International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

Performance Enhancement for Geospatial Time-Series Data Prediction using Hash-Naive Bayes (HNB) Classification

Gyati Mittal[1], Mohini Mittal[2]

[1]K P Engineering College (UPTU), Uttar Pradesh, India

[2]Indraprastha College IPEC (UPTU) , Uttar Pradesh, India

Abstract: ***Due to the increasing demand for cloud services and the threat of privacy invasion, the user is suggested to encrypt the data before it is outsourced to the remote server. The safe storage and efficient retrieval of d-dimensional data on an untrusted server has therefore crucial importance. The paper proposed a new data distribution model which offers spatial order-preservation for d-dimensional data. In our Research Hybrid HASH based naïve bayes classification system can be calculated using symmetric keys. We have similarly involved a specification of possessions of an distributed hash arrangement when using it to convinced use and matched our suggestion beside it. We have presented a security investigation of our method beside with recognized and enhance of the system using MATLAB 2014Ra.***

Keywords: Distributed hash, Big data, geo spatial, naïve HASH table, data mining etc

1. Introduction

A complex spatial data set (which generally describes any kind of data where the location in space of object holds importance). We based this research on the analysis of some spatial characteristics of certain objects. We began with describing the spatial pattern of events or objects with respect to their attributes; we looked at how to describe the spatial nature/characteristics of entities in an environment with respect to their spatial and non-spatial attributes. We also looked at modeling (predictive knowledge management of complex spatial systems), querying and implementing a complex spatial database (using data structure and algorithms). Critically speaking, the presence of spatial auto-correlation and the fact that continuous data types are always present in spatial data makes it important to create methods, tools and algorithms to mine spatial patterns in a complex spatial data set. This work is particularly useful to researchers in the field of data mining as it contributes a whole lot of knowledge to different application areas of data mining especially spatial data extraction.

1.1. Geospatial, Time-Series datasets:

Data with spatio-temporal properties are commonly found in epidemiology, atmospheric and climate modeling, environmental and ecological systems, traffic and congestion analysis, and commercial sales tracking systems. Observations or features of interest are measured across geographical locations over long periods.

1.2. Hashing and Storage Methodology

Galileo supports streaming data that incrementally enters the system from a variety of sources. These data items are constantly evolving over time and can share a number of common attributes. Therefore, simply applying a standard hash function on the incoming data results in an approximately even distribution of files across all the nodes in the system, but does not account for similarity in the data being stored.

1.3. Geohash: Spatial Hashing Algorithm

To obtain a location-based hash for spatial groupings, we applied the Geohash Algorithm on incoming data [24]. A Geohash is a string-based representation of a bounding box around a location created by interleaving bits obtained from latitude and longitude pairs. For example, the latitude and longitude coordinates of N 39.54, W 107.32 fall within the Geohash bounding box of 9x58vy4. Longer Geohash strings represent more precise spatial regions, a characteristic which can be exploited during the hashing process to obtain a specific granularity for positioning data in the system.

1.4. Feature Hashing

Once a group has been chosen for a data item based on its spatial characteristics, an additional level of hashing is required to select a destination node within the group. At this stage in the hashing process, any number of the remaining available data dimensions can be used as input for the hash function. Our particular dataset has a temporal range associated with each data item, so we used the initial recording time as input to the SHA-1 hash algorithm and then divided the hash space among the nodes in each group.

1.5. Data Distribution and Load Balancing Evaluation

To determine the impact of our hierarchical hashing scheme on how files are distributed in the system, we compared the distribution results of our controlled dispersion strategy against the same data inserted using a flat SHA-1 hash of all metadata values.

1.6. Indexing and Retrieval

Once data has been stored across the nodes in the system, an efficient means for retrieval is necessary. In a traditional DHT a hash function is used for both storage and retrieval operations, which constrains the expressiveness of the queries that can be performed during retrieval. We address this weakness with our indexing system. Additionally, data storage needs often evolve over time and may require changes to be made to the hashing hierarchy, meaning that all previously-stored data would not be reachable using a new set of hash functions.

1.7. Feature Graph Implementation and Structure

Any node in the system can be contacted to perform a storage operation, which will then route the request directly to its destination node. Upon arrival, the data is fully inspected to determine its attributes, which could include spatial location, temporal information, features, and details about the device that generated the data.

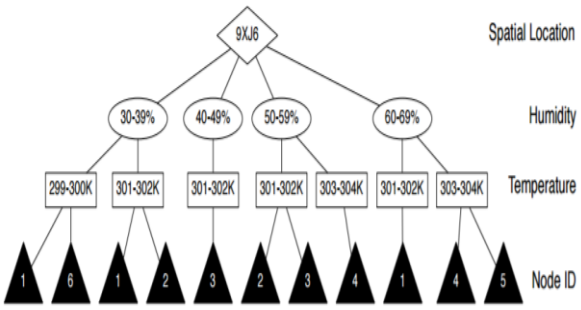

Figure 1.1: A simplified view of a feature graph for three dimensions. References to storage node IDs are included at each vertex in our implementation, but they are omitted in this example for clarity.

2. Related Work

Below are various reviews as per author studies:

MongoDB [2] shares several design goals with Galileo, butis a document-centric storage platform that does not support analytics directly. However, MongoDB has rich geospatial indexing capabilities and supports dynamic schemas through its JSON-inspired binary storage format, BSON. MongoDB can usethe Geohash algorithm for its spatial indexing functionality, and is backed by a B-tree data structure for fast lookup operations. For load balancing and scalability, the system supports sharding ranges of data across available computing and storage resources, but imposes some limitations on the breadth of analysis that can be performed on extremely large datasets.

SciDB [4] is a scalable scientific storage system that supports multidimensional data. Although its name implies a link with relational databases, SciDB is not concerned with providing ACID guarantees or strong transaction support. Instead, SciDB focuses on incremental scalability and petabyte scale datasets. The system also provides built-in computation and analysis tools, whereas Galileo is only concerned with storage; analysis can be performed outside the system within the Granules framework or some other distributed computation engine. Metadata is stored in a centralized system catalog implemented as a PostgreSQL database, contrasting with the combination of feature graph and metadata graphs used in Galileo.

Citing the use of hierarchies in traditional distributed applications such as multicast and DNS, Ganesan, Gummadi, and Garcia-Molina [6] propose a paradigm called Canon, which provides a hierarchy on top of existing flat DHTs. Canon subdivides system computational nodes into domains, which provide logical groupings of resources. Domains can contain any number of subdomains, and a domain that contains system nodes is referred to as a leaf domain. Leaf domains are structured in the same way as a traditional flat DHT

2T-DHT [7] implements a two-tier DHT hierarchy for publish/subscribe systems. In 2T-DHT, the hierarchy is used to organize nodes based on their uptime and available resources. All nodes begin in a lower tier and then migrate to the higher tier as they demonstrate their stability. The 2T-DHT network is implemented as multiple Chord rings, which reduces the amount of communication required to publish.

3. System Model

In research methodology discovery Graph improves upon (and supersedes) the feature and metadata graphs by making knowledge extraction part of the indexing process. As records are streamed into the system, the Discovery Graph maintains a variety of statistics at each vertex that describe the underlying data distributions and their interactions. Maintaining this information boosts index fidelity, greatly improves the speed of queries meant to generate synopses, and serves as a platform for development of functionality.

a) Query MapReduce Framework
While the summary statistics and models in the Discovery-Graph are lightweight and incur minimal processing costs, they provide nuanced insights about the underlying dataset. Galileo includes rich retrieval functionality to allow this information to be queried and used individually by end users, aggregated across dimensions, or used to locate phenomena and features of interest autonomously.

b) Detecting and Quantifying Feature Relationships
Most real-world systems involve several interrelated features. These relationships may represent dependencies or correlations between events or variables; for instance, absolute humidity is impacted by temperature and pressure, and precipitation may be classified as rain, hail, or snow depending on the current temperature (along with other atmospheric conditions).

c) Artificial Neural network
Our framework includes an interface that enables arbitrary models and prediction methods to be placed at graph vertices in a similar fashion to the multiple linear regression instances, and can produce output datasets in the form of classifications, function approximations, or forecasts. We have incorporated support for online artificial neural

networks (ANNs) machine learning library to accommodate nonlinear predictive models. Compared to the linear methods discussed in previous sections, ANNs generally involve more complex computations for training. Furthermore, they often do not completely converge on one final set of model parameters, so training is an inexact and iterative process.

d) Time-series Forecasting: ARIMA

Autoregressive integrated moving average (ARIMA) models are specifically designed for time-series data and allow predictions to be made on non-stationary data types. ARIMA models are parameterized by three parameters, p, d, and q, which correspond to the autoregressive, integrated, and moving average components of the model, respectively. Our implementation allows these parameters to be chosen autonomously by the system or specified at query time by client applications.

e) A Naive Bayes classifier

Uses the probabilities associated with events or features in the dataset to make predictions. The key assumption of this type of model is that all features are independent: completely unrelated to any of the other features.

Despite the fact that this assumption may not always hold, naive Bayes has proven to be effective in practice for a variety of classification tasks, including text categorization and medical diagnosis. To classify a given set of samples, naive Bayes uses the combined probabilities of the events to choose the most probable outcome.

4. Result and Discussion

This work introduces the mixture cryptography of the examination we surveyed the best aggregate methodologies in the cryptography of a piece figure framework for Geospatial on cloud.

The resultant of distributed key Processes is symmetric, that is to roughly use to encode the content or given content by client is not the same as the key used to decode the message. The encryption key, distinguished as the Public key which used to encode a correspondence, yet the message must be deciphered through the data that has the decoding key, perceived as the private key.

This kind of encryption has an amount of focal points over regular symmetric Ciphers.

It implies that the beneficiary can make their open key roughly accessible somebody inadequate to send them a correspondence uses the strategy and the collector's open key to do as such. A watcher may have both the methodology and the general population key, yet will in any case not be competent to unravel the content. Individual the beneficiary, with the private key can unscramble the message.

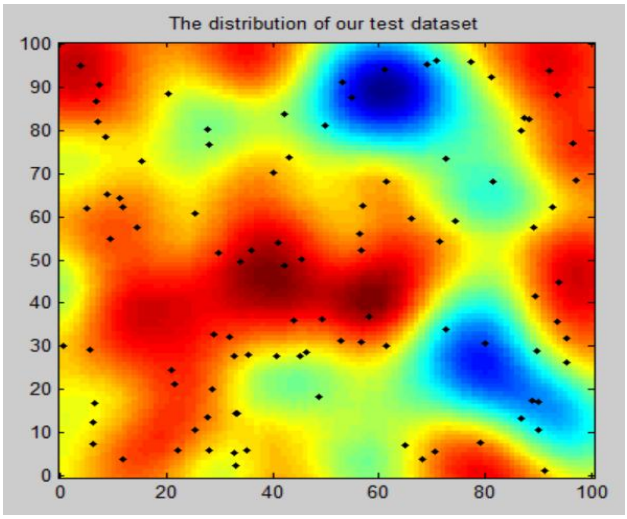

Figure 1: Distribution of Test Dataset such as high frequent clustered location in dark color and black datapoints are different location of dataset

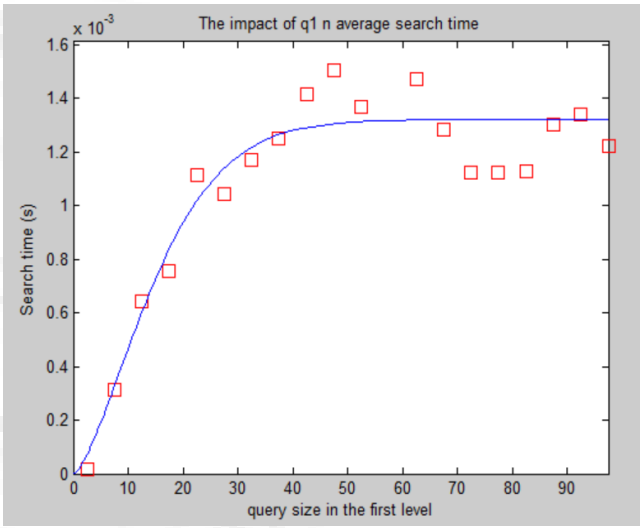

Figure 2: Searching time improvement for Distribution of Test Dataset with query size minimization inis different location, red mark specification for base work fluctuation of time as far our proposed approach having sequential time.

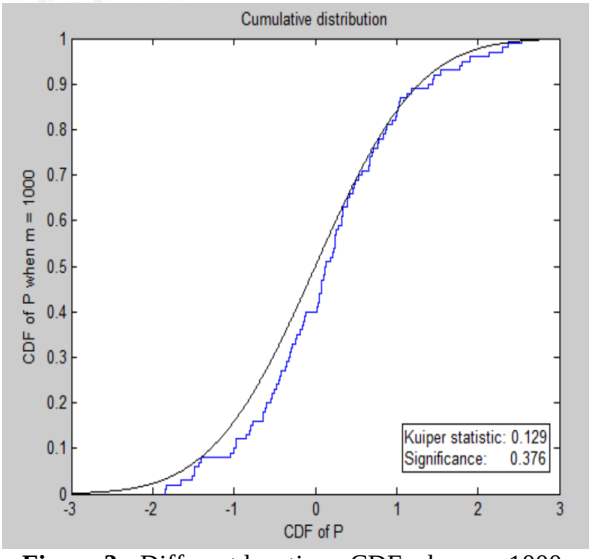

Figure 3: Different location , CDF when m=1000, specification for our proposed approach having sequential data sizes using proposed hash naïve byes classification.

5. Conclusion and Future Work

The partitioning and indexing scheme we have implemented in Galileo allows clients to make efficient exact-match and range queries across a number of dimensions, a feature not supported by traditional DISTRIBUTED HASH storage systems. This functionality is made possible by:

1) Ensuring that data items with some similarity, e.g., spatial locality, time series, or another attribute are stored using our controlled dispersion strategy.
2) Indexing the location of these data items.
3) Reasoning about the data stored in the system at a lower resolution, thus providing a higher-level or general view of the information to reduce network traffic and memory consumption.

In Future, Artificial Neural Networks could be used to predict and react to changing query workloads or new resource constraints and provide information that could be used to reorient our feature graph dynamically. Reinforcement learning techniques could be employed for query optimization, load balancing, and fault tolerance operations.

References

[1] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for clustering evolving data streams. In Proceedings of the 29th international conference on Very large data bases-Volume 29, pages 81–92. VLDB Endowment, 2003.
[2] A. Bialecki, M. Cafarella, D. Cutting, and O. OMALLEY. Hadoop: a framework for running applications on large clusters built of commodity hardware. Wiki at http://lucene. apache. org/hadoop, 2005.
[3] E. Brewer. Cap twelve years later: How the" rules" have changed. ComputerIEEE Computer Magazine, 45(2):23, 2012.
[4] P.G. Brown. Overview of scidb: large scale array storage, processing and analysis. In Proceedings of the 2010 international conference on Management of data, pages 963–968. ACM, 2010.
[5] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.
[6] P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in g major: designing dhts with hierarchical structure. In Distributed Computing Systems, 2004. Proceedings. 24th International Conference on, pages 263–272. IEEE, 2004.
[7] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In Foundations of computer science, 2000. proceedings. 41st annual symposium on, pages 359–366. IEEE, 2000.
[8] D. Hastorun, M. Jampani, G. Kakulapati, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazons highly available key-value store. In In Proc. SOSP. Citeseer, 2007.