Fast geo lookup of city/zip for given lat/long

Preliminary version.
Version 0.9

Brief description

Geo lookup is commonly required feature used by many websites and applications. Geo lookup falls into 2 categories:

- For a given latitude and longitude, retrieve full postal address including street, city, zip.
- For a given latitude and longitude, retrieve nearest city with zip.

An overwhelming majority of websites and applications require only city and zip. The scope of this software is for applications requiring such.

Data providers for Lat, long, city and zip

Though there are many providers who sell data and software for a cost, this data is available free from government web sites and .org web sites. Here are 2 of them

- geonames.org : http://download.geonames.org/export/zip/ - File per country. Contains Lat/Long/city/zip.
- http://opengeocode.org/download.php

Geo lookup algorithm

Data source : CityZipLatLong.csv, a US CSV file will contain 49000 records. Each record contains city, zip, state and its corresponding lat/long. There will be one record per zipcode.
The process typically involves the following steps

- Application, such as those from smartphones, sends a latitude and longitude to this server.
- Server queries the city, zip, lat, and lon database and retrieve the nearest zip calculated using nearest distance.

Nearest point calculation involves selecting set of points (around x Km radius) from CityZipLatLong database around the given lat/long and determine the point with minimum distance among the distances from each point. For US, assuming 20KM radius is an optimal choice and assuming 20+ points selected for minimum distance calculation, it might take 50+ milliseconds.

Geographical distance between 2 lat/long is explained here. http://en.wikipedia.org/wiki/Geographical_distance. You can implement this algorithm or use open source software that supports geo point data. Elastic search server is one such open source server that can be used for this purpose.

Typical industry implementations

Most websites uses proprietary paid software for this purpose. In addition such implementations may implement on demand caching. The caching will be done for each lat/long. The caching has few disadvantages
Requires provisioning software (such as Elasticsearch) for new points to make on demand request. Requires maintenance and 100% uptime of Elasticsearch
Typical implementations may cache results for all lat/long as they arrive. This could be huge given the precision of lat/long is 5+ decimals.

Data perspective and alternate approach

For US, we have 49000 records (one per zipcode). 95 percent of adjacent zipcodes have distances between them greater than 8 km. A 0.01 difference in degrees in latitude is around 1 KM and a 0.01 difference in degrees in longitude is around 1.5 km. Given this, the percentage of lat/lon, among all lat/lon in US that may spill to adjacent lat/lon, insignificant. Given the usage context, using a 2 decimal precision should be acceptable for most web sites and applications.

With a 2 decimal precision, we can determine the worst-case scenario of lat/long values for US contiguous states. US Latitude ranges from 24.31 to 49.23. Longitude for US ranges from -124.46 to -65.57. With a 2 decimal precision this results in 15 million (2600 x 6000) possible lat/long values. Nearest zipcode can be calculated for 15 million lat/lon by pre-processing once. This cached data can be used at run time to serve user request. Given the vast area of US, with 20KM radius, only 7 million lat/lon have a nearest city/zip. However we can progressively increase the radius to a higher value(say 30, 40) to get nearest zip to diminish margin of error.

Fast geo lookup implementation

Currently FastGeoLookup is implemented only for the 48 contiguous US.

Steps

Set up

Download file from http://download.geonames.org/export/zip/US.zip
Download and install elastic search from http://www.elasticsearch.org/download/.
Start Elasticsearch server
Run mappings.sh to map the document in ES.

One time pre-processing

1. SHWriter.java- Generates ingestion curl command file, US.sh, to populate the ZIP records into Elasticsearch. Ingestion file (US.sh) will contain 49000 records each representing the curl command to ingest the lat/long/city/zip data.
2. Another file USIndex.csv will be generated. This file is same as the source file with an additional integer (0,1,2,…) id for each record.
3. Run US.sh to populate data in Elasticsearch.

4. GeneratLookupCache : Generate 7 million csv data that contains lat/lon as key and integer seqId as value. Store the 7 million processed records in file GeoCacheData.csv

<to be implemented>
Your web service app

Use cache file and USIndex.csv.
Service initialization
Use the generated file. load it in HashMap memory on service startup.

On user request
Receive the lat/lon in your service. Convert it to 2 decimal precision. Look it up in hashMap for this lat/lon.

References :
Elastic search documentation.
Data source for Lat/Lon/City/Zip data : www.geonames.org :
http://download.geonames.org/export/zip/

Comments/Suggestions: busyvishy@gmail.com

access it on a server without ever releasing its source code to the public. The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version. An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.