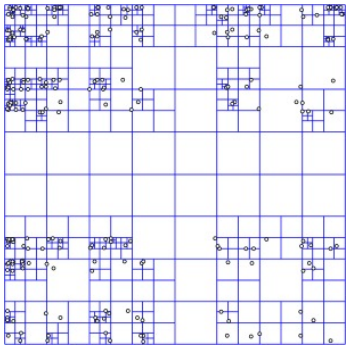


Figure 1: Quadtree Visualization.

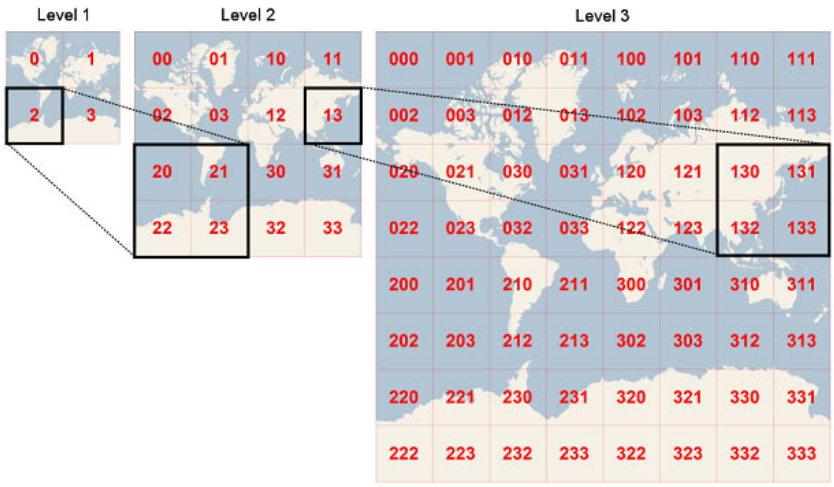


Figure 2: How GeoHash store locations.

2 Algorithm Description

2.1 GeoHash

GeoHash Algorithm encrypts the longitude and latitude with following steps. It is a function that returns a string which includes the location information. In order to run GeoHash algorithm, we need longitude, latitude and an integer number indicates precision. The whole algorithm is combined with the following two parts. Firstly, covert the longitude and latitude to binary code and mix them up. Then encode the binary number with the base32 method.

2.1.1 Binary Code Conversion

To obtain the latitude and longitude bits from an initial pair of coordinates representing a target point in space, the algorithm is applied recursively across successively more precise geographical regions bounding the coordinates.

The remaining geographical area is reduced by selecting a halfway pivot point that alternates between longitude and latitude at each step. If the target coordinate value is greater than the pivot, a 1 bit is appended to the overall set of bits; otherwise, a 0 bit is appended. The remaining geographic area that contains the original point is then used in the next iteration of the algorithm. Successive iterations increase the accuracy of the final Geohash string. An appealing property of the Geohash algorithm is that nearby points will generally share similar Geohash strings. The longer the sequence of matching bits is, the closer two points are.

For example, given a latitude, 46.5, the matched sequence of that is 1011100. On the opposite, if we know 1011100, we just know the approximate range of the latitude rather than the accurate number. In the other word, this algorithm discards part of accuracy.

It is a kind of approximate calculation. It cannot be unlimited recursive, so the precision indicates the time of the loop. According to precision and the range of latitude is from -90 to 90, the maximum error of the encoding algorithm can be

$$error = 90/2^N$$

N denotes the precision. As it said, if N = 20, the maximum error is 0.00009, which is nearly 100 meters.

2.1.2 Base32 Encoding

Base32 is one of several base 32 transfer encodings using a 32-character subset of the twenty-six letters A–Z and ten digits 0–9. There are many implementations for Base32, in this case, the alphabet the algorithm chose to use is shown in Table 1. The overall process is shown in Figure 3.

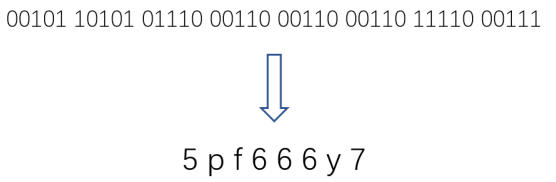


Figure 3: Overall Procedure.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
base32	0	1	2	3	4	5	6	7	8	9	b	c	d	e	f	g	
Decimal	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
base32	h	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z	

Table 1: Base32 character map

2.2 Quadtree

2.2.1 Simple Insert and Search

Similar to the binary tree, the insert functions is used to insert a node into an existing QuadTree. This function first checks whether the given node is within the boundaries of the current quad. If it is not, then we immediately cease the insertion. If it is within the boundaries, we select the appropriate child to contain this node based on its location. This function is O(Log N) where N is the size of distance.

The search function is used to locate a node in the given quad. It can also be modified to return the closest node to the given point. This function is implemented by taking the given point, compared with the boundaries of the child quads and recursing. This function is O(Log N) where N is the size of distance.

2.2.2 Optimization

As what you have realized, the insert and search function seems to be the simplest version of binary tree function. But in the quadtree, it's harder to keep it a full tree. But quadtree can build an index. Too much splitting may reduce the speed of the program. In order to avoid that situation, set P percent the maximum number of data as the cache range. In the range, update partial index rather split or merge blocks.

So consider the new insert and search functions.

Insertion: Insertion may cause split of the data block. When inserting data O, it should return the updated index. Steps as follows.

1. Find the block that data O might be in, O could be in multiple data blocks.
2. Insert O to the target data block.
3. If the data block size (D) $\leq$ (1+P)·M, calculate Hcode of the data O. Insert Hcode of the data record in the local index, the offset and the length in the data block file.
4. If size (D) $>$ (1+P)·M
   (a) split the data block and combine the blocks that the sum of data is smaller than M.
   (b) delete the MD5 code in global index and generate new code for a new record.

Delete When inserting data O, it should return the updated index. Steps as follows.

1. Find the block that data O might be in, O could be in multiple data blocks.
2. Delete O from the target data block.
3. If the data block size (D) $\leq$ (1+P)·M, calculate Hcode of the data O. Delete partial index record from the data block.

4. If size (D) > (1+P)·M
   (a) Find nearby district according to the O's code.
   (b) merge the blocks such that the sum of the number of blocks is smaller than M.
   (c) split the data block and combine the blocks that the sum of data is smaller than M.
   (d) Update indexes of merged blocks.
   (e) delete the MD5 code in global index and generate new code for the new record.

2.2.3 Hit detection

Let's say you have a bunch of points in a space. If some arbitrary point p is within your bunch of points. How can you find out if you have that point?

You could compare every single point you have, till you got p, but if you had 1000 points, and none of them was p, you'd have to do 1000 comparisons to find that out. Alternatively, you could get very fast lookup by keeping a grid (a 2D array) of booleans for every single possible point in this space. However, if the space these points are on is 1,000,000 x 1,000,000, you need to store 1,000,000,000,000 variables.

Or you could set up a quadtree. When you have it search for p, it will find out which quadrant it is inside. Then, it will find out what quadrant of that quadrant it is inside. And so forth. It will only have to do this at most seven times for a 100x100 space (assuming points can only have integer values), even if there are 1000 points in it. For a 1,000,000x1,000,000 space, it's a maximum of 20 times. After it finds its way to that rectangle node, it merely needs to see if any of the four children equal p.

2.3 Application Usage

The GeoHash algorithm can be used to divide geographic regions into a hierarchical structure like it is shown in Figure 2. A Geohash string represents a fixed spatial bounding box. For example, the latitude and longitude coordinates of 45.557, 18.675 falls within the Geohash bounding box of "u2j70vx29gfu". Appending characters to the string would make it refer to more precise geographical subsets of the original string. Furthermore, thanks to the encoding algorithm, the prefix of GeoHash code can be used to locate a larger area. With this feature, locations stored with GeoHash can be visited with a nearby searching algorithm.

Quadtree and its extend skill has plenty of applications, when initialization of an imaging system, each object creates a data structure that consists of a conventional sparse octree representation of its spatial occupancy. The construction of the tree is carried out in the local coordinate system of the object, in a standard way, by subdividing cubic volumes into eight equal sub-cubes. The resulting sub-cubes are then enclosed in spheres, and the polygons representing the faces of the polyhedron are checked against these spheres.[PG95]

3 Performance Analysis

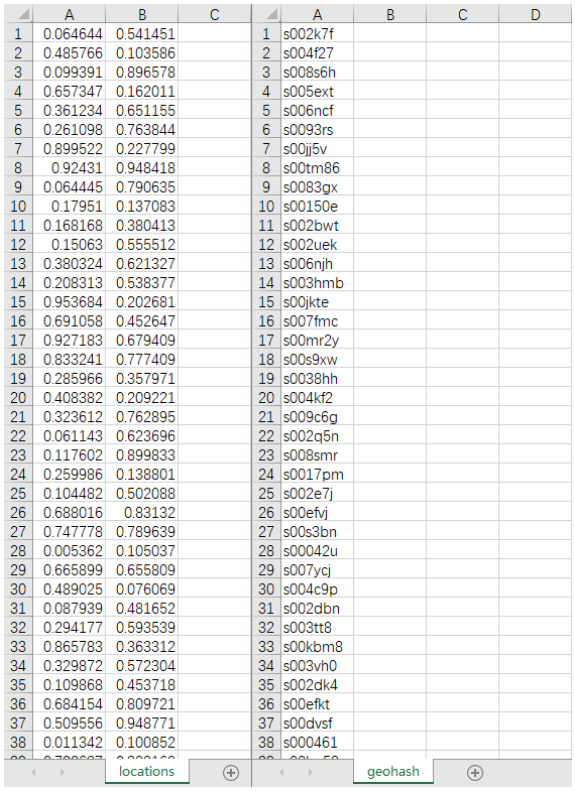
3.1 GeoHash

In order to show the advantage of GeoHash in actual use situations, I set the following implementations. Given a situation that I am going to implement the people nearby searching algorithm. I implemented the same function into two versions, one is simply calculated with longitude and latitude, the other is implemented with GeoHash.

The sample location data is randomly built into Excel. Imagine that we have 10,000 points whose latitude and longitude belong to [0,1]. Samples are partly listed in Figure 4.

In practice, I noticed that, after I disabled SQL cache, even I have 10,000 records to search and index, GeoHash still cannot have many advantages comparing with lng/lat calculating method. The theory behind the GeoHash have better efficiency is that the SQL query is similar when the user changes his location.

In order to perform more like actual use, firstly, with cache closed, I run the program 9 times and 17 times with the changing center coordinate from 0.3 to 0.7. After that, I got a satisfied result in Table 2 and Table 3. Step two, I open the cache and got the following runtime result in



	A	B	C
1	0.064644	0.541451	
2	0.485766	0.103586	
3	0.099391	0.896578	
4	0.657347	0.162011	
5	0.361234	0.651155	
6	0.261098	0.763844	
7	0.899522	0.227799	
8	0.92431	0.948418	
9	0.064445	0.790635	
10	0.17951	0.137083	
11	0.168168	0.380413	
12	0.15063	0.555512	
13	0.380324	0.621327	
14	0.208313	0.538377	
15	0.953684	0.202681	
16	0.691058	0.452647	
17	0.927183	0.679409	
18	0.833241	0.777409	
19	0.285966	0.357971	
20	0.408382	0.209221	
21	0.323612	0.762895	
22	0.061143	0.623696	
23	0.117602	0.899833	
24	0.259986	0.138801	
25	0.104482	0.502088	
26	0.688016	0.83132	
27	0.747778	0.789639	
28	0.005362	0.105037	
29	0.665899	0.655809	
30	0.489025	0.076069	
31	0.087939	0.481652	
32	0.294177	0.593539	
33	0.865783	0.363312	
34	0.329872	0.572304	
35	0.109868	0.453718	
36	0.684154	0.809721	
37	0.509556	0.948771	
38	0.011342	0.100852	

locations

	A	B	C	D
1	s002k7f			
2	s004f27			
3	s008s6h			
4	s005ext			
5	s006ncf			
6	s0093rs			
7	s00jj5v			
8	s00tm86			
9	s0083gx			
10	s00150e			
11	s002bwt			
12	s002uek			
13	s006njh			
14	s003hmb			
15	s00jkte			
16	s007fmc			
17	s00mr2y			
18	s00s9xw			
19	s0038hh			
20	s004kf2			
21	s009c6g			
22	s002q5n			
23	s008smr			
24	s0017pm			
25	s002e7j			
26	s00efvj			
27	s00s3bn			
28	s00042u			
29	s007ycj			
30	s004c9p			
31	s002dbn			
32	s003tt8			
33	s00kbm8			
34	s003vh0			
35	s002dk4			
36	s00efkt			
37	s00dvsf			
38	s000461			

geohash

Figure 4: Sample Data.

Type - index	1	2	3	4	5	avg
GeoHash (ms)	506	494	502	499	502	500.6
Lng/lat (ms)	522	540	598	538	527	545

Table 2: Runtime with 9 times running w/o cache

Type - index	1	2	3	4	5	avg
GeoHash(ms)	555	558	558	565	564	560
Lng/lat (ms)	628	621	616	618	604	617.4

Table 3: Runtime with 17 times running w/o cache

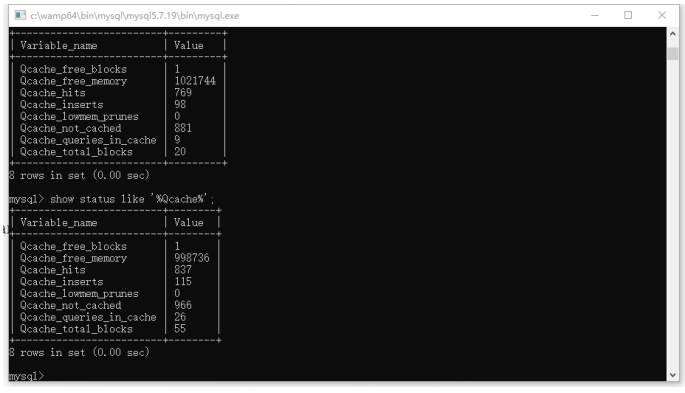


Figure 5: Checking the SQL Cache Status.

-	1	2	3	4	5
GeoHash(ms)	566	523	527	532	519
Lng/lat (ms)	619	551	534	534	520

Table 4: Runtime with 17 times running w/ cache

Type - index	before running	after GeoHash	after lng/lat method
Query Hit	693	769	761
Query Insert	89	98	113
Query in Cache	0	9	17

Table 5: MySQL cache status

Table 4. At the same time, I also record the query hit for MySQL showed in Table 5. It clearly shows that GeoHash has a briefly better cache hit and with more data, it will perform better.

As the analysis concluded, there are two advantages of GeoHash. First one is that when searching multiple times as a SQL server, GeoHash encoding function is friendly to SQL query w/cache. The other one is, when querying, building index on GeoHash code column will have a better performance.

4 Conclusion

Geohash implicitly defines a recursive Quadtree over the world-wide longitude-latitude rectangle. Geohash provides some notable properties. The encoding of GeoHash is actually more advanced Quadtree index method. As the Figure 1 shows, it is a z-curve that traverse all blocks in the map. There is also a lot traverse function like Hilbert curve. In quadtree, it exactly traverses the whole district. When using GeoHash code, the code itself can denote the part of the block.

We can take advantage of existing RDBMS feature which has been natively developed to support string data operation like searching or manipulating the string. Even in popular RDBMS like MySQL has a specific Spatial Geohash Functions to encode or decode a hash string. This fact will enhance the overall performance of GIS System for corporate especially in data filtering. To fully answer the second aspect of our research we need explore in more detail relating the information search model.[SDSL15]

5 Relevant Work

This implementation is to build on Java (JDK version:9.0). So I have to implement some related libraries.

5.1 SQL connection

I build a full MySQL-JDBC connection library which supports query, update and drop. The database connection can be used for any function including connection to a SQL server. The id of each column is unique, so the id can denote the number of each searched place.

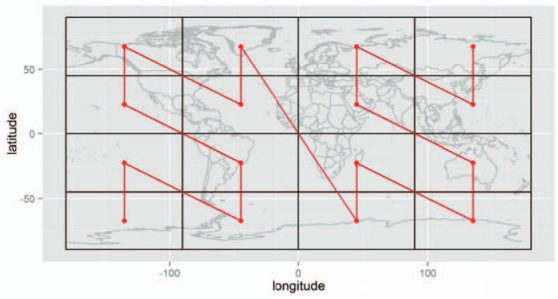


Figure 6: Relationship between Quadtree and GeoHash.

5.2 GeoHash Class

In GeoHash class, there are functions including base32 encode, convert to binary code, and merge binary code.

5.3 QuadTree

Implemented the quadtree data structure API. It can insert, split the node into 4 subnodes, find nodes, and clear the tree.

When inserting the object into the quadtree, if the node exceeds the capacity, it will split and add all objects to their corresponding nodes.

References

[KRA02] Ravi Kanth V Kothuri, Siva Ravada, and Daniel Abugov. Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 546–557. ACM, 2002.

[Nie08] Gustavo Niemeyer. Geohash, 2008.

[PG95] Ian J. Palmer and Richard L. Grimsdale. Collision detection for animation using sphere-trees. In *Computer Graphics Forum*, volume 14, pages 105–116. Wiley Online Library, 1995.

[Sam84] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

[SDSL15] Iping Supriana Suwardi, Dody Dharma, Dicky Prima Satya, and Dessi Puji Lestari. Geohash index based spatial data model for corporate. In *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*, pages 478–483. IEEE, 2015.

[YS83] Mark A Yerry and Mark S Shephard. A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications*, 3(1):39–46, 1983.