Implementing Range Queries with a Decentralized Balanced Tree over Distributed Hash Tables

Nuno Lopes* and Carlos Baquero

CCTC-Department of Informatics
University of Minho
Braga, Portugal

Abstract. Range queries, retrieving all keys within a given range, is an important add-on for Distributed Hash Tables (DHTs), as they rely only on exact key matching lookup. In this paper we support range queries through a balanced tree algorithm, Decentralized Balanced Tree, that runs over any DHT system.

Our algorithm is based on the B^+-tree design that efficiently stores clustered data while maintaining a balanced load on hosts. The internal structure of the balanced tree is suited for range queries operations over many data distributions since it easily handles clustered data without losing performance.

We analyzed, and evaluated our algorithm under a simulated environment, to show it's operation scalability for both insertions and queries. We will show that the system design imposes a fixed penalty over the DHT access cost, and thus inherits the scalability properties of the chosen underlying DHT.

1 Introduction

Distributed Hash Table (DHT) systems [1,2,3,4] are used as efficient distributed dictionary implementations, offering a scalable and robust P2P framework that efficiently locates objects given a key [5,6]. However, such efficiency is achieved by an exact key matching lookup interface. The discrete key lookup interface uses an hash function on the key value to locate objects. This hash function removes locality properties from keys which restricts it's use for range queries. A range query consists in retrieving all keys that fall within a specific range interval. Range query is a desired feature when using data that is indexed by contiguous values (consider for example, numeric spatial coordinates).

Previous systems have offered range queries by either using specially designed structures [7,8,9] or building on top of generic DHTs [10,11,12]. Because the first class of systems is bound to some particular basic storage structure, it offers a limited solution that may not be as efficient as some DHT systems are. This

* Supported by a Ph.D. Scholarship from FCT-Foundation of Science and Technology, the Portuguese Research Agency.

T. Enokido, L. Barolli, and M. Takizawa (Eds.): NBiS 2007, LNCS 4658, pp. 197–206, 2007.

makes the second class of systems, building a tree structure over a generic DHT, the most flexible choice. By building on a generic DHT, one can choose the best DHT implementation available for the system, while maintaining the range query functionality. However, recent structures available in the literature: Prefix Hash Tree (PHT) [10] and Distributed Segment Tree (DST) [12], are sensitive to clustered data.

Clustered data is common on real data sets, in particular when data depicts geographical placement of items that are tied to human activity. For instance, the concentration of WiFi access points is clustered around cities and along roads [10], so that sharing access point locations and querying for nearby access points will yield a response depicting clustered data.

This steams from population concentration patterns, where clustered data typically follows a power-law distribution, or a combination of power-laws centered on several focus points [13]. This common setting depicts a few higher density key regions while most of the data is sparsely distributed across the key domain.

In this paper we show that the Decentralize Balance tree (DEB tree) algorithm, an algorithm based on the B^+-tree design [14], offers a structure suitable for storing clustered data on block oriented storage (in this case a DHT) while supporting range queries without loss of performance.

The algorithm is capable of running on top of any generic DHT without incurring in a significant overhead. Insertions can be reduced to O(1) complexity in terms of DHT operation requests, if caching is used at clients. Each DHT request cost depends on the DHT implementation selected. In this sense, the scalability of the tree design closely follows the scalability properties of the used DHT.

Query cost depends on the data stored on the index rather than on the range size. Additionally, it is possible to parallelize the query operation, reducing latency to a logarithmic factor on the stored data size in terms of DHT operations.

2 Related Work

Related work can be divided into two groups: range query systems with specific underlying structures and range query systems using a tree structure over a generic DHT interface. Due to space restrictions we will focus on the later group and only provide a brief mention to some systems in the first group.

Mercury [7] supports multi-attribute range queries using a circular overlay, similar to Chord, but without key hashing, so that locality is preserved. Skip graphs [8] are a generalization of skip lists in which nodes are part of distributed linked lists that form a distributed binary tree. Baton [15] builds a binary balanced tree structure where each tree node is mapped into a peer host and maintains connections to the peers containing tree node neighbours.

All previous systems maintain a specific routing algorithm that offers logarithmic cost in terms of network hops to access data. Our DEB-tree algorithm assumes a generic DHT implementation is used. Such assumption enables the