

# Toward Personalized Peer-to-Peer Top-k Processing

Xiao BAI

Marin BERTIER

Rachid GUERRAOUI

Anne-Marie KERMARREC

31 March, 2009



# Outline

- Background and Motivation
- Personalized Peer-to-Peer Top-k Processing
- Evaluation
- Conclusion

# Collaborative tagging and Top-k

- Collaborative tagging systems
  - $U(\text{sers}) * I(\text{tems}) * T(\text{ags})$
  - $\text{Tagged}(u, i, t)$ : User  $u$  annotates item  $i$  with tag  $t$
- Top-k Processing
  - Inverted list per tag 

Item $i$	$\text{Score}_{t_j}(i)$
----------	-------------------------
  - Query  $q = \{t_1, \dots, t_n\}$
  - $\text{Score}(i) = f(\text{Score}_{t_1}(i), \dots, \text{Score}_{t_n}(i))$
  - $k$  items with highest scores as results

# Motivations

- Why collaborative tagging?
  - Flexibility of folksonomy
  - Searching photos, videos, ...
- Why personalization?
  - Variety of user preferences

# How to personalize?

- Network-aware search
  - Interest-based personal network
    - $Network(u) = \{v_i \mid link(u, v_i)\}$
    - $link(u, v)$  iff  $Strength_{link(u, v)} = |\{i \mid Tagged(u, i, t) \& Tagged(v, i, t)\}| > threshold$
  - Personalized score
    - $Score_{t_j}(i, u) = |Network(u) \cap \{v \mid Tagged(v, i, t_j)\}|$

# Why peer-to-peer?

- Exact

- Inverted list per (user, tag)
- Space intensive

- Global Upper-Bound

- Inverted list per tag
- Time consuming:  $Score_{t_j}(i)$  at query time

Item $i$	$UB(i, t_j)$	$users(i)$
----------	--------------	------------

- User Clustering

- Inverted list per (cluster, tag)
- Trade-off between Exact and Global Upper-Bound

# Scalability & Efficiency calls for Decentralization

# Peer-to-Peer Solution

- Goals
  - Top-k results as centralized network-aware search
  - Time consumption as Exact
  - Limited storage per user

# Personal network discovery

- Two-layer gossip protocol
  - Top layer: Personal network
    - Measure the similarities between users
  - Bottom layer: Random Peer Sampling
    - Keep the overlay connected
    - Feed top-layer with new related peers
- Gossip framework
  - Peer selection
  - Data exchange
  - Data processing

# Inverted lists & Query processing

- Inverted list per (user, tag)
- Inverted lists built and stored by concerned user
- $Q = (u, t_1, \dots, t_n)$
- Query processed locally with NRA

# NRA (No Random Access)

- $Score(i) = \sum_{t_j \in Q} Score_{t_j}(i)$
- Inverted lists are scanned sequentially in parallel
- Information for each seen item:
  - Score lower-bound
  - Score upper-bound
- Seen items are sorted on score lower-bound
- Termination condition:  $lower-bound(k\text{-th seen item}) \geq \max\{ upper-bound(i) \mid i \in \{Seen\ items\} / \{top-k\} \}$

# Personal network optimization

- At most  $n$  users per personal network
- Derived from all users meeting the threshold
- Strategies
  - Random
  - Biased Random:  $p_{link(u, v)} \propto Strength_{link(u, v)}$
  - Nearest: highest  $Strength_{link(u, v)}$
  - Nearest With Enhanced Link Strength
    - $Strength_{link(u, v)} = | \{ (i, t_j) \mid Tagged(u, i, t_j) \ \& \ Tagged(v, i, t_j) \} |$

# Evaluation

- Implementation
  - PeerSim
- Data
  - Real trace from del.icio.us
  - 10,000 users
  - 101,144 items
  - 31,899 tags
  - 9,536,635 tagging actions

# Evaluation metrics

- Convergence speed

- $$\text{Speed} = \frac{1}{|U|} \sum_{u \in U} \frac{|Current\ Network\{u\}|}{|Network\{u\}\ of\ Centralized\ Setting|}$$

- Top-k quality: Recall

- $$R_k = \frac{\text{Number of Retrieved Relevant Items}}{\text{Total Number of Relevant Items}}$$

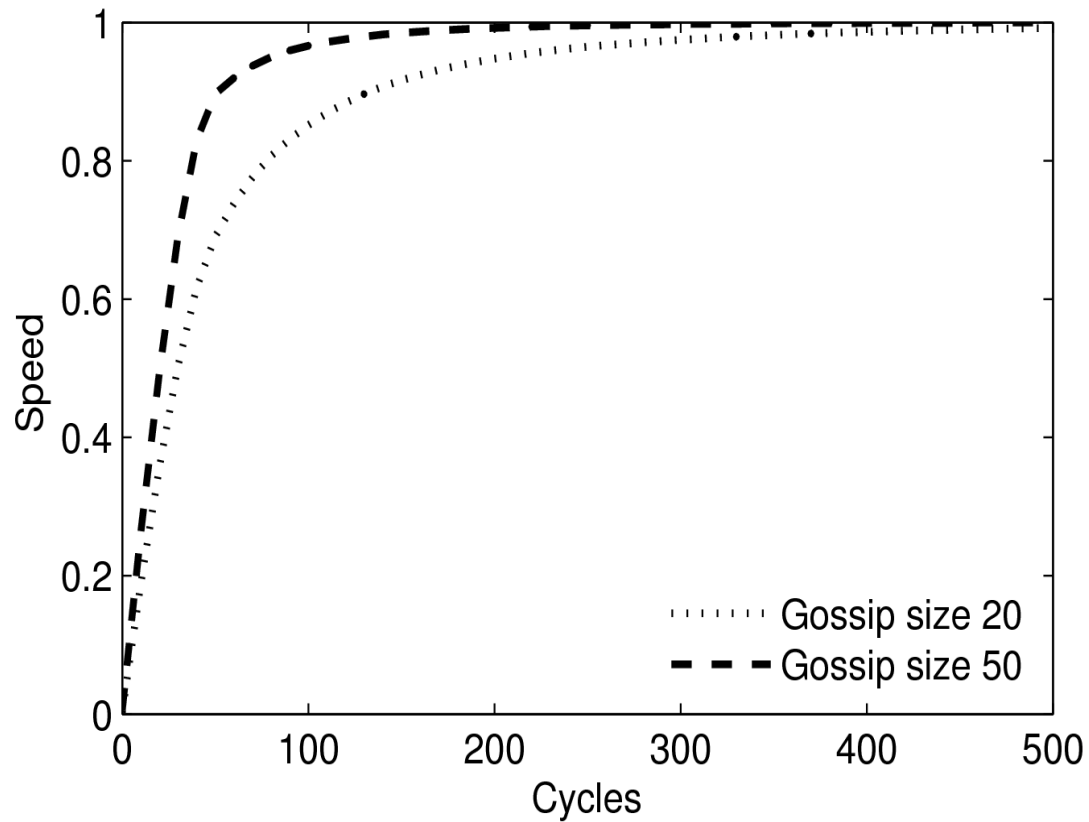
- Storage space per user

- Length of inverted lists
  - Length of neighbors' profiles

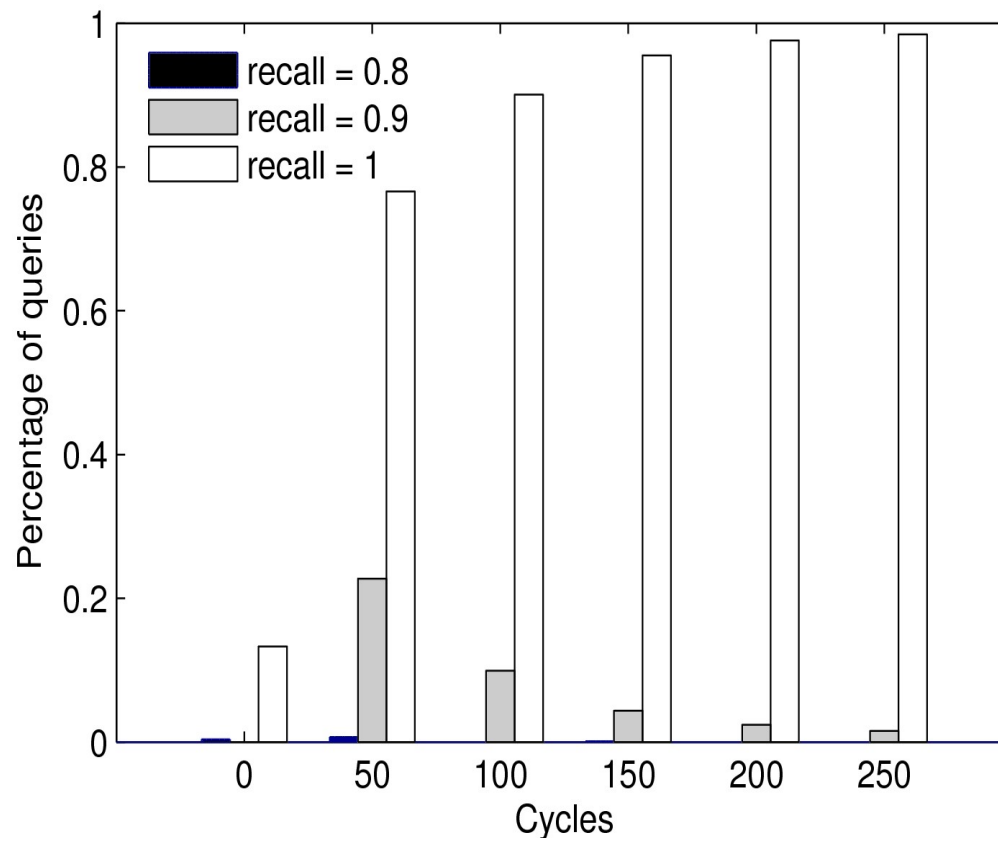
- Processing time

- Number of sequential accesses

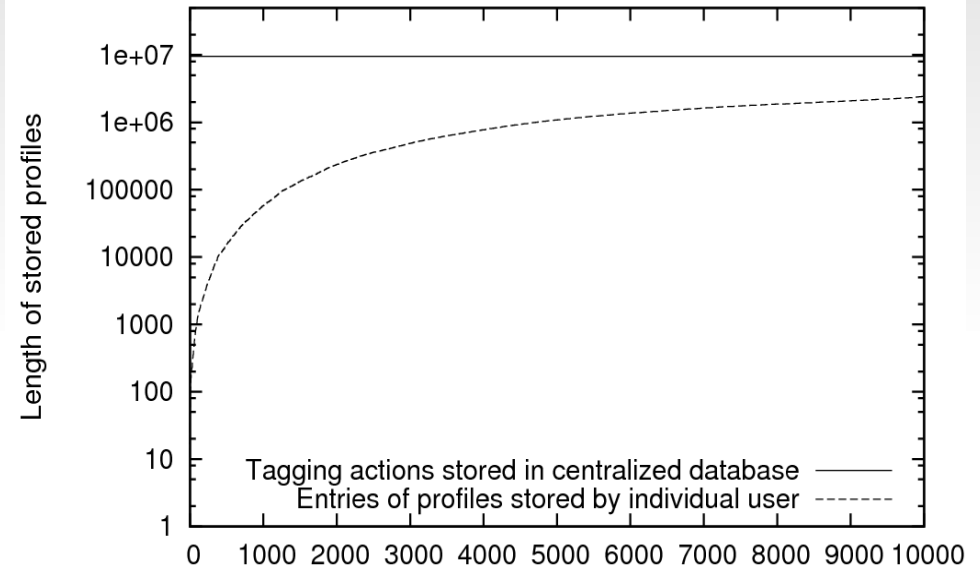
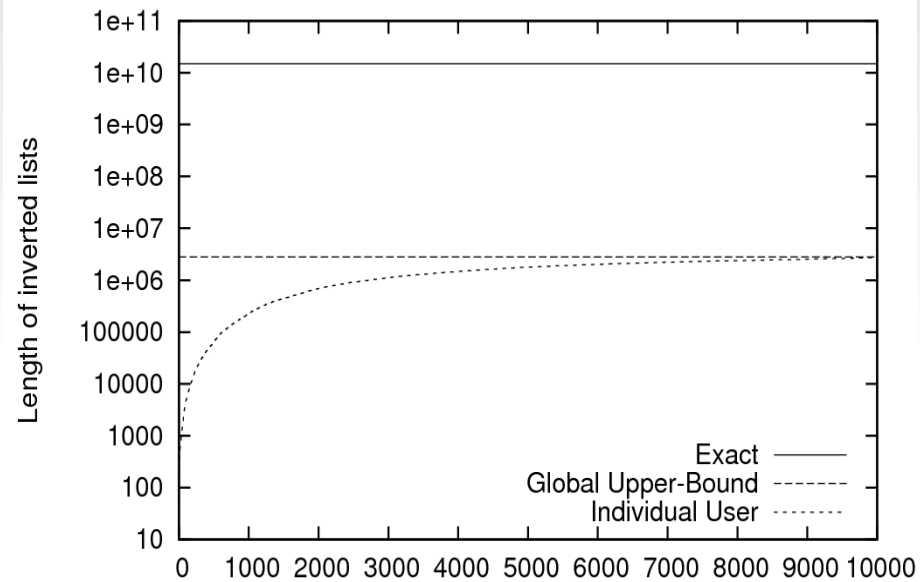
# Convergence speed



# Recall



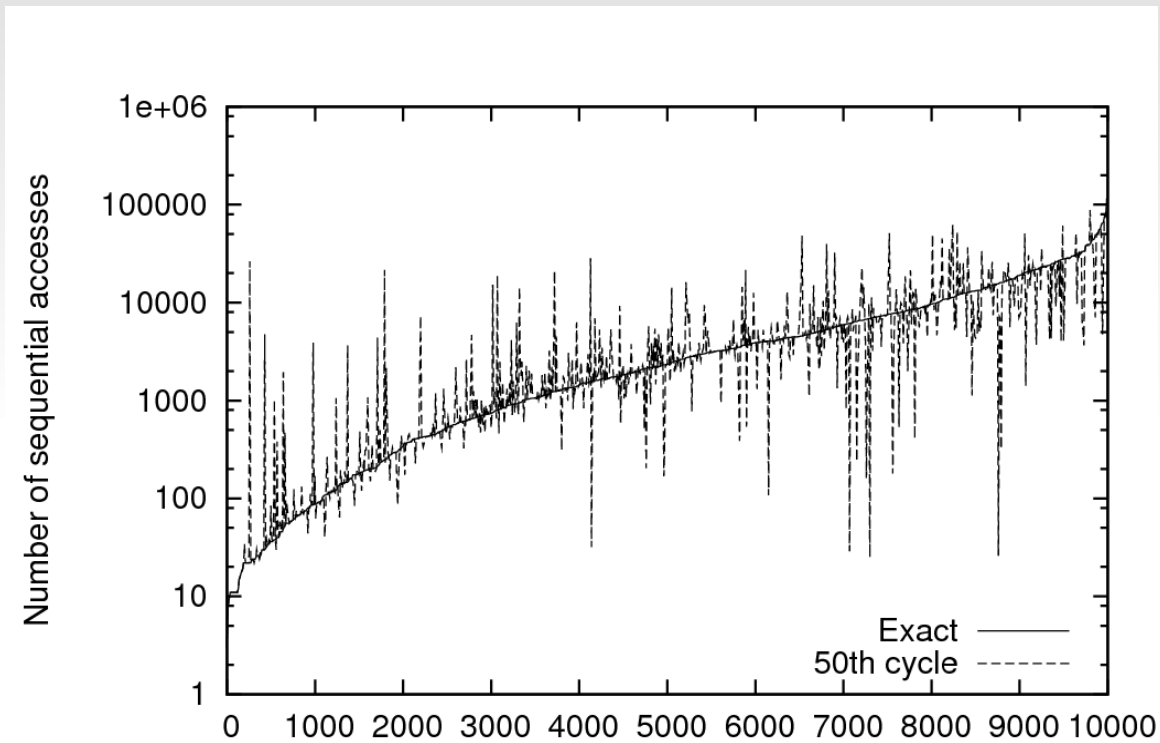
# Storage space



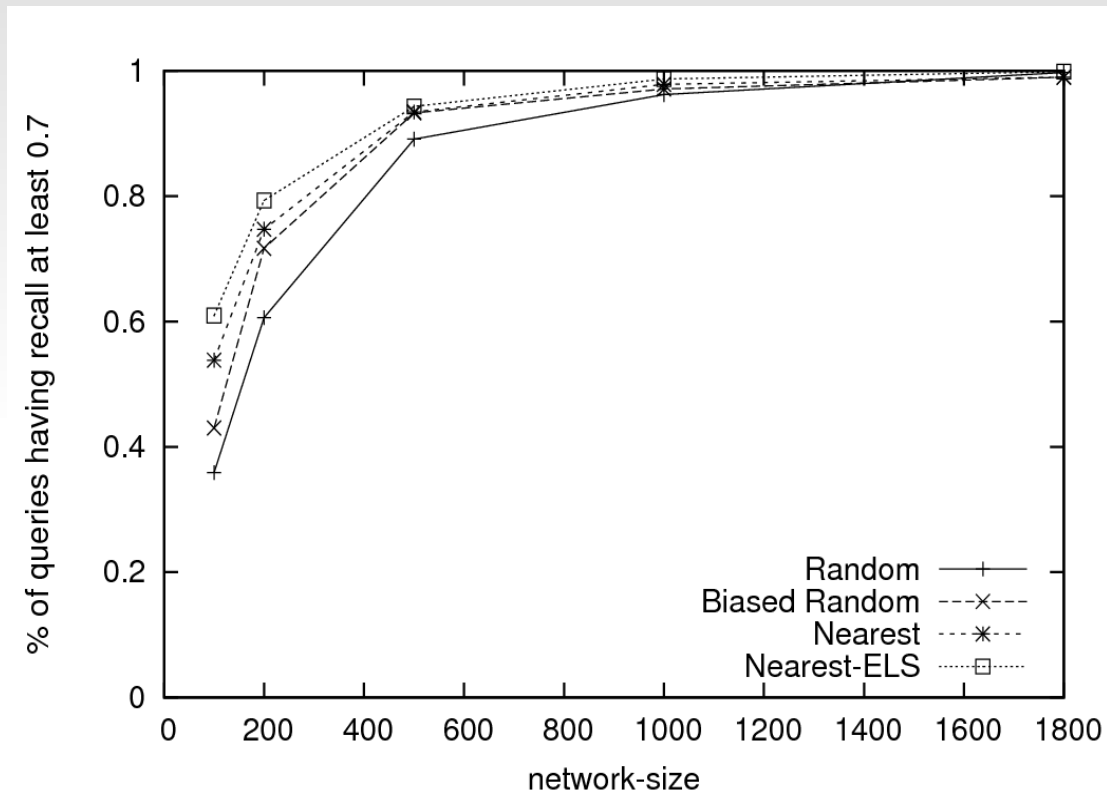
- Inverted lists

- Profiles

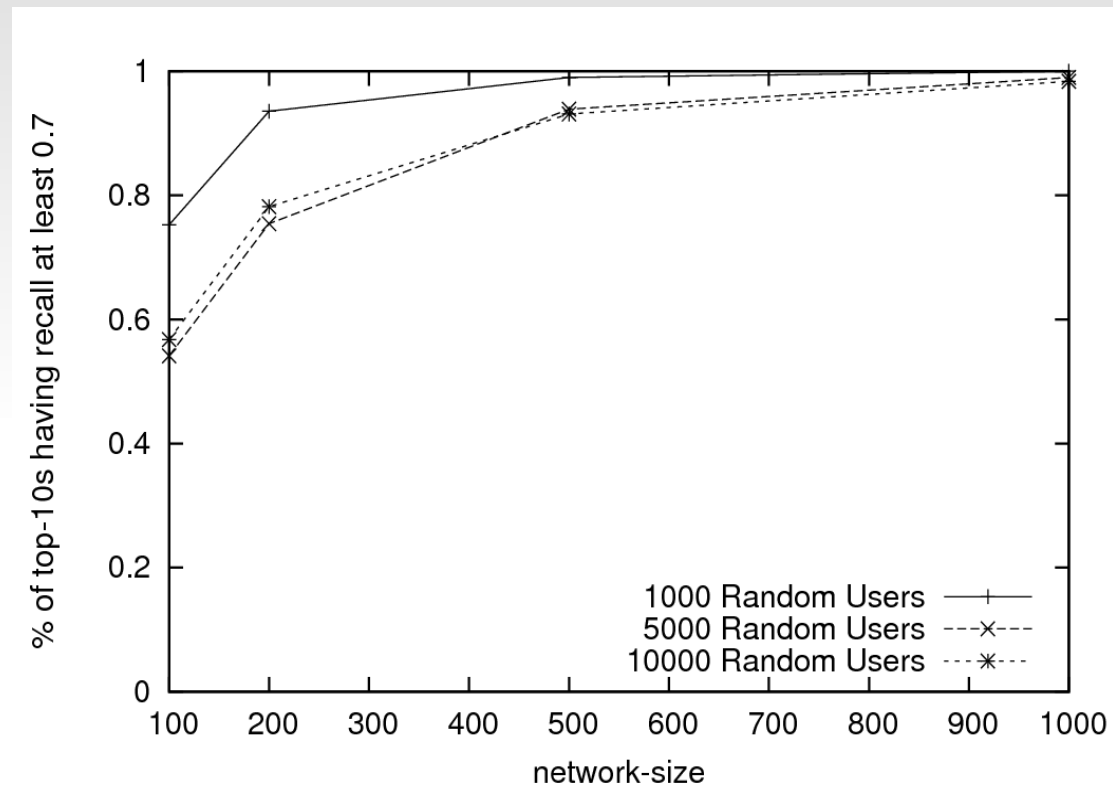
# Processing time



# Comparison of optimization



# Scalability



# Conclusion

- Decentralization is the right way to provide scalable personalized top-k processing
- Future work
  - Churn
  - Privacy

***Thank you!***