

Incrementally Distributed B⁺ Trees: Approaches and Challenges

Pallavi Tadepalli
Overstock.com
6350 South 3000 East
Salt Lake City, UT 84047
1-801-947-3864

patadepalli@overstock.com

H. Conrad Cunningham
University of Mississippi
201 Weir Hall
University, MS 28677
1-662-915-5358

cunningham@cs.olemiss.edu

ABSTRACT

B⁺ trees have proven efficient and effective in the role of indexes for data stored in databases. With the explosion in the number of datasets being stored in a distributed manner, a scalable and efficient index is needed to locate data. In this paper, the issues in designing a distributed B⁺ tree are examined with a specific emphasis on incrementally distributing the tree across a network.

Categories and Subject Descriptors

E.1 [Data Structures]: Distributed data structures

General Terms

Algorithms, Design.

Keywords

Distributed B⁺ tree, Grid Nodes

1. INTRODUCTION

Data are ubiquitous. Data sets may be very large and either concentrated at a single site or spread across a network that could be a grid, peer-to-peer, sensor or some other kind. Thus there can be several independent data sources that can be accessed without a central authority. When there are distributed data sources in a network with limited or no central authority, searching for the appropriate data sources that contain relevant data is an issue. Thus the issue can be described as a problem of data location. To address the issue there needs to be a comprehensive index designed and implemented as a modified, distributed B⁺ tree. This can be used as a scalable indexing and searching mechanism to locate distributed data. To distribute the B⁺ tree, replicating a part of the tree on various grid nodes is essential along with keeping a low overhead in maintenance of various copies.

2. PROBLEM DEFINITION

A distributed data structure is effective in “managing large

volumes of distributed and dynamically changing data” [14]. According to [4], the problem of data location on a grid can be solved by an efficient index that is created using complex distributed data structures that are variants of the B-tree [2] like B* trees. If an index was created for each data item spread over hundreds of nodes on a grid, the size of the index would be huge. Essentially it is like creating an index for a very large database. B and B* trees have been used to create traditional database indexes. These indexes are generally used to search for a record or range of records given a particular key. Storing an index of such a magnitude in a central location creates an access bottleneck. In distributed computing with large volumes of dynamically changing data, high-throughput access is essential. Distributing a B⁺ tree enables it to handle concurrent data access requests in parallel which could lead to higher throughput. The assumption is that the results will be combined at the requesting “client” site.

By distributing a search tree, increased parallelism on various operations is expected. Yet, the throughput is limited by the single rooted structure of the tree since all operations at the root of the tree. This scenario is known as the *root bottleneck*. By replicating the root and other nodes of the search tree there should be an increase in efficiency, decrease in latency and improvement in performance. In a distributed environment, multi-user access to the tree can cause concurrency issues such as contention management which are solved by the use of concurrent algorithms. The manner of distributing the search tree could affect node placement, the number of replicas needed and the manner of replication. Once tree nodes are replicated, it is necessary to keep track of the various tree nodes and copies. Thus node replication and replica coherence are inherent requirements in a distributed structure to support concurrent operations.

3. BACKGROUND

3.1 Distributed Indexing Structures

The problem of indexing distributed data in a decentralized network has been studied in many different ways and there have been many attempts to solve the problem in various distributed storage systems using a diversity of distributed data structures.

An existing distributed data structure is the Distributed Hash Table (DHT) [18]. DHTs were invented as a result of the widespread growth and use of peer-to-peer file sharing systems like Gnutella [7], Napster [17] and KaZaa [12]. The goal of DHTs has been to provide efficient location of data sources. Given a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACMSE'09, March 19–21, 2009, Clemson, SC, USA.

©2009 ACM 978-1-60558-421-8/09/03 ...\$10.00

