Distributed Indexing

Arvind Krishnamurthy
Fall 2004

Motivation

- Distributed Indexing generalizes in two ways:
  - System stores objects with multiple attributes
  - Objects can be retrieved by using one or many of the attributes
  - Should be able to support complex operations such as:
    - Nearest-neighbor queries
    - Range queries

- Motivating applications:
  - Large number of documents
  - Networking probing information
  - Sensor networks gathering data
Approaches

- Three current solutions:
  - PIER system (no range queries or nearest-neighbor queries)
  - pSearch system
  - Mercury system
  - SkipIndex system

- PIER:
  - Uses a hybrid of “unstructured” and “structured” networks
  - Uses the “unstructured” network without guarantees
  - Uses the “structured” network to guarantee retrieval

Unstructured Networks

- Ad-hoc topology
- Queries are flooded for bounded number of hops
- No guarantees on recall
- E.g. Gnutella and Kazaa

Query: “xyz”
Structured Networks

- Distributed Hash Tables (DHTs)
- Hash table interface: \texttt{put(key,item)}, \texttt{get(key)}
- \(O(\log n)\) hops
- Guarantees on recall

Keyword Search using DHTs

- \textit{Inverted Lists} hashed by keyword (term) in the DHT

Query: \texttt{“T1 AND T2”}
PIER Approach

Flood-based Network
(All items)

DHT
(Partial Index on Rare Items)

Flood Query

Very Few Results?

DHT Query

More Results

Simpler alternative to either optimizing unstructured or structured networks.

Unstructured Networks: Queries with Small Result Sets

Small result set → items have few copies in network
Query Latency

Queries that return few results have poor response times

DHT-based Search

- Advantages:
  - Avoid flooding query in network
  - Guarantee recall (critical for small result sets)

- Disadvantages:
  - Hashing inverted lists into the DHT is costly
  - So is intersecting inverted lists at query time
  - Infeasible for Google-like datasets (IPTPS '03)

- Feasible for querying rare items:
  - Queries with $\leq 10$ results ship 7x fewer inverted list entries compared to the average query
  - Query optimization can reduce communication overhead: intersect rare terms first
Mercury: Supporting Range Queries

- Start with single-attribute data objects
  - Use DHT’s without hash functions
    - Results in load-imbalance
  - Address load-imbalance
    - Results in longer routing distances
  - Modify routing tables
- Generalize to multiple attributes

Using DHTs for Range Queries

- No cryptographic hashing for key → identifier
ARB1  difference between cryptographic hashing and not hashing -- animate... hash(x) = blah, hash(y) = blah_2
Ashwin, 8/25/2004
Using DHTs for Range Queries

- Nodes in popular regions can be overloaded
- Load imbalance!

DHTs with Load Balancing

- Mercury load balancing strategy
  - Re-adjust responsibilities
- Range ownerships are skewed!
DHTs with Load Balancing

- Each routing hop may not reduce node-space by half!
- \( \Rightarrow \) no \( \log(n) \) hop guarantee

Finger pointers get skewed!

Ideal Link Structure
Mercury’s Solution

- Need to establish links based on node-distance

If we had the above information...
- For finger $i$
  - Estimate value $v$ for which $2^i$ th node is responsible

Mercury

- Need to establish links based on node-distance
Histogram Maintenance

- Measure node-density locally
- Gossip about it!

Load Balancing

- Basic idea - leave-join
  - “light” nodes leave
  - Re-join near “heavy” nodes; split the range of the heavier node
"push" samples to $k_2$ nodes
Ashwin, 8/25/2004
Load Balancing

- Basic idea: leave-rejoin
- Steps
  - Find average, check if heavy or light
  - Light nodes perform a leave and rejoin

Multi-attribute Range Queries

- Send data to all rings
- Send query to only ring