# Building Inverted Indexes Using Balanced Trees Over DHT Systems

Nuno Lopes
PhD student
University of Minho - Portugal
nuno.lopes@di.uminho.pt

Carlos Baquero
University of Minho - Portugal
cbm@di.uminho.pt

Distributed Hash Table (DHT) systems are scalable and efficient data structures for object storage and location using a simple put/get interface. These systems place objects over a very large set of hosts using a multitude of algorithms in order to distribute objects uniformly among hosts using logarithmic (or lower) costs for routing table sizes and message hops [1, 2]. However, these systems assume that object size (storage load) and popularity (communication load) follow an uniform distribution. When unbalanced data is used on a DHT, hotspots are created at some specific (random) hosts. Although one might argue that storage is not a critical resource, due to the current trend on secondary storage capacity, storing such large objects creates network bottlenecks, which in turn may limit data availability on very large heterogeneous systems.

A typical example of unbalanced data is a textual inverted index created from a collection of documents. When implementing an inverted index over a DHT system directly, one maps index keywords to DHT keys and their posting lists (document references) as the DHT values associated with the keys. Since an inverted index follows a Zipf distribution where some keywords are far more popular than others, hosts which store the popular words will have far more load than the others [3]. Later implementations handle hotspots by using a random factor at the DHT key in order to spread load over a DHT range instead of assigning it to a unique key [4]. However, even when using this method there is the possibility of overloading individual DHT keys.

Having identified a limitation of current DHTs, the inability to handle unbalanced data, we propose a new algorithm to uniform both storage and network loads for each DHT key. Our distributed algorithm is based on balanced trees which were designed to split data into bounded sized blocks while minimizing the number of block accesses per operation. These design goals, which were imposed by secondary storage, are also valid for very large systems where network latency and bandwidth are important limitations. Unlike other tree-based routing infra-structures, which replace the DHT algorithms, we designed our algorithm to use a key based routing interface. Such decision allows our algorithm to use any DHT system as a basic building block since all these systems can provide a simple key based routing interface. Furthermore, by using a balanced tree rather than a Prefix Hash Tree our data structure grows as a function of the object size, creating well balanced trees, instead of growing spuriously due to the unbalanced prefix variations of data [5].

Our algorithm is capable of storing unbalanced data over a generic DHT without loosing scalability. By deploying a well known structure, B+ trees, over generic DHTs, we are building a generic index functionality that can be used as a basic building block for new large-scale P2P applications.

**References**

[1] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, (Germany), 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of the ACM SIGCOMM'01 Conference*, pp. 161–172, 2001.

[3] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *Proceedings of the 4th ACM/IFIP/USENIX International Middleware Conference*, (Brazil), 2003.

[4] C. Tang and S. Dwarkadas, "Hybrid global-local indexing for efficient peer-to-peer information retrieval," in *Proceedings of First Symposium on Networked Systems Design and Implementation*, (San Francisco, USA), March 2004.

[5] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, J. Hellerstein, and S. Shenker, "A case study in building layered dht applications," in *Proceedings of the ACM SIGCOMM'05 Conference*, pp. 97 – 108, 2005.