Magnolia: a novel DHT architecture for keyword-based search

Ashish Gupta, Manan Sanghi
Peter A. Dinda, Fabian Bustamante
Department of Computer Science,
Northwestern University
{ashish,manan, pdinda, fabianb}@cs.northwestern.edu

Why Magnolia ?

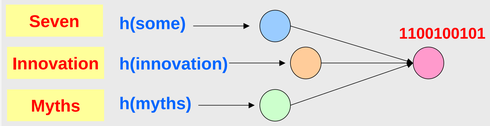
1

DHT based systems à big improvement over unstructured systems
(1) O(log n) routing and lookup
(2) Effective Load balancing

Problem à Keyword search difficult because of hashing based lookup

"Seven Innovation Myths" —h(title)→ **1100100101** ←✗— **"Innovation"**

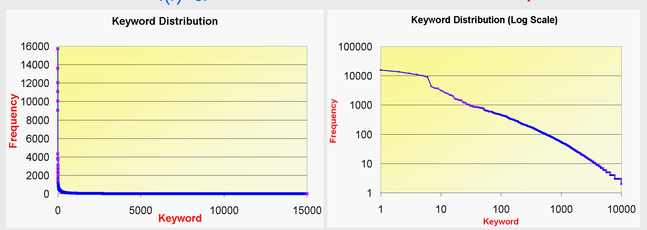
Current approach: Hash each keyword separately and store pointers at h(keyword)

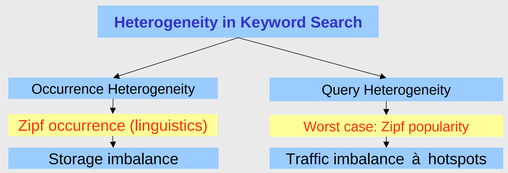


Need to balance out load and provide scalability ,
without sacrificing DHT properties (routing efficiency and state)

f(r)=cr $^{-1.2}$

Data: DBLP Paper Database





1. Node overload with keywords
2. Popular nodes fail : entire keyword set gone
3. Routing and Query hotspots for popular keywords

Storage and Lookup

3

Insert

Keyword h_P à SiblingGroup ID
Locate a sibling node via SIFT
Random Sibling

Lookup

Keyword
O(1)
Replies
Group Broadcast or Multicast

- Incremental Search
- Popularity based ranking
- Boolean Searches
- Multiple Attributes

Analytical Bounds

m' (sibling ID length) = 12, n (number of nodes) = 1 million

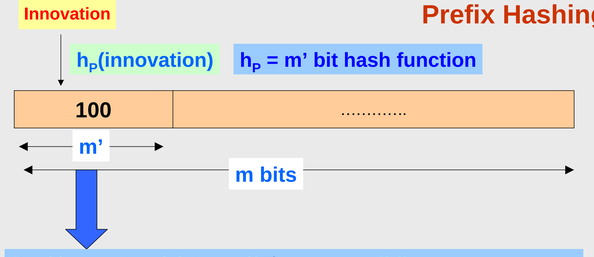
Routing and Lookup = O(1)	Numbers of Nodes visited = $O(m'+n/2^{m'})+r$ ~ 268 nodes for complete query
Total Traffic Generated = $O(m'+n/2^{m'})+r$ r = number of replies Traffic ~ 268 + r units for complete query	Routing state per node = $O(\log n + 2^{m'}+n/2^{m'})$ ~40 Kbytes

Prefix-based Hashing and Sibling Partitioning

2

Prefix Hashing

Innovation
h_P(innovation)
h_P = m' bit hash function



Partitions network into ~ $n/2^{m'}$ separate ***sibling groups***

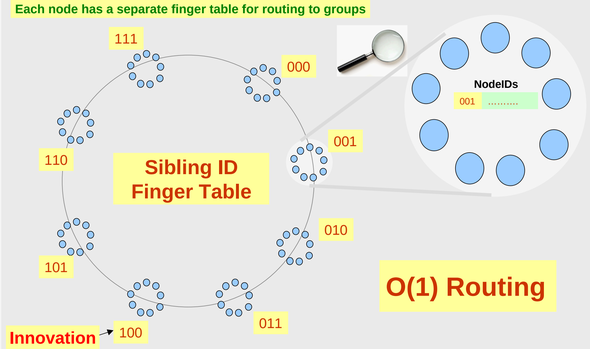
n = nodes, m' à partitioning factor
For m'=12, n= 1 million, ~ 256 nodes will share same prefix
Assumption: ***h*** is uniformly distributed

Balancing

Innovation
Balanced over the sibling group
100
Sibling group ID=100
All siblings in a group share the same prefix

- Very Effective Implicit Balancing
- Greatly simplifies routing in DHT systems
- Group sizes can be tightly controlled
- Bounded and low amount of traffic
- Can return replies in constant time ! < O(log n)
- More resilient

Each node has a separate finger table for routing to groups



Prefix Routing

- Each node maintains a siblingGroup ID finger table (SIFT)
- Even for 1 millions nodes, can store pointers for all siblingGroup IDs
- Each SIFT entry stores pointer to ONE of siblings
- Can quickly route to the correct group in O(1)

O(1) Routing

Internal Routing in a Group

Each sibling has a Group Membership Table (GMT), knowing about all other siblings

100 →

100 00110	194.201.146.1
100 01010	204.123.28.30
100 01100	212.187.13.73
100 10011	66.17.148.141
100 01101	66.13.170.193
100 10101	26.34.109.191
100 00001	210.131.75.80
100 10000	164.71.1.148

Good Balancing Properties

4

Question: What is the variance of load over all the nodes ?

For a Zipf keyword distribution, final load variance can be computed from original distribution variance

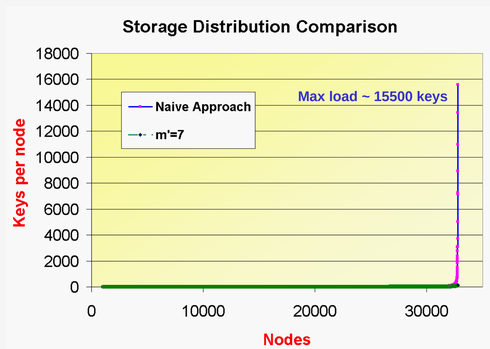
$$V_{final} = \frac{V_{orig} \cdot k \cdot 2^{m'}}{n^2}$$

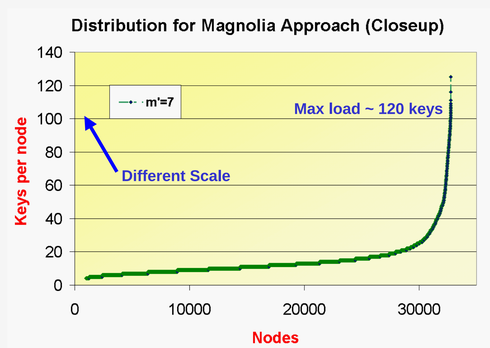
V_{orig} = Keyword occurrence variance in input
V_{final} = Load variance over all nodes
k = number of keywords

Simulation Results

5

DBLP Paper Database: 50,000 paper title insertions
Total keywords = 412,040, Distinct Keywords = 40789





Load Variance over all the nodes drops from 27292 to 131

Predicted Theoretical Variance = 124

Similar Benefits for traffic distribution, max traffic per node, reply traffic.