International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

Amendments to the R*-Tree Construction Principles in Distributed Environment

F. Sagayaraj Francis
Pondicherry Engineering College
Puducherry
India

P. Xavier
Sacred Heart College
Tirupattur
India

***Abstract*—This paper identifies the rooms for improvement in the processes that are reported in the literature for the construction of R*-Trees in Hadoop and MapReduce environment. Subsequently, the amendments that are required to amend these deficient processes have been suggested. The effectiveness of the proposed amendments have ben experimentally verified and presented.**

Keywords—R*-Trees, Hadoop, MapReduce, Hilbert values

I. INTRODUCTION

We are in an age of burgeoning big data. The data that are generated are voluminous, varied, complex and come at a rate that is challenging to data management and data processing systems. They come from every imaginable quarter of all perceivable domains. This necessitates the development of robust and scalable data management and data processing systems. Hence parallel and distributed storage as well as computing models have been developed and deployed. Hadoop and MapReduce is one such approach that is extensively and exhaustively employed today to build such systems. Hadoop is a distributed computing framework, where clusters with many computing and storage facilities are dynamically formed. The management of the clusters are transparent to the users. MapReduce is a programming paradigm for Hadoop distributed computing framework.

R-Tree is data partitioned multi-dimensional indexing technique. Fig. 1 gives a sample R-Tree. The original R-Tree has undergone a sea change and wider ramifications. R*-Tree is one such efficient version of R-Tree. In this paper the terms R-Tree and R*-Tree are used interchangeably to refer to R*-Trees and its construction principles. For the first several years people attempted at improving the performance by enhancing the splitting methods of the nodes and forcing reinsertions during the insertion of a single point. Later attempts were made to insert objects parallel and concurrently. Subsequently the behaviour of the R-Trees was studied during bulk loading. Today efforts are made to employ effectively construct and deploy R-Trees in a distributed environment.

The general approach to construct an R-Tree in a distributed environment, specifically Hadoop and MapReduce, consists of three phases. In the first phase a function computes a representational single key value for each of the points in the n dimensional space. In the second phase these key values are used to partition the dataset into a predetermined set of partitions. Subsequently a small R-Tree is constructed for each partition. In the third and the final phase these small R-Trees are merged into a single R-Tree with all the data points originally considered. While the first two phases use the MapReduce paradigm to achieve their objectives, the third phase is carried out sequentially due to lesser computational constraint. The phases are pictorially illustrated in Fig. 2.

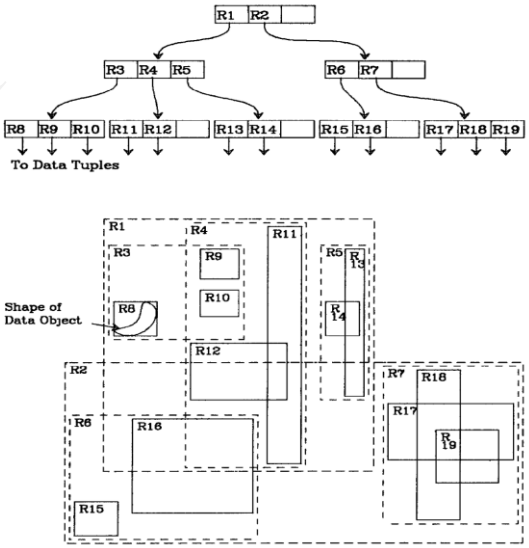

Fig. 1. A sample R-Tree and the corresponding spatial objects in 2D

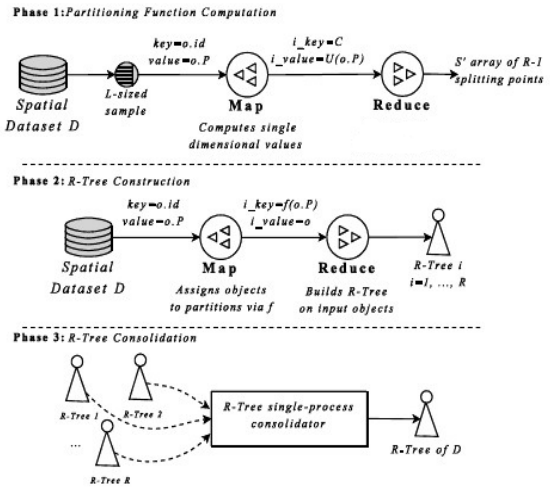

Fig. 2. Phases of R-Tree construction in a distributed system

The goal of this paper is to present the shortcomings observed in the methods proposed in the literature that are used to construct R-Tree and its clones using Hadoop and MapReduce and provide solutions to these shortcomings.

The paper is organized as follows: Section II gives a brief literature review of the relevant works pertaining to the realm of the problem addressed in this paper. Section III lists the very important observations made in the literature regarding R-Trees in the context of Hadoop and MapReduce that form the basis and motivation of this work. These observations are simply the gaps in achieving the desired results of the problem addressed in this paper. Section IV presents the actual contributions of this paper. The proposals given in this section enhances the existing processes and also fills the gaps in the existing processes. Section V presents the results of the experiments conducted. Section VI concludes the paper and suggests possible continuation for the work reported in the paper.

II. LITERATURE REVIEW

The database community relied on B-Tree [1, 2] and its clones for indexing large volume of data. But they failed miserable when adopted for higher dimensions. R-Tree [3] provided an answer for this challenge by recursively partitioning data to form a structure like a B-Tree but preserving spatial proximity of data. It used the Minimum Bounding Rectangles (MBRs) to encompass the data partitions. The edges of the rectangles are orthogonal to the axes that form the data space. The method was easily scalable for higher dimensions. R*-Tree [4] ameliorated the performance of R-Tree and answering nearest neighbour queries and directional queries became easy and straight forward. A review of a host of ramifications of R*-Tree and their applications in various fields is reported in [5, 6]. The performance of R-Trees in the context of fractal dimensions were later studied and reported in [7, 8]. The fundamentals of constructing and querying R-Trees were further refined in [9].

The applicability of the R-Trees was very much evident in various domains where large volume of multidimensional data was involved. Bulk insertions using small-tree-large-tree method [10], sort based parallel loading method [11] method and several such methods were proposed to handle data in these domains. Improvement in the response time was achieved by efficiently storing the R-Tree in the physical medium by controlled duplication of the nodes [12]. A wide range of literature is available related to the estimations of various parameters of R-Trees during point and range queries. One such work is reported in [12]. An insight into R-Trees in P2P systems is reported in [14, 15].

The advent of Hadoop framework and MapReduce programming paradigm brought interesting advancements in R-Trees. Researchers attempted to study the behaviour and performance of R-Trees in distributed environments that are based on Hadoop and MapReduce [16] which forms the basic plot of this paper. Incidentally bulk loading [17], estimations [18] and every field of exploration of R-Tree migrated into this new environment and the findings are regularly reported. A related survey is found in [19]. This survey also gives a comprehensive analysis of shortcomings of the Hadoop and MapReduce paradigm.

III. INFERENCE FROM THE LITERATURE SURVEY

The first observation that was made during the review of literature is the aberrations caused by the single representational values of the multidimensional points obtained by applying space filling curves. Consider a point in a 2-dimensional space. It has eight neighbors at a distance of one unit from it. If the basic pattern of the space filling curve uses four points, then at least half of the neighbors do not form a sequence in the one dimensional space. The space filling curves use smaller sized blocks to build bigger sized blocks and connect the points recursively with the same pattern. As the number of blocks becomes more and the sizes of the blocks become big, the spatial proximity of good number of points is completely lost and results in poor partitioning of points. This poor partitioning results in inefficient grouping of data with more overlapping of partitions and increased perimeters of the partitions. These are the most undesired characteristics that generate an inefficient R-Tree or its clones. The aberration is exponentially magnified for higher dimensions.

The second observation is regarding merging R-Trees. Given a space S and partitions s_i of S, the R-Trees of s_i, when merged are not comparable with the R-Tree of S. The differences in the measures of the parameters of R-Tree are so severe that the approach itself leaves a lot of questions than answers. Moreover deciding the partitions even before the R-Tree is constructed does not form the correct launching of the process for the purpose. The problem becomes even more complicated when the points are not uniformly distributed or any other distribution is followed. In short, the problem exaggerates when the points form clusters and partitioning is done arbitrarily without considering this information.

The third observation pertains to the slanting of the results due to the inherent limitation of the MapReduce paradigm, viz., approximate result in quick time. This limitation infiltrates into the solutions that employ the paradigm for problem solving.

Henceforth in this paper, the 'RTS' refers to the R*-Tree construction principles adopted and also the R*-Tree constructed in a single node non-distributed environment; 'RTHEx' refers to the R*-Tree construction principles and also

the R*-Tree constructed in a Hadoop and MapReduce environment using the best techniques reported in the literature. 'RTHPr' refers to the R*-Tree construction principles adopted and also the R*-Tree constructed in a Hadoop and MapReduce environment using the improvements and refinements suggested in this paper.

IV. IMPROVISED R*-TREE CONSTRUCTION METHOD IN HADOP AND MAPREDUCE ENVIRONMENT

The improvised R*-Tree construction process retains all the three stages described in Fig. 1, but refines the computations and the sub processes each of them. Accordingly, the first amendment was made to the calculation of representational single key value for each of the points in the dataset. Along with the first function f_1, another function f_2 also was applied to the coordinates of the points, thus obtaining two keys for the same point. Both keys are Hilbert values, but each Hilbert curve corresponding to the functions f_1 and f_2 have either different starting positions for the same space filling curve or completely two different space filling curves starting at the same position. The necessity for this approach is explained below.

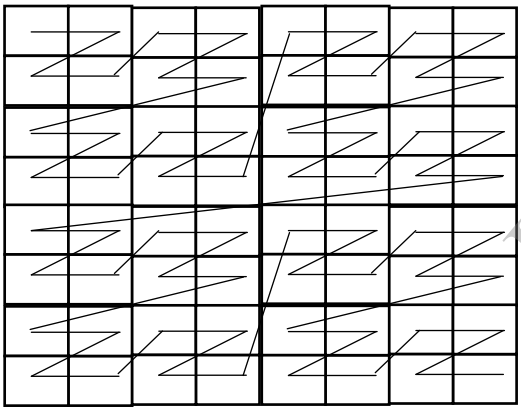

Fig 3 Illustration for the computation of f_1 and f_2

In Figure 3, the Hilbert values computed by f_1 for the cells (4, 4) and (1, 5) will fix them next to each other in the sequential ordering of cells' key values that does not reflect the actual Euclidean distance between the two cells. On the other hand the Hilbert values computed by f_1 for the cells (4, 4) and (4, 5) will not fix them next to each other in the sequential ordering of the cells' key values that does not reflect the near proximity of the two cells. But the positions in the sequential order of the Hilbert values of the cells (5, 5), (5, 6), (6, 5) and (6, 6) reflect their proximities. Hence f2 must be chosen carefully that eliminates this anomaly. Subsequently, both f_1 and f_2 were used for partitioning the dataset into pre-determined set of partitions in the second stage. The R-Trees that were obtained covered the near optimal spaces. In the third stage the final R-Tree was constructed using sort based bulk insertions.

V. EXPERIMENTS AND RESULTS

The experiments were conducted using hypothetical data sets of sizes 100K, 200K, 500K and 1000K and dimensionalities 2, 3 and 5. The data followed a random distribution. The results are displayed in Figure 4, Figure 5 and Table 1. The results show that the RTHEx method produces a R-Tree that is less efficient when compared with RTS method. This is due to the inherent drawbacks of Hadoop and MapReduce environment such as partitioning and redundant computations. RTHPr method reduces this deficiency and produces a R*-Tree that is almost nearer to the RTS method, but still not bettering the performance of RTS method. But this is acceptable in the context of following observations:

i. Hadoop and MapReduce are scalable for large volume of data. RTS methods ended up crashing the systems for datasets larger than 500K in 2D and for proportionally smaller datasets for higher dimensions.

ii. The time taken by RTHPr is less when compared with RTS (ignoring the network parameters).

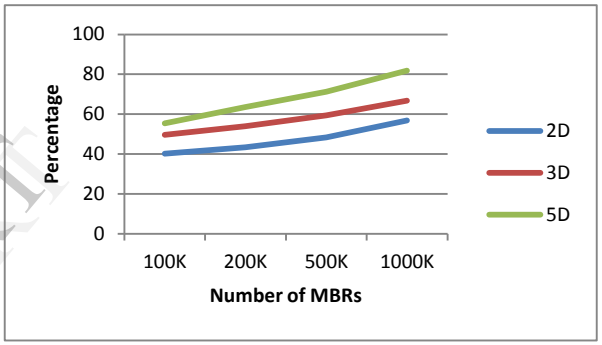

Fig. 4. Percentage increase in total area covered by RTHEx compared with RTS

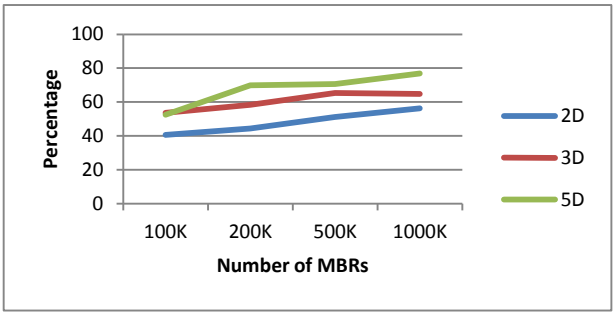

Fig. 5. Percentage increase in total perimeter length by RTHEx compared with RTS

TABLE I. Comparison of RTHEx and RTHPr

Number of Points	Dim	RTHEx		RTHPr	
		% increase in the total volume of space	% increase in the total volume of space	% increase in the total volume of space	% increase in the total volume of space
100K	2	40.23	40.63	9.11	9.20
	3	49.68	53.65	9.84	10.23
	5	55.34	52.57	11.31	10.98
200K	2	43.45	44.32	10.67	10.14
	3	53.96	58.28	11.84	11.49
	5	63.61	69.97	13.26	13.40
500K	2	48.34	51.24	12.39	12.76
	3	59.34	65.27	13.26	13.92
	5	71.29	70.58	15.11	15.72
1000K	2	56.78	56.21	15.85	16.17
	3	66.83	64.83	16.64	17.31
	5	81.88	76.97	17.97	17.43

VI. CONCLUSION

This paper in the first place has identified the gaps and the rooms for improvement in the construction of R*-Trees in Hadoop and MapReduce environments. Next, suggestions to filling these gaps and ameliorations to the existing processes in various stages have been presented. The effectiveness of the proposals has been verified with experimental results.

REFERENCES

[1] J. Nievergelt, "Binary search trees and file organization," ACM Computing Surveys, vol. 6, no. 3, pp. 195-207, 1973.

[2] D. Comer, "The ubiquitous B-tree," ACM Computing Surveys, vol. 11, no. 2, pp. 121-137, 1979.

[3] Guttman, "R-trees: A dynamic index structure for spatial searching," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 47-57, 1984.

[4] N. Beckmann, H. -P. Krieger, R. Schneider and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 322-331, 1990.

[5] V. Gaede and 0. Guenther, "Multidimensional access methods," ACM Computing Surveys, vol. 30, no. 2, pp. 170-23 1, 1998.

[6] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos and Y. Theodoridis, "R-trees have grown everywhere," Technical Report, Available at http://citeseer.ist.psu.edu/706599.html (2003)

[7] Faloutsos and I. Kamel, "Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension," Proceedings of the 13th ACMPODS Conference, pp. 4-13, 1994.

[8] I. Kamel and C. Faloutsos, "Hilbert R-tree: An improved R-tree using fractals," Proceedings of the 20th International Conference on Very Large Databases, pp. 500-509, 1994.

[9] S. Brakatsoulas, D. Pfoser and Y. Theodoridis, "Revisiting R-tree construction principles," Proceedings of the 6th ADBIS Conference, pp. 149-162, 2002.

[10] L. Chen, R. Choubey and E. A. Rundensteiner, "Bulk-insertions into R-trees using the small-tree-large-tree approach," Proceedings of the 6th ACM GIS Conference, pp. 161-162, 1998.

[11] Daniar Achakeev, Marc Seidemann, Markus Schmidt and Bernhard Seeger, "Sort-based parallel loading of R-trees," Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, pp. 62-70, 2012.

[12] F.Sagayaraj Francis and P.Thambidurai, "Efficient Physical Organization of R-Trees Using Node Clustering", Journal of Computer Science, vol. 3, no. 7, pp. 506-514, 2007.

[13] F.Sagayaraj Francis and P.Thambidurai, "An Efficient Method to Estimate the Number of Node Accesses in R*-trees for Window Queries", Journal of Advance Research in Computer Engineering, vol. 1, nos. 1-2, pp. 9-24, 2007.

[14] X. Wei and K. Sezaki, "DHR-Trees: A distributed multidimensional indexing structure for P2P systems," Proceedings of the 5th International Symposium on Parallel and Distributed Computing, pp. 281-290, 2006.

[15] Dafei Yin, Bin Chen, Yu Fang and Zhou Huang, "Fully distributed R-tree for efficient range query dissemination in peer-to-peer spatial data grid," Proceedings of the Conference on Geoinformatics, Geospatial Information Science, doi:10.1117/12.761369, 2007.

[16] Ariel Cary, Zhengguo Sun, Vagelis Hristidis and Naphtali Rishe, "Experiences on Processing Spatial Data with MapReduce," Proceedings of the 21st International Conference on Scientific and Statistical Database Management, pp. 302-319, 2009.

[17] Himanshu Gupta, Bhupesh Chawda, Sumit Negi, Tanveer A. Faruquie, L. V. Subramaniam and Mukesh Mohania, "Processing multi-way spatial joins on map-reduce," Proceedings of the 16th International Conference on Extending Database Technology, pp. 113-124 , 2013.

[18] Yi Liu, Ning Jing, Luo Chen and Huizhong Chen, "Parallel bulk-loading of spatial data with MapReduce: An R-tree case," Wuhan University Journal of Natural Sciences, vol. 16, No. 6, pp. 513-519, 2011.

[19] Christos Doulkeridis and Kjetil Norvag, "A Survey of Large-Scale Analytical Query Processing in MapReduce," The VLDB Journal, DOI: 10.1007/s00778-013-0319-9, 2013.