A Buffer Model for Evaluating R-tree Performance

Scott T. Leutenegger Mario A. Lopez
Mathematics and Computer Science Department
University of Denver Denver, CO 80208-0189
{leut,mlopez}@cs.du.edu

1 Introduction

R-trees [3] are a common indexing technique for multi-dimensional data and are widely used in spatial and scientific databases. By storing the bounding boxes of arbitrary geometric objects, such as points, polygons, or more complex objects, R-trees can be used to determine which objects intersect a given query region. Several variants have been proposed to improve performance [1, 5, 9].

Performance studies of various R-tree algorithms, have used the number of nodes accessed as the primary metric. In real databases some portion of the tree is buffered in main memory. Since such buffering can significantly affect performance, we postulate that performance prediction should be based on number of I/Os required to satisfy a query. To this end we present an analytical model that predicts the number of disk accesses for an LRU buffer manager given the minimum bounding rectangles (MBRs) of the R-tree nodes and a buffer size. Our model can be used to evaluate the quality of any R-tree update operation, such as node splitting policies [3] or packing algorithms [4, 6, 8], as measured by query performance of the resulting tree. The model is very accurate and simple to understand, making it easy for researchers to integrate it into their studies. It can easily be modified to accommodate different buffer management policies – for example, keeping the top few levels of the R-tree in the buffer. Furthermore, the model is not only applicable to R-trees, but can easily be modified to predict B-tree performance.

Our buffer model is a simple analytic model, but our overall performance model is hybrid. We start with a specific algorithm to create actual R-trees for various data sets. Then, we compute the MBRs of tree nodes and use these as input to our buffer model.

[0]

SIGMETRICS 96-5/96 Philadelphia, PA, USA

2 Buffer Model Description

In deriving our model we use the following notation:

T = Specific R-Tree being modeled
M_i = Number of nodes at the ith level of T
M = Total number of nodes in T, i.e., $\sum_{i=0}^{H} M_i$
R_{ij} = jth node at the ith level of T
A_{ij} = Probability that a query (point or region) will require access to R_{ij}
B_{ij} = Number of accesses to R_{ij}
N = Number of queries performed so far
N^* = Expected number of queries to fill the buffer
B = Buffer size
$D(N)$ = Number of distinct nodes (at all levels) accessed in N queries

For random queries and objects within the unit square, Kamel and Faloutsos [4] argue that A_{ij} is the area of R_{ij} for point queries, and the area of $R_{ij} \oplus Q$ for region queries of size Q[1].

Bhide *et al* [2] analyze the LRU buffer replacement policy for databases consisting of a number of partitions with uniform page access within each partition. While modeling performance during buffer warm-up they observed that the buffer hit probability at the end of the warm-up period is a good estimator of the steady state buffer hit probability. We conjectured that similar behavior would occur in the context of R-trees when applying the LRU replacement policy, namely, that the steady state buffer hit probability is virtually the same as the buffer hit probability when the buffer *first* becomes full. Our conjecture was verified experimentally as discussed in Section 3.

Under uniformly distributed point queries, the probability of accessing node R_{ij} while performing a query is A_{ij}.

[1] $A \oplus B$ denotes the Minkowski sum of sets A and B. So $R_{ij} \oplus Q$ is just R_{ij} expanded by the extents of Q along each dimension.

	Rtree1			Rtree2			Rtree3		
bufsize	simul	model	% dif	simul	model	% dif	simul	model	% dif
50	2.3120	2.2682	1.189	2.2060	2.1651	1.185	1.5892	1.5807	0.53
100	1.7008	1.6862	0.86	1.6288	1.6166	0.75	1.2336	1.2275	0.49
200	1.1627	1.1599	0.24	1.1194	1.1172	0.20	0.8720	0.8710	0.11
400	0.9347	0.9340	0.07	0.9009	0.9007	0.02	0.6944	0.6944	0.00
1000	0.4701	0.4700	0.02	0.4534	0.4532	0.04	0.3551	0.3551	0.00
1400	0.1770	0.1769	0.06	0.1714	0.1712	0.12	0.1368	0.1368	0.00

Table 1: Validation: average number of disk accesses per query for model and simulation.

Accordingly, the probability that R_{ij} is not accessed during the next N queries is $P[B_{ij} = 0|N] = (1 - A_{ij})^N$. Thus, $P[B_{ij} \geq 1|N] = 1 - (1 - A_{ij})^N$ and the expected number of *distinct* nodes accessed in N queries is,

$$D(N) = \sum_{i=0}^{H} \sum_{j=1}^{M_i} P[B_{ij} \geq 1|N] = M - \sum_{i=0}^{H} \sum_{j=1}^{M_i} (1 - A_{ij})^N$$

Note that $D(0) = 0 < B$ and $D(1) = \sum A_{ij}$, (which may or may not be bigger than B). The buffer, which is initially empty, first becomes full after performing N^* queries, where N^* is the smallest integer that satisfies $D(N^*) \geq B$. The value of N^* can be determined by a simple binary search.

While the buffer is not full the probability that R_{ij} is in the buffer is just $P[B_{ij} \geq 1]$. The probability that a random query requires a disk access for R_{ij} is $A_{ij} \cdot P[B_{ij} = 0]$.

Since the steady state buffer hit probability is approximately the same as the buffer hit probability after N^* queries, the expected number of disk accesses for a point query at steady state is

$$\sum_{i=0}^{H} \sum_{j=1}^{M_i} A_{ij} \cdot P[B_{ij} = 0|N^*] = \sum_{i=0}^{H} \sum_{j=1}^{M_i} A_{ij} \cdot (1 - A_{ij})^{N^*}$$

In order to model a buffer management policy that pins the top few levels of the R-tree in the buffer, just reduce the number of buffer pages by the number of pages in these levels and omit the top levels from the model.

3 Model Validation

We validated the model by comparing it with simulation. The simulation models an LRU buffer and, like the model, takes as input the list of the MBRs for all nodes at all levels. It then generates random point queries in the unit square and checks each node's MBR to see if it contains the point. Confidence intervals were collected using batch means with 20 batches of 1,000,000 queries each, resulting in confidence intervals of less than 3% at a 90% confidence level.

We ran comparisons for three different R-trees and 6 different buffer sizes for each tree. Each of the R-trees has 1668 nodes, but the MBRs of the nodes are different, as produced by the three different tree construction algorithms [7]. In Table 1 we report the average number of pages required from disk per point query as predicted by the simulation, model, and the percent difference (relative to the simulation). All of the results are within 2%, which is less then the confidence intervals returned from the simulation. Other comparison experiments not shown resulted in agreement within 2% or less. Simulation of region queries gave similar results.

References

[1] N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger, "The $R^\star$-tree: an Efficient and robust Access Method for Points and Rectangles", Proc. ACM SIGMOD, p. 323-331, 1990.

[2] A. Bhide, A. Dan, D. Dias, "A Simple Analysis of LRU Buffer Replacemtn Policy and Its Relationship to Buffer Warm-up Transient", Proc. IEEE Data Eng., 1993.

[3] A. Guttman, "R-trees: a Dynamic Index Structure fo Spatial Searching", Proc. ACM SIGMOD, p. 47-57, 1984.

[4] I. Kamel, C. Faloutsos, "On Packing R-trees", Proc. 2nd International Conf. on Information and Knowled Management (CKIM-93), p. 490-499, Arlington, VA, November 1993.

[5] I. Kamel, C. Faloutsos, "Hilbert R-tree: An improved R-tree Using Fractals", Proc. International Conf. on Very Large Databases 1995 (VLDB-95).

[6] S.T. Leutenegger, J. Edgington, M.A. Lopez, "Efficient Bulk Loading of R-trees", University of Denver Computer Science Technical Report #95.1. To be submitted for publication.

[7] S.T. Leutenegger, M.A. Lopez, "A Modeling Study of R-tree Packing Algorithms", University of Denver Computer Science Technical Report #95.2. To be submitted for publication.

[8] N. Roussopoulos, D. Leifker, "Direct Spatial Search on Pictorial Databases Using Packed R-trees", Proc. ACM SIGMOD, 1985.

[9] T. Sellis, N. Roussopoulos, C. Faloutsos, "The R+ Tree: A Dynamic Index for Multidimensional Objects", Proc. 13th International Conference on Very Large Databases (VLDB-87), p. 507-518, September 1987.