# An efficient multilevel approach for reverse geocoding in Big Data Environment

Guided by: Dr Jaiprakash Verma
Ph D, Computer Science & Application,
Associate Professor,
Computer Science & Engineering Department,
Institute Of Technology,
Nirma University, Ahmedabad

Nirav R Madhani,
Institute of Technology, Nirma University,
Ahmedabad, Gujarat, India
18bce135@nirmauni.ac.in

## Abstract

With the magnitude of spatial data continually increasing, the amount of reverse geocoding computations continues to grow. As a result, it makes sense to develop a more efficient reverse geocoding approach to keep up with the increased demand. In this paper, I propose a method that takes advantage of the fact that we do not need to query the entire database to find accurate result if we know which subset to query. Based on this I propose a multi-level framework that first determines subset of dataset to query and only queries that portion of dataset. This substantially outperforms traditional implementation over sparks framework.

## Introduction

The process of translating a geographic coordinate system (latitude, longitude) to a human-readable address or place name is called reverse geocoding. The phrase reverse geocoding refers to the inverse of forward geocoding (commonly referred to as address geocoding or simply "geocoding"). Reverse geocoding enables the identification of adjacent street addresses, locations, and/or geographic subdivisions such as neighbourhoods, counties, states, and countries. In conjunction with geocoding and routing services, reverse geocoding is a vital component of mobile location-based services and Enhanced 911 because it converts a GPS position to a legible street address that is easier for the end user to grasp but may not be as accurate.

Reverse geocoding has grown in popularity in recent years [6-8]. There are several public reverse geocoding systems, the most of which are provided for free. These services feature user-friendly application programming interfaces (APIs), which are simple to use for programmers. ‘GeoNames’ reverse geocoding web service is a representative example of these types of services, since it includes tools for finding local place names, Wikipedia articles, streets, neighbourhoods, countries, country subdivisions, and other location data from a coordinate [9].

With the widespread use of location-aware mobile electronic devices and the extensive application of LBS, the capability of reverse geocoding is in high demand. Meanwhile, with the constantly expanding of spatial data size, the computing quantity of reverse geocoding continues to increase. So, in order to keep up with the high demand, it does make sense to find out a new reverse geocoding method which efficiently handles big data.

Research proposed in this paper mainly focuses on distributed and parallel processing and storing of data.

## Related Research Work

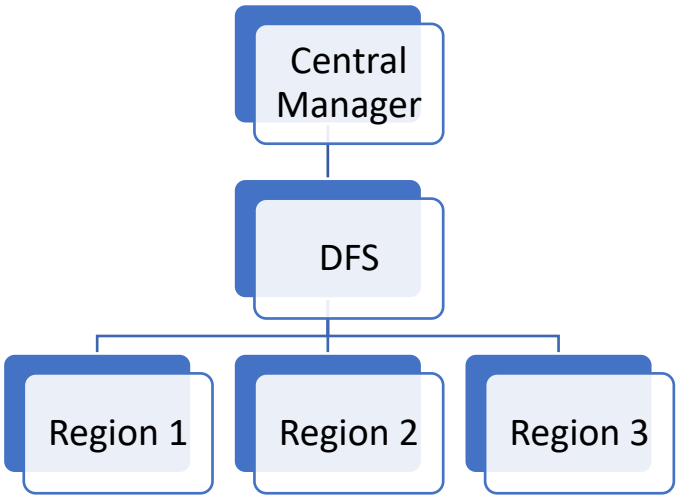
| Ref. | Objective | Data Domain | Methodology | Findings | Limitations |
|---|---|---|---|---|---|
| **R-trees: A dynamic index structure for spatial searching [1]** | In order to handle spatial data efficiently, as required in computer aided design and geo-data applications. | - | Proposed a tree like data-structure for querying spatial representations | Very efficient as compared to traditional indexing | Parellel and distributed aspects not discussed |
| **The R*-tree: An efficient and robust access method for points and rectangle [2]** | To address operations like map overlay, rectangle enclosing etc | Exact sources not provided | Added more functionality to R-Tree | Building cost slightly higher, but supports a variety of operations | Parellel and distributed aspects not discussed |
| **https://github.com/richardpenman/reverse_geocode [3]** | Reverse Geocode takes a latitude / longitude coordinate and returns the country and city | Custom / Repository | Use kD Tree | Very Fast. Highly recommended if kD tree can fit into memory. | kD tree needs to fit in memory. No Parellel Processing. Point Based and not polygon based lookup |
| **https://github.com/thampiman/reverse-geocoder [4]** | Reverse Geocode takes a latitude / longitude coordinate and returns the country and city | Custom / Repository | Parallelised implementation of K-D trees which promises an improved performance especially for large inputs. | Very Fast. Highly recommended if kD tree can fit into memory. Parellely processes makes it suitable for bulk query. | kD tree needs to fit in memory. Point Based and not polygon based lookup |
| **An Efficient Reverse Geocoding Method based on Global Subdivision Model [5]** | Efficiently return street address from Latitude / Longitude | Beiging_POI, Beijing_Road, Beijing_DISTRICT etc | Hashing. A key point in this step is encoding spatial line objects and region objects with an adaptive Geohash method | Time taken was 2/3 of R-Tree based methods | Performance not evaluated against kD trees. Distributed / Parellel processing have not been discussed. |

## Proposed Research Work

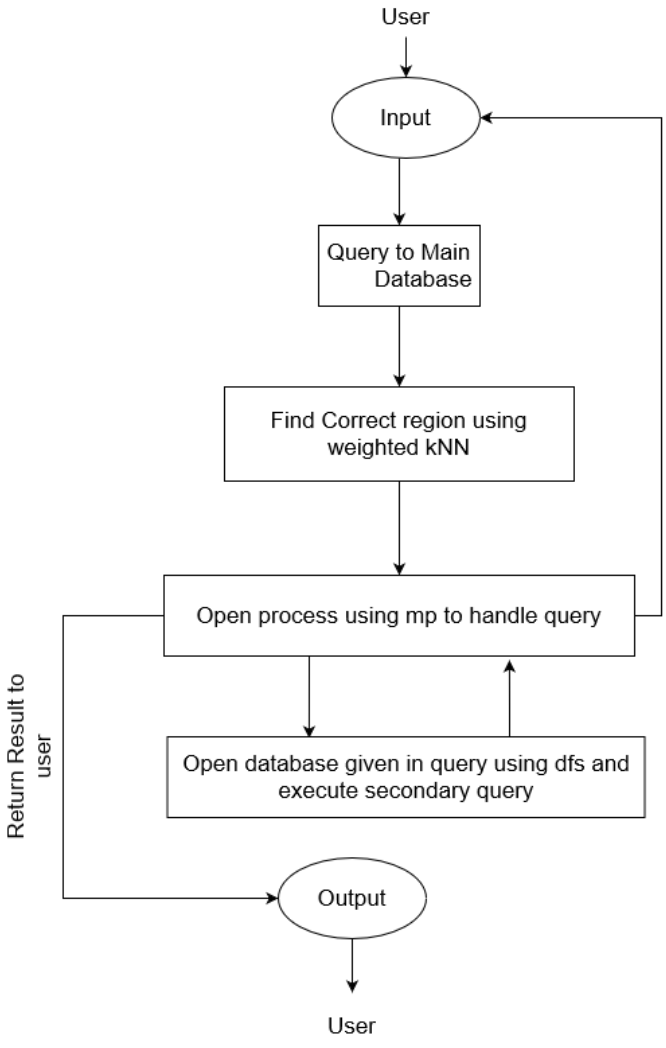
In this paper, I propose a recursive approach method to carry out reverse geocoding efficiently by using a pyramid of Level of detail for Geocodes. For scope of this paper, I discuss two level pyramid. At the apex, is the user who queries for latitude and longitude. At first level of pyramid, we get redirected to appropriate dataset of country containing that latitude or longitude. This is useful as:

1. Data can be distributed and decentralized
2. Different countries tend to adopt different address representation methods, so that can be dealt with separately.
3. It allows use of semi-structured data.

Structure of Database




System Flow Diagram



The working of system is as follows. A central top-level database contains sparsely distributed data of Latitude, Longitude and Region. Country Code in this case. When user enters his / her query. A primary query is performed by **central manager** to identify desired sub region. This is done using weighted kNN.

Once the correct region has been identified, a subprocess is generated to do further calculation and return result. This is where distributed part comes in. Now all further processing will be done by parallel thread. Meanwhile **central manager** can process other queries.

New process works load specific file from **dfs**, executes query on it and returns result to main thread / central manager. When central manager receives result from the subprocess it returns it to the user and kill the subprocess.

# Methodology and Concepts applied

## Technology and concepts applied:

1. Pyramid Building for querying the data at different level of details.
2. Distributing the database across multiple files.
3. Parallel processing using multiprocessing module in python

## Pseudo-Code for the Algorithm:

```
def Address(lat,long):

        targetDatabase = query in lower LOD database with (lat,lon)

        result = query in targetDatabase with lat,lon using new Process

        return result


def Split():

        for a in Select COUNTRY_CODE from database group by COUNTRY_CODE:

                create separate file with name a for data ->

                SELECT * FROM database WHERE COUNTRY = a
```

# Experimental Analysis

| METHOD | DATABASE SIZE | AVERAGE EXECUTION TIME |
|---|---|---|
| **SPARKS** | 41K | 1.3s |
| **SPARKS** | 144K | 1.9s |
| **SPARKS** | 3.17 M | 63.5s |
| **SPARKS** | 11.1M | 83.59s |
| **PANDAS** | 3.17 M | 7.06s |
| **PANDAS** | 11.1 M | 54s |
| **PROPOSED METHOD** | 104+ M* | 12.5s* |

*Since the proposed method primarily relies on divide and conquer + lazy loading, the entire dataset has not been loaded while querying.

## Results and Findings with discussion

When performing querying at small and medium size of dataset (upto 2-5 GB), pandas conspicuously outperforms sparks. Furthermore, it is very fast at handling transformations and functions on dataframe which is main component of the query. However, the data loaded from csv file is likely to inflate to up to 3 times its original size when it is stored in memory therefore imposing serious challenges beyond this level. Although sparks is built to handle large dataset in distributed fashion, it does not support any sort of indexing and is slow with transformations and operations on database. Here we can exploit the fact that we do not need to check entire database if we are certain about its subset where we can find what we are looking for and that's what the proposed method does, thereby greatly reducing querying time for the real Big Data. Further, the proposed method can be improved by generating and serializing kD trees in the file.

## Conclusion

We have taken comparison of three approaches and discussed their advantage and disadvantages. If the data is not Big Data, then we should use pandas with scipy to run queries, whereas when the data gets big, we can use divide and conquer approach with lazy computation to achieve great advancement in performance.

## References


1. Guttman, A. (1984, June). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data* (pp. 47-57).
2. Beckmann, Norbert, et al. "The R*-tree: An efficient and robust access method for points and rectangles." *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*. 1990.
3. Richardpenman. "Richardpenman/reverse_geocode." *GitHub*, https://github.com/richardpenman/reverse_geocode.
4. Thampiman. "Thampiman/Reverse-Geocoder: A Fast, Offline Reverse Geocoder in Python." *GitHub*, https://github.com/thampiman/reverse-geocoder.
5. Ma, Mengyu, et al. "An efficient reverse geocoding method based on global subdivision model." *2016 24th International Conference on Geoinformatics*. IEEE, 2016.
6. J. Krumm, "Inference attacks on location tracks," IEEE Pervasive Computing, vol. 4480, pp. 127-143, 2007. doi: 10.1007/978-3-540- 72037-9 8
7. X. Cao, G. Cong, C. S. Jensen, "Mining significant semantic locations from GPS data," Proceedings of the VLDB Endowment, vol. 3(1), pp. 1009-1020, 20 10. doi: 10.1477811920841.1920968

8. 1. Rekimoto, T. Miyaki, T. Ishizawa, "LifeTag: WiFi-Based Continuous Location Logging for Life Pattern Analysis," in Proc. 3rd international conference on Location-and context-awareness, Oberpfaffenhofen, Germany, 2007, vo1.l8, pp.952-955. doi: 10.1007/978-3-540-75160-1_3
9. GeoNames, "Reverse Geocoding Web services," [Online]. Available: http://www.geonames.org/exportlreverse-geocoding.html

## Appendix:

URL for Code of performed tests:

https://colab.research.google.com/drive/13C1BCfau8X-X4eT6EeTkIZzou8yrH5Dj?usp=sharing

URL for Code of proposed method:
https://colab.research.google.com/drive/1BwcWRw3mq2qm3uJuRYwTSmij79HzXGaT?usp=sharing