Similarity Searching: Towards Bulk-loading Peer-to-Peer Networks

Vlastislav Dohnal Jan Sedmidubský Pavel Zezula David Novák
Faculty of Informatics
Masaryk University
Botanická 68a, Brno, Czech Republic
{dohnal, xsedmid, zezula, david.novak}@fi.muni.cz

Abstract

Due to the exponential growth of digital data and its complexity, we need a technique which allows us to search such collections efficiently. A suitable solution seems to be based on the peer-to-peer (P2P) network paradigm and the metric-space model of similarity. During the building phase of the distributed structure, the peers often split as new peers join the network. During a peer split, the local data is halved and one half is migrated to the new peer. In this paper, we study the problem of efficient splits of metric data locally organized by an M-tree and we propose a novel algorithm for speeding the splits up. In particular, we focus on the metric-based structured P2P network called the M-Chord. In experimental evaluation, we compare the proposed algorithm with several straightforward solutions on a real network organizing 10 million images. Our algorithm provides a significant performance boost.

1. Introduction

Current data processing applications use data with considerably less structure and much less precise queries than traditional database systems. An example is multimedia data like images or video clips that offer query-by-example search. This situation is what has given rise to similarity searching. The most general approach to similarity search, still allowing construction of index structures, is modeled in a metric space. Many index structures were developed and surveyed recently [25, 30]. The latest efforts in this area focus on the design of distributed access structures which exploit more computational and storage resources [2, 13, 4, 3] in order to cope with the problem of exponential growth of digital data that must be processed. A suitable solution arises from peer-to-peer (P2P) networks which are scalable.

In this digital-explosion age, a P2P index structure needs to organize a non-trivial amount of data, e.g., tens of millions of data objects (records) or even more. In this respect, each peer of this network has to also maintain a large number of data objects, so the search within the peer is usually sped up by a local (centralized) index structure. During the operation of this P2P structure, peers get overloaded due to the insertion of new data or an increasing number of queries processed. This situation is handled by peer splits. Consequently, the data content of the overloaded peer must be halved, that is the local index structure must be split.

In this paper, we concentrate on effective splits of local indices. We study this problem on the P2P index structure called the M-Chord [22] which uses the M-tree [8] to index peer's local data. This M-tree is enriched with extensions of the Slim-tree [28] and the PM-tree [27]. The paper is structured as follows. After related work, Section 2 contains the description of the architecture of our image retrieval system. In Section 3, we present four algorithms to split peers. The paper concludes with a performance comparison.

1.1. Related work

Multimedia data are often modeled as high-dimensional spaces, so a variety of high-dimensional index structures were proposed, e.g., the R-tree, the TV-tree, the SS-tree or the X-tree. A more generic approach to index such data is to exploit the metric space model, e.g. the M-tree [8], the Slim-tree [28], the GNAT [6], the SAT [21] or the D-index [11].

Due to the need to organize large databases, many techniques to bulk-load index structures were proposed. The Hilbert R-tree [17] is probably the first modification of R-tree aiming at optimized insertion of a large number of data records. It is based on sorting the elements first and then building the tree bottom-up. Earlier, similar techniques were proposed for the B-tree [18] and the quadtree [23, 15], for illustration. A bulk-loading algorithm which also optimizes the tree in order to improve search efficiency is proposed in [5]. It is demonstrated on the X-tree but it can be applied to any R-tree-like structure. Another variant of R-tree called the Priority R-tree [1] is slightly less efficient