Efficient Graph Models for Retrieving the Top-k News Feeds from Ego Networks

René Pickhardt, Thomas Gottron, Jonas Kunze, Ansgar Scherp Steffen Staab

How to retrieve more than 10'000 temporal ordered news feeds per second in social networks with millions of users like Facebook and Twitter by using graph data bases (like neo4j) and Graphity
Joint collaboration with

Thomas Gottron

Jonas Kunze (from metalcon.de)

Ansgar Scherp

Steffen Staab
• Introduction to the newsfeed problem

• Why relational Data bases won't do the job

• The construction and idea of STOU

• The construction and idea of graphity

• Example 1: retrieval of news feeds (top-k n-way merge)

• Example 2: Creating new Content Items

• Evaluation on Wikipedia data set.
A "typical" social network graph
Retrieving Node A's news stream
Retrieving Node A's news stream
Some Challenges

Social networks like Twitter and Facebook have several thousand requested news feeds per second.

News feeds change fast: Several hundred newly created content items per second. (600 tweets/sec in 2010)

News feeds are different for every user.

**Realtime** (retrieval should be as low as micro seconds)

Friendship graph changes over time.

Overall: This is a very **dynamic problem** with a lot of **chaotic & unpredictable behaviour**.
• Introduction to the newsfeed problem

• Why relational Data bases won't do the job

• The construction and idea of STOU

• The construction and idea of graphity

• Example 1: retrieval of news feeds (top-k n-way merge)

• Example 2: Creating new Content Items

• Evaluation on Wikipedia data set.
First we have some Users in a social Network
They follow other users

<table>
<thead>
<tr>
<th>ID</th>
<th>from</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

User: a, b, c, d, e

Followers: b, c, d, e

Diagram: Graph showing user relationships with arrows indicating follow directions.
Everyone produces status updates and content.

<table>
<thead>
<tr>
<th>User</th>
<th>Follower</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>from</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
</tr>
<tr>
<td>e</td>
<td>b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ContentItems</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
</tbody>
</table>
Our Query joins over huge Follower Matrix

```
SELECT ci.User, ci.time, ci.Content
FROM ContentItems ci
JOIN Follower f on ci.User = f.to
JOIN User u on u.ID = f.from
WHERE u.ID like "a" ORDER BY ci.time DESC
```
• Introduction to the newsfeed problem
• Why relational Data bases won't do the job
• The construction and idea of STOU
• The construction and idea of graphity
• Example 1: retrieval of news feeds (top-k n-way merge)
• Example 2: Creating new Content Items
• Evaluation on Wikipedia data set.
From the standard social network graph we move to (temporal) ordered lists for rectangular nodes
Introducing linked lists for status updates!

What are the pros and cons of this change?
Pros of this approach

- dynamic retrieval possible (friendship graph may change)
  - very flexible data structure
- inserts and removes are very fast (all operations are O(1))
• unclear which edge to traverse first!
  • ==> entire ego network must be sorted
• Size \( d \) of an ego network is usually much bigger than the number of retrieved items \( k \).
  • ==> Sorting seems to much effort
• Introduction to the newsfeed problem

• Why relational Data bases won't do the job

• The construction and idea of STOU

• The construction and idea of graphity

• Example 1: retrieval of news feeds (top-k n-way merge)

• Example 2: Creating new Content Items

• Evaluation on Wikipedia data set.
The key concept: going from star topology to lists
Graphity index for the node "a"

Graphity rules:

- for every node (a & b) that follows others we create a linked list
- this linked list contains all the nodes that are being followed by this node.
- The followed nodes are sorted by the timestamp of their most recent content item
second graphity index for node "b"
Index

• Introduction to the newsfeed problem
• Why relational Data bases won't do job
• The construction and idea of STOU
• The construction and idea of graphity

Example 1: retrieval of news feeds (top-k n-way merge)

Example 2: Creating new Content Items

Evaluation on Wikipedia data set.
Rearranging the graph...
top-k n-way Merge for retrieval in $O(k \log(k))$

Stream:

Priority Queue:
19

push(19) to PQ

Pointer tmp
top-k n-way Merge for retrieval in $O(k \log(k))$

Stream: (19,d)

Priority Queue:
14
3

push(14) to PQ
push(3) to PQ

Pointer tmp
top-k n-way Merge for retrieval in $O(k \log(k))$

Stream: (19,d) (14,b)

Priority Queue:
12
11
3

push(12) to PQ
push(11) to PQ
**top-k n-way Merge for retrieval in O(k log(k))**

Stream: (19,d) ; (14,b) ; (12,c)

Priority Queue:

- 11
- 5
- 3

Pointer tmp

push(5) to PQ
top-k n-way Merge for retrieval in $O(k \log(k))$

Stream: (19, d) ; (14, b) ; (12, c) ; ...

Priority Queue:
- 11
- 5
- 3

retrieve 11 now!
• Introduction to the newsfeed problem
• Why relational Data bases won't do the job
• The construction and idea of STOU
• The construction and idea of graphity
• Example 1: retrieval of news feeds (top-k n-way merge)

   • Example 2: Creating new Content Items

• Evaluation on Wikipedia data set.
Creating new Content items

b creates a new content item
Creating new Content items

b created 20

- update linked list of b's content items
Creating new Content items

b created 20

- update linked list of b's content items
- now look in which ego networks b is member of. (our case just a)
Creating new Content items

b created 20

- update linked list of b's content items
- now look in which ego networks b is member of. (our case just a)
- interlink b's predecessor and successor
Creating new Content items

b created 20

- update linked list of b's content items
- now look in which ego networks b is member of. (our case just a)
- interlink b's predecessor and successor
- use the follow edge from a to b and the first ego:a to insert b in the beginning of ego:a
• Introduction to the newsfeed problem
• Why relational Data bases won't do the job
• The construction and idea of STOU
• The construction and idea of graphity
• Example 1: retrieval of news feeds (top-k n-way merge)
• Example 2: Creating new Content Items
• Evaluation on Wikipedia data set.
Wikipedia as a social network

• Every article ==> User

• Every link in an article ==> Follow relationship

• Every Revision of an article ==> Status update of a user

• Remark: if in a new revision the outlinks of the wikipedia article change
  • We don't take this as a status update
  • we interpret this revision as a change to the friendship graph
demonstrating independence of node degree

\[ \alpha = (37 \pm 2) \text{kHz} \]
demonstrating linear dependence of $k$

\[ \alpha_1 = (181 \pm 6) kHz \]
\[ \alpha_2 = (186 \pm 22) kHz \]
\[ \frac{\alpha_1}{k} \]
\[ \frac{\alpha_2}{k \cdot \log(k)} \]
Index maintaining - inserting new content items

![Bar graph showing created content nodes per second vs. wiki dump year (2004-2008). The graph compares two methods: GRAPHITY (red) and STOU (blue). Graph shows a decrease in created content nodes per second from 2004 to 2008.]
updating graphity for new friendship relations
updating graphity if friendships break
time to build the index
Conclusion

• We built two graph indices for top-k news feed retrieval

• STOU is fastest in writing operations (with moderate reading speed)

• Graphity is faster in retrieving operations

• Empirical study on a graphs with up to 2 mio. Users, 32 mio. follow relationships and 50 mio. content items shows that graphity even performs better than the theoretical runtime

• Especially for graphity we saw
  • retrieval of social news feeds of k items in $O(k \log(k))$
  • Almost as good as redundant content lists
  • But no redundancy in content data
So which one to take?

42
Thank you for your attention

More information + Slides on: http://www.rene-pickhardt.de/graphity

Thanks to
• Mattias Persson and Peter Neubauer from neotechnology.com
• the neo4j community on the neo4j mailinglist for helpful advices
• Knut Schumach for coming up with the name GRAPHITY
• Matthias Thimm & Leon Kastler for helpful discussions

This project is founded by the EU Projects Social Sensor and ROBUST.

Sorce code & data sets on: http://www.rene-pickhardt.de/graphity-source-code/
Backup slides

• Backup slides
Rearranging the graph...
Updates need to be done in the following situations

- **new created content item** \( (O(d)) \)
  - index of every follower needs to be updated

- **new created follow relation** \( (O(d)) \)
  - index of follower needs to be updated

- **friendship relation breaks** \( (O(d)) \)
  - index of the former follower needs to be updated

- **most recent content item of a user is deleted** \( (O(d^2)) \)
  - index of every follower needs to be updated
Future work

• Generalize / built theory on top-k joins

• Distributed system

• Partially do graphity index
  • (ever update only yields updating a constant number of graphity indices)

• Tie strength (filtering / ranking)
Retrieving streams on all data sets

[Bar chart showing retrieved streams per second for different years (2004 to 2011) across different data sets (GRAPHITY, STOU, ST).]
Simulating a social network

Simulation rates with Graphity on Wiki dumps

Simulated transactions per second

Wiki dump year

2004  2005  2006  2007  2008

Add:  
Removes:  
Updates:  

Summary - We created a graph model with:

- fast retrieval of social news feeds of k items in $O(k \log(k))$
- dynamic retrieval method
- no redundancy in content data
- Creating new Status Updates yields updating of d graphity indices of following nodes
- Each Graphity index update is $O(1)$

We also conducted an evaluation of a graph with:

- ~ 2 mio. users
- ~32 mio. follow relations
- ~50 mio. Status updates

giving empirical proof of our theoretical findings.