Squid: Flexible Information Discovery in Decentralized Distributed Systems

Cristina Schmidt and Manish Parashar
The Applied Software Systems Laboratory
Rutgers, The State University of New Jersey
Outline

- Introduction
- Related Work
- Design
- Evaluation
- Ongoing work
Motivation

- The need for information discovery in large, decentralized, distributed resource sharing environments, in the absence of global knowledge of naming conventions

- Examples:
  - P2P Document Sharing Systems
  - Grid Resource Discovery
  - Web Service Discovery
Overview

- Squid is a Peer-to-Peer (P2P) indexing and information discovery system
- Supports complex queries containing partial keywords, wildcards and range queries
- Guarantees that all existing data elements matching a query will be found with bounded cost in terms of number of messages and nodes involved
Related Work

- **Unstructured (Gnutella-like)**
  - Unstructured overlay network, use flooding

- **Hybrid (Napster)**
  - Unstructured overlay network, use centralized directories for search

- **Data-lookup (CAN, Chord, Pastry, etc)**
  - Structured overlay, Internet-scale DHT

- **Structured keyword search**
  - Structured overlay, extend data-lookup protocols
  - Examples:
    - *Distributed Inverted Indices*
    - *Space Filling Curve*
System components:

- Locality preserving mapping that maps documents to indices – using Space Filling Curves (SFC)
- Overlay network of nodes (Chord)
- Load balancing mechanisms
- A query engine that supports complex queries
Documents have assigned keywords

3-dimensional keyword space for storing computational resources, using the attributes: storage space, base bandwidth and cost

2-dimensional keyword space for a P2P sharing system
Design - Overview

- Basic operations:
  - Store documents in the system
    - Attach keywords to document
    - Index the document using Hilbert Space Filling Curve
    - Store the document at the appropriate node in Chord

```
Document (kw1, kw2, ..., kwD) ➔ Point in a D-dimensional space

Peers (P1, P2, ...Pk, ...) ➔ Point in a 1-dimensional index space
```

HPDC, June 2003
Design - Overview

- Query the system
  - Detect the index regions that are defined by the query
  - Query the appropriate nodes in Chord overlay
  - Get the data

Query (kw1, kw2, …, kwN) → Segment in the 1-dimensional index space

Peers (P1, P2, …Pk, …) → Data

Area in the D-dimensional space

SFC
Hilbert Space-Filling Curve (SFC)

- \( f: \mathbb{N}^d \rightarrow \mathbb{N} \), recursive generation

![Recursive Generation Diagram]

- Properties:
  - Digital causality
  - Locality preserving
  - Clustering
Using SFC to generate the index space

- the d-dimensional keyword space is mapped to a 1-dimensional index space using SFC
The overlay network

- Use Chord as overlay network

Overlay network with 5 nodes and an identifier space from 0 to 64

Each node stores the keys that map to the segment of the curve between itself and the predecessor node.

Cost to look-up data: $O(\log_2 N)$
The Query Engine

- Query: combination of keywords, partial keywords, wildcards, ranges

- Example:
  - (computer, network)
  - (computer, net*)
  - (comp*, *)
  - (256-512MB, *, 10Mbps-*) (memory, cost, base bandwidth)
Query Processing

- Step 1: Translate the query to relevant clusters on the SFC-based index space
  
  Query, e.g. (computer, *)

- Step 2: Query the appropriate nodes in the overlay
  
  Query the nodes 13 and 32
Query optimization

- Not all clusters that are generated for a query exist in the network => optimize!
- SFC generation recursive => clusters generation is recursive => the process of cluster generation can be viewed as a tree
- Optimization: embed the tree into the overlay, and prune nodes during the construction phase
Query optimization – illustration

Solve query: (011, *)
Query optimization – illustration

Embed the leftmost tree path (solid arrows) and the rightmost path (dashed arrows) onto the overlay network topology.
Experimental evaluation

- 1000 to 5400 nodes
- Up to $10^6$ keys (unique keyword combinations)
- Metrics:
  - Number of routing nodes
  - Number of processing nodes
  - Number of data nodes
  - Number of messages
- Query types:
  - **Q1**: (computer, *), (comp*, *, *)
  - **Q2**: (comp*, net*), (computer, network, *)
  - **Q3**: range queries

HPDC, June 2003
2D space – Q1 and Q2 queries

- System size increases from 1000 to 5400 nodes, keys from $2 \times 10^5$ to $10^6$
3D space – Q1 and Q2 queries
3D space – range queries
Load balancing

- Load balancing at node join:
  - generate more than one ID for the new node, send join requests in the network and join with the ID that places the node in the most crowded part of the network

- Load balancing at runtime:
  - run a local load balancing algorithm between neighbors (from time to time), and redistribute the load
  - use virtual nodes that can migrate to less loaded physical nodes
Load balancing

The distribution of the keys in the index space. The index space was partitioned into 5000 intervals. The Y-axis represents the number of keys per interval.

The distribution of the keys when using only the load balancing at node join technique.

The distribution of the keys when using both the load balancing at node join technique, and the local load balancing.
Ongoing and future work

- Develop new methods to further prune the clusters that do not exist in the network
- Ranking
- New overlay topology
- Improve the system’s availability (replication and caching)
Questions