

Scalable Data Access in P2P Systems

Using Unbalanced Search Trees

Extended Abstract

Karl Aberer

Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
karl.aberer@epfl.ch

1 Introduction

With the appearance of Peer-2-Peer system the interest in scalable and decentralized data access structures is increasingly attracting interest. Such access structures emerge from the interaction of multiple peers. Each peer is storing a small fraction of the data and maintains a routing table to neighboring peers to forward search requests that it can not answer. Different organisations of such routing schemes have been proposed. Most frequently they are constructed based on an underlying abstract model of binary search trees. This approach has been pursued by us [1], but also in [3], [4] and [5]. As a result queries can be answered with at most $O(\log N)$ messages, where N is the number of peers, since the trees are balanced and thus the path length is limited.

However, with skewed data distributions, this approach may lead to very unevenly distributed workloads for the peers. In classical database indexing this problem is addressed, by using balanced tree structures like B-Trees, rather than binary search trees. In this paper we will demonstrate that the use of balanced tree structures for data indexing is not necessarily required in a P2P environment. Thus we can continue to follow the approach of scalable binary search trees, rather than adapting DB-oriented indexing structures like B-Trees to a decentralized environment, which might pose substantial problems in terms of coordination among peers.

The key observation that we make is, that in a P2P environment not the depth of a search tree is the critical measure for performance, but the number of messages exchanged for a search. Our assumption is that the cost of long traversals of paths in trees that are made locally can be neglected as compared to the cost of message exchanges.¹ We will show that even if the (virtual) search trees, as implicitly represented in the structure of the routing tables, tend to be extremely unbalanced, the number of messages used to process a query still scales gracefully when assuming that the tree depth, the data distribution, and the distribution of peers associated with leaves of the tree are correlated. This result shows that in distributed, and in particular decentralized data management, we have to rethink some of the very fundamental assumptions on data management, that have evolved in the context of mostly centralized databases.

2 System Model and Access Structure

We give an informal introduction into the decentralized, scalable access structure on which our subsequent discussion is based. More details are found in [1, 2]. Similar structures have also been proposed in [3], [4] and [5].

¹In case this is not true, we still may consider to use any local indexing structure to speed up those traversal.

We assume that a set of peers P of size N is given. Each peer $p \in P$ stores data identified by a binary key $b = b_1 \dots b_{k_p}$, $b_i \in \{0, 1\}$. In other words, each peer is associated with a key b . Note that the key length may differ for different peers. For each prefix $b_1 \dots b_l$ of b of length l , $l = 1, \dots, k_p$ the peer p maintains a routing table, which consists of a set of references to other peers p' , that have the same prefix of length l , but a different value at position $l + 1$. We will call these references to other peers, p 's references at level l . These references are used to route search requests for keys with prefix $b_1 \dots b_l$ that do not match at position $l + 1$ with the key b associated with p .

A search request can start at any peer and proceeds in the obvious way: starting from the first bit the search key is compared bit by bit to the peers' keys. When the local key matches the next bit is compared, if it does not match the query is forwarded to one of the peers from the routing table that have at the next position a matching bit.

In [1] we have shown that there exists an efficient decentralized algorithm for constructing such an access structure based on random interactions among the peers. The basic idea is that whenever peers meet they refine their keys into opposite directions. If a global maximal keylength k_{max} is given, this algorithm results in a uniform distribution of keys over the peers. Each key will be associated on average with $\frac{N}{2^{k_{max}}}$ peers. From a global viewpoint the resulting access structure consists of fully balanced binary search trees, where each peer supports the search along one path from the root to a leaf of the tree. Thus the total amount of resources scales linearly in number of peers and the search cost, both in time and number of messages generated scales logarithmically.

3 Skewed Data Distributions and Unbalanced Search Trees

With skewed data distributions, as they can be expected in realistic applications, using a balanced binary tree for data distribution and search would imply an unevenly distributed workload among the peers. Therefore we use a modified version of the construction algorithm, that has been introduced in [1]. This modified algorithm results in unbalanced search trees. We assume that the peers store already some data that reflects the actual data distribution. Throughout the construction process, whenever peers meet, they not only extend their corresponding keys, but they also exchange their data correspondingly. In that way the data is globally disseminated. We no longer make use of a maximal keylength parameter k_{max} . Rather peers extend their key only if sufficient data, i.e. a minimal number of data items, is known to them to justify a further extension of the key. This has the following effects:

1. Each peer will store on average the same number of data items and thus require comparable resources for storage.
2. The lengths of the keys that are supported by the peers are related to the density of data items in the region around the key. Thus, for key regions where many data items occur, the keys tend to grow longer than in key regions with fewer data items.
3. No global parameter k_{max} is required. It is replaced by the global data distribution, which we can assume to be available. This is a more natural form of global knowledge to exploit in the search tree construction than global knowledge on the maximal keylength. Thus we further reduce global knowledge required for index construction and increase the degree of self-organization.

As a result we obtain from a global viewpoint an unbalanced, binary search tree, where for each of its leaves (corresponding to keys) on average the same number of peers is associated with and each peer at a leaf node has to store on average the same number of data items. Simulations show that this behavior can in fact be achieved.

This poses however another question as it is no longer guaranteed that search paths remain short. In fact, the trees may now contain paths that in the worst case have length $O(N)$ which would defeat the purpose of having a search tree.

4 Search Path Length vs. Number of Search Messages

At this point we have to consider the physical characteristics of P2P systems. Actually we are not so much interested in the absolute length of the search path from the root to the leaf of a binary search tree, but rather in the number of messages that need to be exchanged in order to find a data item. This number will be lower than the path length if we can traverse multiple tree nodes at the same peer. Local processing of the traversal of multiple tree nodes is not critical, as we may assume that the cost of local processing is by orders of magnitude faster than message exchanges between peers. Even if the local processing cost were non-negligible it could be sped up by locally using an indexing structure.

Therefore in the following we consider only the number of message exchanges in a search process. The main finding is: even in the case a search tree is extremely unbalanced, the number of messages will remain low, i.e. $O(\log N)$, assuming the search tree has been constructed such that each peer stores about the same amount of data and replicas are also evenly distributed. This property is on average achieved by our construction algorithm.

To show the claim we analyze a search process. We consider one specific key at a leaf node of the search tree. A search for this key starts at some peer. In each step (going from one bit of the key to the next) either the peer itself has a matching bit, and therefore no message is required, or a message has to be sent to another peer, following one of the references stored at the corresponding level of the routing table. There the same process continues until the leaf level is reached.

Let us look at the search paths and the number of peers responsible for a fixed key. At level 0 we have $n_0 = N$ peers, namely all peers are responsible for the empty key. Of these peers n_1 do not match the first bit, whereas $n_0 - n_1$ do. Thus, with probability $\frac{n_1}{n_0}$ a message is required, in order to arrive at a peer with the first bit matching. Then we look at the next bit. Again n_2 peers, among the $n_0 - n_1$ remaining peers that match the first bit, will not match the second bit. If we assume that the distribution of peers within routing tables is the same as the global distribution of peers (which is reasonable to do) with probability $\frac{n_2}{n_0 - n_1}$ a message is sent to find a peer matching the bit. We can continue now this process for k steps if the key has length k . The whole process is illustrated in the Figure 1.

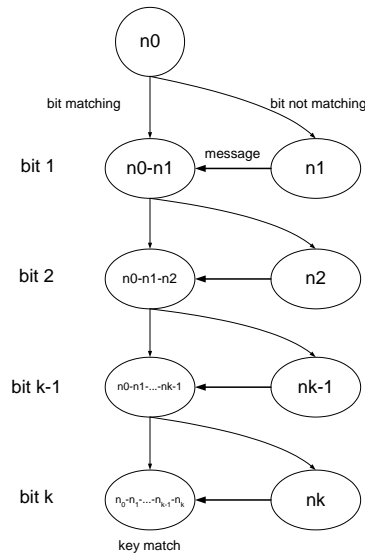


Figure 1: Search process

Adding up the expected number of messages in each step results in the expected total number of messages for the search process:

$$\sum_{i=1}^k \frac{n_i}{n_0 - \dots - n_{i-1}}$$

for an arbitrary sequence of positive numbers n_1, \dots, n_k with $\sum_{i=1}^k n_i < n_0$. In order to estimate the value of the expected number of messages we can proceed as follows. Each term can be considered as the area of a rectangle. We have arranged these rectangles as in Figure 2.

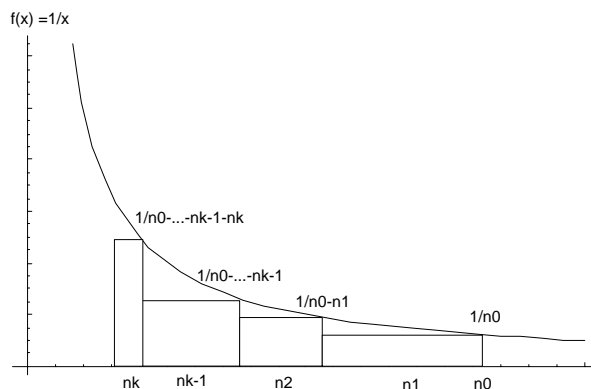


Figure 2: Deriving an upper bound for the number of expected messages

One can see from this Figure, that the rectangles fall exactly under the curve of the function $f(x) = \frac{1}{x}$. Therefore the total area of the rectangles is smaller than

$$\int_r^{n_0} \frac{1}{x} dx = \log n_0 - \log r$$

where $r = n_0 - \sum_{i=1}^k n_i \geq 1$. This results in a bound

$$\sum_{i=1}^k \frac{n_i}{n_0 - \dots - n_{i-1}} < \log n_0 = \log 2 \log_2 N.$$

which shows that independently of how the search tree is unbalanced the number of messages will always remain of order $O(\log N)$. This bound is in fact very tight. For example, it is close to $\frac{1}{2} \log_2 N$, which is the expected number of messages if the search tree were exactly balanced.

5 Conclusion

We have shown in this paper that the property of search trees being balanced is of a quite different relevance if we look at decentralized system environments and take into account the physical characteristics of them. The result is not contradicting findings on the impossibility of constructing distributed balanced search trees [6], rather it relativizes the importance of the question of having balanced search trees.

References

- [1] K. Aberer, *P-Grid: A self-organizing access structure for P2P information systems* Proc. of the Ninth International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, 2001.

- [2] K. Aberer, M. Puceva, M. Hauswirth, R. Schmid, *Improving Data Access in P2P Systems*, to appear: IEEE Internet Computing, Jan./Feb. 2002.
- [3] S. Rhea et al. *Maintenance-free Global Data Storage*, IEEE Internet Computing, vol.5, no.5, Sept./Oct.2001, pp. 40-49.
- [4] C. Greg Plaxton, Rajmohan Rajaraman, Andra W. Richa, *Accessing Nearby Copies of Replicated Objects in a Distributed Environment*, Proceedings of the 9th Annual Symposium on Parallel Algorithms and Architectures, pp. 311-20, 1997.
- [5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan: *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications* Proc. of the ACM SIGCOMM, 2001.
- [6] B. Kröll, P. Widmayer *Balanced Distributed Search Trees Do Not Exist* WDAS 95, p 50-61, 1995.