Building Inverted Indexes Using Balanced Trees Over DHT Systems

Nuno Lopes and Carlos Baquero

{nuno.lopes,cbm}@di.uminho.pt

DI/CCTC, University of Minho, Braga, Portugal

DHTs

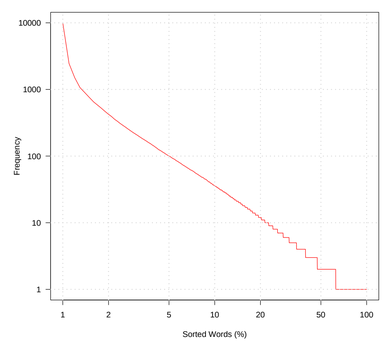
- Scalable and decentralized structured overlay networks
- Highly efficient in routing/locating an object given it's key.
- Distribution of keys (and objects) over hosts relies on hash function over an identifier space. However, object size is assumed to fit a single host.

Inverted Index

- Builds relations between keywords and document location lists (for textual data).
- Index lists follow Zipf distribution where popular keywords have very large document location lists and most keywords (majority) have smaller lists.

The Problem

Objects containing the document locations for popular keywords are sufficiently large to create storage hotspots at some hosts. Since each object is assigned to a single key, DHT key based load balancing techniques are incapable of splitting the object through several hosts. Furthermore, caching techniques only reduce network load for query operations and not handling network load during insert operations.



Frequency of text keywords over raw text documents (follows Zipf distribution).

Distributed Balanced Tree Algorithm

- We propose a new distributed tree algorithm based on B-link trees (and B^+ trees) suitable for storing unbalanced data (e.g. textual inverted index) over DHT based systems.
- We used the following index model: each index keyword is assigned a distributed tree and document locations for that keyword are stored as tree items.
- Each tree block is stored in the DHT by it's globally unique key.

The algorithm:

- Uses a Key Based Routing interface, therefore running over any DHT system.
- Is completely decentralized, such that any tree block may execute operations locally at the host containing it.
- Provides high availability because client operations on tree blocks never block and can be executed concurrently with tree management operations.

System Architecture

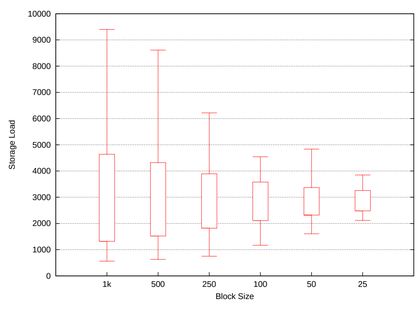
- Index layer: provides an inverted index interface to client applications and converts them into block operation requests.
- Block layer: implements the distributed algorithm on tree blocks.
- Key Based Routing layer: routes messages according to the target block key.

Properties

- Host storage receives objects of bounded size (tree blocks).
- Insertions are bounded by tree height on number of blocks accessed.
- Queries use sequential leaf access or ordered tree traversal on internal data for range queries.

Initial Results

- We simulated building the index by inserting locations of a small collection of text documents.
- Smaller block sizes generate smaller variations on host load and thus uniforms storage.
- The largest block size (1k) is roughly equivalent to a direct DHT usage.



Impact of block size on load distribution among hosts.

Final Remarks

- Our algorithm offers a generic index functionality that can be used as a building block for new large-scale P2P applications.
- Queries may use other heuristics and caching to improve scalability.
- Evaluation of network bandwidth usage is subject of ongoing work.
- Considering other data sets beyond text to show range query properties as future work.