ISSN:0975-9646

M. Prosant Kumar et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (4) , 2012,4689 - 4691

# Fractal Based Management of Heterogeneous Data Streams

M.Prosant Kumar[1], N.Naga Subrahmanyeswari[2] , S. Rama Sree[3]

[1,3]*Department of CSE,* [2]Department of IT

*Aditya Engineering College*

*Surampalem, Kakinada*

***Abstract* - Efficient handling of huge amount of data stream is become mandatory in various domains such as business, finance, sensor networks etc. Come through these data streams poses challenges those are distinct from those addressed by orthodox database management systems. The enormous increase of Sensor Networks, GPS enabled devices and RFIDs results in highly impulsive environments where objects as well as queries are continuously moving. This paper presents a continuous query processing designed specifically for highly dynamic environments. This paper presents a data stream indexing system that satisfies the requirements in evolving data streams. This paper applying the fractal based indexing scheme that enable the system for fast indexing and querying. This paper also address the issue of memory based and disk based fractal management to support skew distributions of data streams.**

***Keywords*— Data stream, Indexing,Querying, Self similarity, Fractal Dimension,Fractal Tree**

## I. INTRODUCTION

A fractal can be defined by the self-similarity property, i.e., an object that presents roughly the same characteristics over a large range of scales [Schroeder 1991]. Accordingly, a real dataset exhibiting fractal behaviour is exactly or statistically self-similar, such that parts of any size of the data present the same characteristics of the whole dataset. From the Fractal Theory, the fractal dimension [3][4] is particularly useful to data analysis, as it provides an estimate of the intrinsic dimension D of real datasets. The intrinsic dimension gives the dimensionality of the object represented by the data regardless of the dimension E of the space in which it is embedded. In other words, D measures the non-uniformity behaviour [6] of real data [Faloutsos and Kamel 1994; Traina et al. 2005]. For instance, a set of points defining a plane embedded in a three-dimensional space (E = 3) has two independent attributes and a third one correlated to the others, resulting in D = 2. The fractal dimension of real datasets can be determined by the Correlation Fractal Dimension D2. An effcient approach to measure the fractal dimension [1] of datasets embedded in E-dimensional spaces is the Box-Counting method [Schroeder 1991], which defines D2 as presented in Equation 1, where r is the side of the cells in a (hyper) cubic grid that divides the address space of the dataset and Cr,i is the count of points in the ith cell.

$$D_2 \equiv \frac{\partial log(\sum_i C_{r,i}^2)}{\partial log(r)} \qquad r \in [r_1, r_2]$$

An efficient algorithm [2] (linear cost on the number of elements in the dataset) to compute D2 was proposed by Traia et al. [2000]. Thus, D2 can be a useful tool to estimate the intrinsic dimension D of real datasets with feasible computational cost.

Concepts from the Fractal Theory have been applied to several tasks in data mining and data analysis, such as selectivity estimation [Baioco et al. 2007], clustering[7] [Barbará and Chen 2000], time series forecasting [Chakrabarti and Faloutsos 2002], correlation detection [Sousa et al. 2007] and data distribution analysis [Traina et al. 2005].

The information of intrinsic behaviour provided by the fractal dimension D2 can also be applied to detect behaviour changes in evolving data streams. Essentially, the idea is to continually measure the fractal dimension of the data stream over time in order to monitor its evolving behaviour. Thus, significant variations in successive measures of D2 can indicate changes in the intrinsic characteristics of the data.

### 1.1 Challenges in Existing System Incremental Evaluation of Continuous Queries:

Most of the spatio-temporal queries are continuous in nature. Unlike snapshot queries that are evaluated only once, continuous queries require continuous evaluation as the query result becomes invalid with the change of information.

Existing algorithms for continuous spatiotemporal queries aim to optimize the time interval between each two instances of the snapshot queries with the large number of continuous queries, re-evaluating a continuous spatial, temporal query, even with large time intervals, poses a redundant processing for the location-aware servers.

### 1.2 Large Number of Concurrent Continuous Queries:

Most of the existing spatio-temporal algorithms focus on evaluating only one spatio-temporal query In a typical location-aware server, there is a huge number of concurrently outstanding continuous spatiotemporal queries. Handling each query as an individual entity dramatically degrades the performance of the location-aware server.

### 1.3 Wide Variety of Continuous Queries:

Most of the existing query processing techniques focus on solving special cases of continuous spatio-temporal queries are valid only for moving queries on stationary objects, are valid only for stationary range queries

## II. MOTIVATION

Thus, a system that manages infinite heterogeneous data streams must satisfy the following requirements

- Mechanism for rate control must be in place to deal with data streams that are nearly always too fast for any indexing system.
- There must be capacity control because the data streams are infinite while the storage space is finite.
- The structure of the data varies in time (i.e., heterogeneity of data streams), therefore adaptive indexing methods must be developed.
- Data should be efficiently stored.
- Volume can be huge, therefore everything must be appended -only; both indexing and query processing must only use forward-only file access.
- Users can query and filter the index streams efficiently

### 2.1 Proposed Approach

Proposed approach considers not only the problem of efficiently archiving heterogeneous data streams, but also that of querying them. Queries of interest include filtering for specific attributes in data readings and filtering them based on their values. Common queries are simple in complexity, but because of the huge volume of data streams and queries, the query response time must be minimized

## III. FRACTALS

Fractals are of rough or fragmented geometric shape that can be subdivided in parts, each of which is a reduced copy of the whole. They are crinkly objects that defy conventional measures, such as length and are most often characterized by their fractal dimension. They are mathematical sets with a high degree of geometrical complexity that can model many natural phenomena. Almost all natural objects can be observed as fractals [5] (mountains, coastlines, trees, and clouds). Their fractal dimension strictly exceeds topological dimension

### 3.1 Fractal dimension

The number, very often non-integer, often the only one measure of fractals it measures the degree of fractal boundary fragmentation or irregularity over multiple scales it determines how fractal differs from Euclidean objects (point, line, plane, circle etc)

### 3.2 Self-similarity/ Semi-self similarity

Fractal is strictly self-similar if it can be expressed as a union of sets, each of which is an exactly reduced copy (is geometrically similar to) of the full set. The most fractal looking in nature do not display this precise form Natural objects are not union of exact reduced copies of whole. A magnified view of one part will not precisely reproduce the whole object, but it will have the same qualitative appearance. This property is called statistical self-similarity or semi-self-similarity

### 3.3 Fractal Tree data structure Advantages

A drop-in B-tree replacement supporting fast insertions for high entropy data. 100x faster inserts good data locality with no memory-specific parameterization. Data can be indexed quickly on disk, even if the data arrival order is independent of data-query order.

Thus, insertion performance is a currency that we can use to achieve faster query performance through indexing Fractal Trees are generally faster, easier to implement, and platform independent Fractal Trees scale with disk bandwidth not seek time.

### 3.4 Fractal Tree Properties

1. log N arrays, one array for each power of two
2. Each array is completely full or empty
3. Each array is sorted
4. Searching in a Simplified Fractal Tree
5. Fractal Tree indexes can use 1/100th the power of B-trees
6. Fractal Tree indexes will make good use of cores.
7. Fractal Tree indexes ride the right technology trends

In the future, all storage systems will use Fractal Tree indexes

## IV. RESULTS

Menu
1. Insert
2. View
3. Search
4. Exit

**Enter your choice: 1**
Key: 25
Current Record No: 1
Key: 38
Current Record No: 2
Bin of 2 is 01
Cur Node:
Array:
Key: 25
Key: 28
Current Record No: 3
Bin of 3 is 11
Key: 44
Current Record No: 4
Bin of 4 is 001
Cur Node:
Array:
Key: 28
Cur Node:
Array:
Key: 25
Key: 38
Key: 34
Current Record No: 5
Bin of 5 is 101
Key: 63
Current Record No: 6
Bin of 6 is 011
Key: 24
Current Record No: 7
Bin of 7 is 111
Key: 55
Current Record No: 8
Bin of 8 is 0001
Cur Node:
Array:
Key: 24
Cur Node:
Array:
Key: 34
Key: 63

Cur Node:
Array:
Key: 25
Key: 28
Key: 38
Key: 44
Key: 65
Current Record No: 9
Bin of 9 is 1001
Key: 36
Current Record No: 10
Bin of 10 is 0101
Key: 26
Current Record No: 11
Bin of 11 is 1101
Key: 48
Current Record No: 12
Bin of 12 is 0011
Key: 43
Current Record No: 13
Bin of 13 is 1011
Key: 20
Current Record No: 14
Bin of 14 is 0111

**Enter your choice: 2 (view all)**
Given Records
Node-
Array:
Key: 0
Node-
Array:
Key: 20
Key: 43
De-Array:
Key: 26
Key: 36
Key: 48
Key: 65
E-Array:
Key: 24
Key: 25
Key: 28
Key: 34
Key: 38
Key: 44
Key: 55
Key: 63

**Enter your choice: 3 (search)**
Enter the Key of the Record to be found: 38
Search Size: 2
Invalid Key, Record Not Found
Search Size: 4
Invalid Key, Record Not Found
Search Size: 8
Record Found @ Node-3
Record Info:
Key: 38

## V. CONCLUSION

Extension of proposed work is to apply structural clustering techniques to perform online clustering of the incoming data stream into more homogeneous streams, each of which can be indexed more efficiently.

## REFERENCES

[1] L. Liebovitch and T. Toth. A Fast Algorithm toDetermine Fractal Dimensions by Box Countig. Physics Letters, 141A (8), 19.
[2] Baioco, G. B., Traina, A. J. M., and Traina, C. Mamcost: Global and local estimates leading to robust cost estimation of similarity queries. In Proceedings of the SSBBM International Conference on Scientific and Statistical Database Management. Ban, Canada, pp. 6–16, 2007
[3] Barbará, D. and Chen, P. Using the fractal dimension to cluster datasets. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Boston, MA, USA, pp. 260–264, 2000.
[4] D. Barbar a and P. Chen. Using the Fractal Dimension to Cluster Datasets. Technical Report ISE-TR-99-08, George Mason University, Information and Software Engineering Department, Oct. 1999.
[5] Chakrabarti, D. and Faloutsos, C. F4: large-scale automated forecasting using fractals. In Proceedings of the CIKM International Conference on Information and Knowledge Management. McLean, VA, EUA, pp. 2–9, 2002.
[6] Faloutsos, C. and Kamel, I. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In Proceedings of the ACM PODS Symposium on Principles of Database Systems. Minneapolis, MN, USA, pp. 4–13, 1994.
[7] A. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, Englewood Cli_s, New Jersey, 1988.