Search in Social Networks with Access Control

Truls A. Bjørklund, Michaela Götz, Johannes Gehrke
Norwegian University of Science and Technology, Cornell University
Content Search in Social Networks

Where's the Chopin video?

 Keys 2010, Michaela Götz
Search System Desiderata

- Want a system that:
  - Given a query (set of keywords) returns top-k most recent posts containing these keywords
  - adheres to the privacy settings of users (can only retrieve friends’ posts)
  - makes new posts immediately searchable
  - answers queries quickly
  - does not consume too much space
Informal Problem Definition

- Given a social network
  - Nodes = Users
  - Edges indicate friendship (selflinks omitted)
- Given posts written by users (the authors)

- Answer conjunctive queries
  - Result of a query
    1. Top-k most recent posts that contain all keywords of the query
    2. Are authored by friends
- Queries with access control
Design Space

- Two axes of enforcing access control:
  - Index axis:
    - A group index contains the posts of a subset of users
    - An index design is a set of group indexes
  - Access axis:
    - A group author list is a sorted list of pairs <post-ID, author-ID> for a subset of users
    - An access design is a set of group author lists
  - Intuition: Query processing
We distinguish designs based on:

<table>
<thead>
<tr>
<th></th>
<th>Index Design</th>
<th>Access Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cardinality</strong></td>
<td># of indexes</td>
<td># of author lists</td>
</tr>
<tr>
<td><strong>Redundancy</strong></td>
<td>avg # of indexes a user is member of</td>
<td>avg # of lists a user is member of</td>
</tr>
</tbody>
</table>
Examples – Index Designs

Global Index
- I1: No redundancy
  Lowest cardinality

Friends Indexes:
- I1:
- I2:  
- I3:  
- I4:  
- I5:  
  High redundancy
  High cardinality
Examples – Access Designs

Global List
- L1: No redundancy
  Lowest cardinality

Friends List:
- L1:
- L2:
- L3:
- L4:
- L5:
  High redundancy
  High cardinality
Terminology

- **Covers:**
  - A set of group indexes *covers* a set of users if each user’s posts are contained in at least one group index.
    - **Exact covers:** no posts of other users.
  - A set of group author lists *covers* a set of users if each user’s posts are contained at least one group author list.
    - **Exact covers:** no posts of other users.
Query Processing with Access Control

- Given query $t_1, \ldots, t_m$ by user $u$
  1. Select indexes covering $u$’s friends.
  2. Select author lists covering $u$’s friends.
  3. Within each selected index:
     a) Intersect posting lists for $t_1, \ldots, t_m$ with the union of the selected author lists.
     b) For each result check whether $u$ is friends with author.
  4. Union the results of the indexes

\[
\bigcup_{\text{indexes}} \sigma_{\text{friends}} \left( \left( \bigcap \text{posting lists for } t_1, \ldots, t_m \right) \cap \left( \bigcup \text{author lists} \right) \right)
\]

- Processing Optimizations:
  - Group indexes are exact cover $\Rightarrow$ no further filtering (skip 2., 3.)
  - Group author lists are exact cover $\Rightarrow$ no friendship check (skip 3.b)
Examples – Index + Access Designs

Friends Indexes / No Group Author lists:

- I1:
- I2:
- I3:
- I4:
- I5:

\[ \bigcup \text{indexes} \sigma_{\text{friends}} \left( \bigcap \text{posting lists for } t_1, \ldots, t_m \right) \cap \left( \bigcup \text{author lists} \right) \]

High redundancy
High cardinality
Optimization: No author lists
Examples - Index + Access Designs

User Indexes / No Group Author lists:

- I1:
- I2:
- I3:
- I4:
- I5:

\[ \bigcup_{\text{indexes}} \sigma_{\text{friends}} \left( \bigcap \text{posting lists for } t_1, \ldots, t_m \right) \cap \left( \bigcup_{\text{author lists}} \right) \]

No redundancy
High cardinality
Optimization: No friends lookup

KEYS 2010, Michaela Götz
Examples – Index + Access Designs

Global index / Global list

Index:

\[ I_1: \text{No redundancy} \quad \text{Lowest cardinality} \]

Author lists:

\[ L_1: \text{No redundancy} \quad \text{Lowest cardinality} \]

\[ \bigcup_{\text{indexes}} \sigma_{\text{friends}} \left( \bigcap \text{posting lists for } t_1, \ldots, t_m \right) \cap \left( \bigcup \text{author lists} \right) \]
Examples – Index + Access Designs

**Global index / User list**

**Index:**

- I1:

**Author lists:**

- L1:
- L2:
- L3:
- L4:
- L5:

\[ \bigcup_{\text{indexes}} \sigma_{\text{friends}} \left( \bigcap_{t_i, \ldots, t_m} \right) \cap \left( \bigcup \text{author lists} \right) \]

No redundancy
Lowest cardinality

No redundancy
High cardinality
Optimization:
No friends look-up

KEYS 2010, Michaela Götz
Examples – Index + Access Designs

Global index / Friends list

Index:
- \( I_1 \)

Author lists:
- \( L_1 \)
- \( L_2 \)
- \( L_3 \)
- \( L_4 \)
- \( L_5 \)

\[
\bigcup_{\text{indexes}} \sigma_{\text{friends}} \left( \bigcap_{t_1, \ldots, t_m} \text{posting lists for } t_i \right) \cap \left( \bigcup \text{author lists} \right)
\]
Implementation - Overview

- Main memory system in Java

  Updates:
  - Small updatable index to add new posts
  - hierarchy of indexes based on geometric partitioning with compression

- Operators over lists:
  - Operators: Union, Intersection, Filter
  - Methods: Next(), SkipTo(value v)
Experiments

- Comparison of the performance
  - Across index designs
  - Across access designs
  - Across different social networks

- Performance measures:
  - Time to answer query
  - Time to add post
  - Space consumption
Experiments

- **Data:**
  - **Network:**
    - real twitter network: 417,000 users
    - synthetic networks (Barabasi’s attachment model) varying size and degree
  - **Posts obtained from twitter**
  - **Queries**
    - generated through a random process
    - Run 100,000 queries returning top-100 posts

- **Environment:** 3.2GHz, 16GB RAM, Red Hat Enterprise 5.3
Scalability

Varying the number of documents

fix 1,000 users, 20 friends per user

Update Cost (s)                  Search cost (s)                             Space (MB)

Global / Global                  Global / Global                                Global / Global
Friends / None                    Friends / None                               Friends / None
User / None                       User / None                                 User / None
Global / None -> no access control

Number of posts \times 10^5

Processing time (s)

Processing time (s)

Processing time (s)
Index Design Performance under Varying Networks

Update Cost (s)
Varying the number of users

Search cost (s)
Varying the number of users

Space (MB)
Varying the number of users

Update Cost (s)
Varying the number of friends per user

Search cost (s)
Varying the number of friends per user

Space (MB)
Varying the number of friends per user

Varying the number of friends per user
fix 10,000 users, 1mm posts

Varying the number of users
fix 100 friends per user, 1mm posts

Varying the number of users

Varying the number of friends per user

Index Design Performance under Varying Networks

Update Cost (s)                  Search cost (s)                             Space (MB)

Update Cost (s)                 Search cost (s)                             Space (MB)

Varying the number of users
fix 100 friends per user, 1mm posts

Varying the number of friends per user
fix 10,000 users, 1mm posts
Access Design Performance under Varying Networks

**Varying the number of users**

- **Update Cost (s)**
  - Processing time
  - $10^4$ to $10^6$

- **Search cost (s)**
  - $10^4$ to $10^6$

- **Space (MB)**
  - $10^4$ to $10^6$

**Varying the number of friends per user**

- **Update Cost (s)**
  - Processing time
  - 10 to 200

- **Search cost (s)**
  - 10 to 200

- **Space (MB)**
  - 10 to 200

**Fix 100 friends per user, 2.5mm posts**

**Fix 100,000 users, 2.5mm posts**

- Global / Global
- Global / Friends
- Global / User
- Global / None
Experiment on Real Twitter Network

Access designs with global index on real network

**Update Cost (s)  Search cost (s)  Space (MB)**

![Bar charts showing different access designs: Global / Global, Global / User, Global / Friends, Global / None.](chart.png)

**Update Cost (s) vs. Search cost (s) vs. Space (MB)**

---

KEYS 2010, Michaela Götz
Conclusions

- **Two axis design space for access control in search:**
  - Index Axis
  - Access Axis

- **Experiments** with five designs:
  - Access designs reveal tradeoffs between index size, update and search performance
    - Global Index / Friends lists
      - fast searches (independent on network)
      - slow updates (dependent on network)
    - Global Index / User lists or Global Index / Global list
      - slow searches (dependent on network)
      - fast updates (independent on network)

- Similar tradeoffs for index designs

- **Recommendation:** Choose between user indexes and the global index with user or friends lists based on workload and network
Future Work

- Explore design space
  - Identify best design for a particular workload and network

- Dynamic design
  - Adapt to changes in the workload
  - Adapt to changes in network

- Distribute system
  - Extend to more advanced ranking functions
  - Include network structure and interactions as features
  - Do not leak private information through ranking
Thank You!

Questions?

goetz@cs.cornell.edu