A Peer-to-Peer Inverted Index Implementation for Word-based Content Search

Nuno Lopes

University of Minho

October 2003

P2P System Characterization

- Scalable up to millions of nodes
- Highly dynamic node membership
- Reduced node uptime: 1 hour on average
- No centralized authority

1st Generation of P2P Systems
File Sharing Oriented

- Napster

  Centralized search with p2p file download

  $\Rightarrow$ Single point-of-failure

- Gnutella

  Broadcast based search

  $\Rightarrow$ Network overloaded

Searching Model

- Local model

  Individual peer search

  Examples: Gnutella, Pedone'02

- Global model

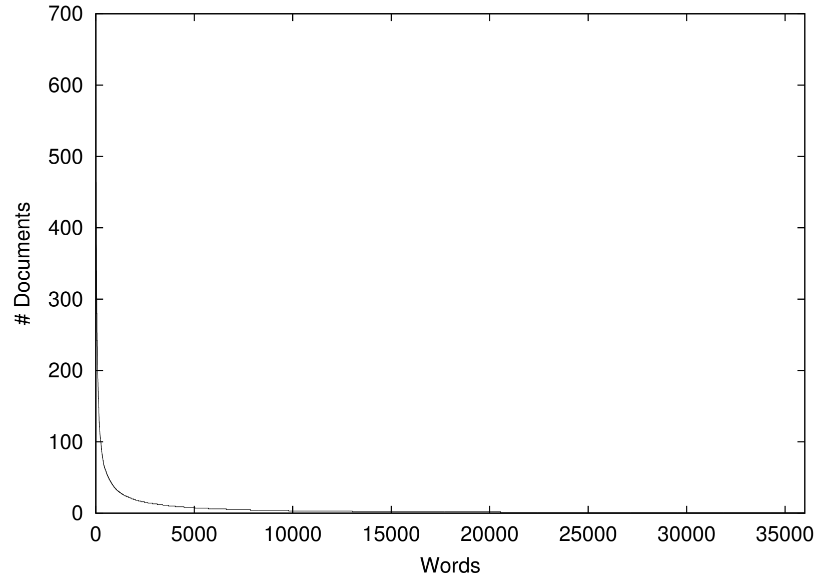
  Information is placed on a global (distributed) shared index

2nd Generation of P2P Systems
Distributed Hash Table (DHT) Based

- Examples: Chord, Pastry, others...
- Simple hash table operations on $(key,value)$ pairs
- Efficient routing: $O(\log N)$ hops for any peer
- Scalable state information: $O(\log N)$ routing entries per peer
- But... *incapable of searching*

Inverted Index Description

- Association $word \mapsto \{document\ location\}_{SET}$
- Document Location Set is highly dynamic
- Follows Zipf distribution



Inverted Index API

- INSERT($word$, $reference$)
- REMOVE($word$, $reference$)
- HAS_REF($word$, $reference$): $bool$
- GET_REF($word$): $reference$
- NEXT_REF($word$, $reference$): $reference$

Inverted Index Implementation

Index is splited in constant size blocks, accessed through 2 layers:

- DHT as base platform for block-oriented storage

  $\Rightarrow$ *Unsuitable as a stand-alone implementation*

- B+ tree for block management

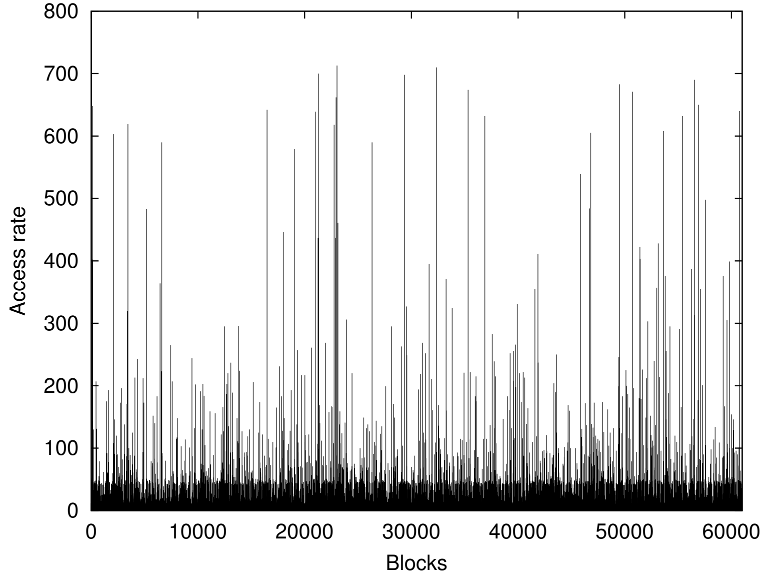
  Responsible for the set implementation to each word

Current Simulation Settings

- Only the B+ tree layer is simulated
- Peers store a single block each
- Messages have an atomic cost
- Single client requests index operations on the system
- Data consists on 1000 small documents with 36499 unique words

Initial Simulation Results

- B+ trees make the storage load uniform across peers
- However... root blocks for popular words have high network load

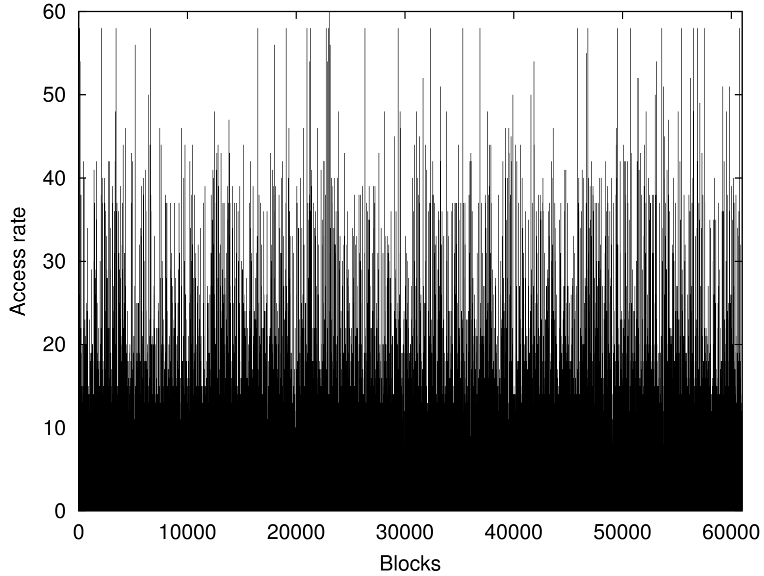


Caching Mechanism

- Clients have high probability of requesting the same blocks for popular words
- Caching of (non-leaf) blocks reduces the number of accesses
- In order to avoid stale copies, leaf blocks are never cached
- Higher level blocks are less probable to become modified and therefore stale

Simulation Results (Using Cache)

- The use of a cache mechanism (LRU) distributes more evenly the network load on peers
- Access rates were reduced by a factor of 10



Open Questions

- Measurement of DHT as stand-alone implementation of inverted index
- Analysis of the block caching mechanism to determine the best cache size for different numbers of peers on the system
- Implementation of multiple blocks to peer association for studying effective peer load
- AND and OR search operators implementation and load measurement