# Fault-tolerant TopK algorithms

Lewis Tseng
Boston College

Ruifan Yang
Cornell University

Zheng Zhou
Boston College

- In elections:

  Which are the k globally most popular candidates?

- In peer-to-peer file-sharing networks:

  which are most popular downloads?

# Problem Statement

A top-k Query :

Across all nodes in the system,

return k objects with highest global attribute ( in this case Ni=∑ j Aij ).

# 2-D array

n items, m peers

Each item i has a local attribute

in each peer j, Aij

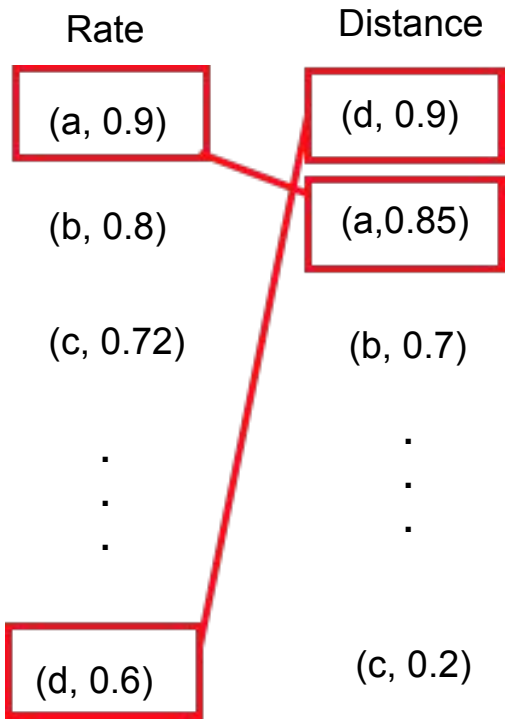
# Threshold Algorithm : Book a hotel

**Step 1:**

Parallel sorted access to each list/peer

For each object seen:

- Get all grades by random access
- Determine Min(A1,A2)
- Keep the current highest 2 in buffer




| ID | $A_1$ | $A_2$ | Min($A_1$,$A_2$) |
|---|---|---|---|
| a | 0.9 | 0.85 | 0.85 |
| d | 0.6 | 0.9 | 0.6 |
| | | | |
| | | | |

# Threshold Algorithm Cont.

**Step 2:**

Determine threshold value T based on objects currently seen under sorted access

T = min(L1, L2)

If 2 objects with overall grade ≥ threshold value
Stop

else
go to next entry position in sorted list
& repeat step 1

| Rate | Distance |
|---|---|
| (a, 0.9) | (d, 0.9) |
| (b, 0.8) | (a,0.85) |
| (c, 0.72) | (b, 0.7) |
| . | . |
| . | . |
| . | . |
| (d, 0.6) | (c, 0.2) |

T = min(0.9, 0.9) = 0.9

| ID | $A_1$ | $A_2$ | Min($A_1$,$A_2$) |
|---|---|---|---|
| a | 0.9 | 0.85 | 0.85 |
| d | 0.6 | 0.9 | 0.6 |
| | | | |
| | | | |

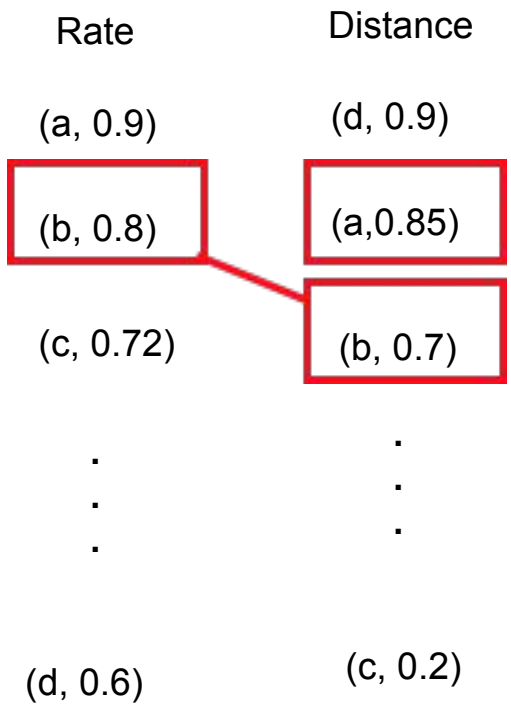
# Threshold Algorithm Cont.

**Step 1 Again**

Parallel sorted access to each list/peer

For each object seen:

- Get all grades by random access
- Determine Min(A1,A2)
- Keep the current highest 2 in buffer




| ID | $A_1$ | $A_2$ | Min($A_1$,$A_2$) |
|---|---|---|---|
| a | 0.9 | 0.85 | 0.85 |
| d | 0.6 | 0.9 | 0.6 |
| b | 0.8 | 0.7 | 0.7 |
| | | | |

# Threshold Algorithm Cont.

**Step 2 Again**

Determine threshold value T based on objects currently seen under sorted access

T = min(L1, L2)

If 2 objects with overall grade ≥ threshold value
Stop

else
go to next entry position in sorted list
& repeat step 1

| Rate | Distance |
|---|---|
| (a, 0.9) | (d, 0.9) |
| (b, 0.8) | (a,0.85) |
| (c, 0.72) | (b, 0.7) |
| . | . |
| . | . |
| . | . |
| (d, 0.6) | (c, 0.2) |

T = min(0.8, 0.85) = 0.8

| ID | $A_1$ | $A_2$ | Min($A_1$,$A_2$) |
|---|---|---|---|
| a | 0.9 | 0.85 | 0.85 |
| b | 0.8 | 0.7 | 0.7 |
| | | | |
| | | | |

# Threshold Algorithm Cont.

**Stop**

**Here**

| Rate | Distance |
|---|---|
| (a, 0.9) | (d, 0.9) |
| (b, 0.8) | (a,0.85) |
| (c, 0.72) | (b, 0.7) |
| . | . |
| . | . |
| . | . |
| (d, 0.6) | (c, 0.2) |

T = min(0.72, 0.7) = 0.7

| ID | $A_1$ | $A_2$ | Min($A_1$,$A_2$) |
|---|---|---|---|
| a | 0.9 | 0.85 | 0.85 |
| b | 0.8 | 0.7 | 0.7 |
| | | | |
| | | | |

# Definition: Safe Algorithm

An algorithm is **safe** if the algorithm will never output an item with no attribute in non-faulty nodes.

# What if there are faulty nodes ?

Ex: Three peers &
one faulty node

| | A | B | C |
|---|---|---|---|
| item 1 | 0 | 0 | 1 |
| item 2 | 0 | 1 | 0 |
| item 3 | 1 | 0 | 0 |

**All safe algorithms perform badly here.**

# Three Faulty Behaviors

(1) Zero: The faulty nodes set all the attributes in it to zero.

(2) Uniform: The faulty nodes set all the attributes in it to a random number between 0 and 10 times the
maximum value of all attributes.

(3) Large: The faulty nodes set all the attributes in it to a random number between 5 and 10 times the
maximum value of all attributes.

# Unsafe Algorithms

Algorithm 1: In the beginning of the algorithm, we randomly picked f peers and assume them to be faulty. Then run the Threshold Algorithm.

Algorithm 2: Randomly pick one node to exclude from our system for this round. Then, run the Threshold Algorithm to nd the top one item among the remaining peer, and add it to the output. Then randomly pick another node. Run the Threshold Algorithm for the remaining peers. Add the highest ranking item that's not in the output yet. Repeat until we have k items.

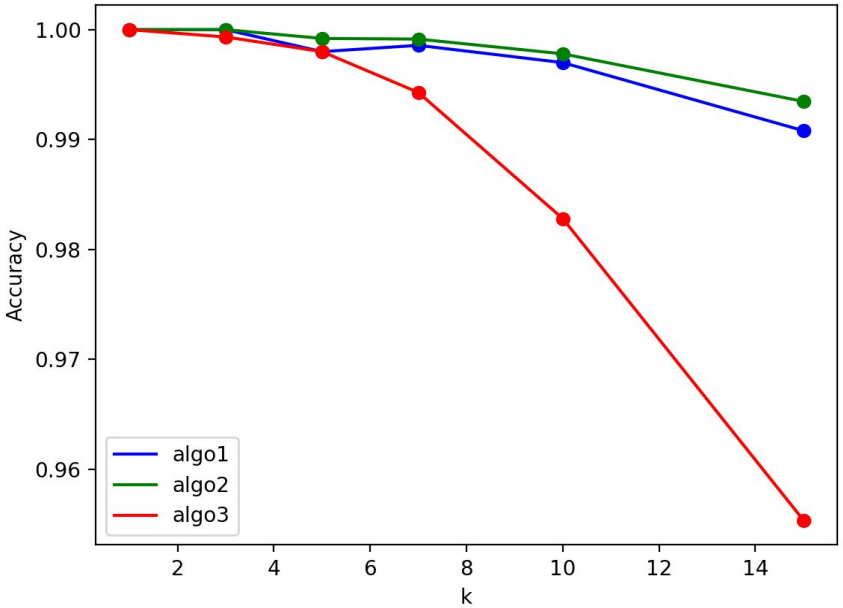
Algorithm 3: We first find the median of each item over all peers, and then give rank of each item according to their median.

## Experiments

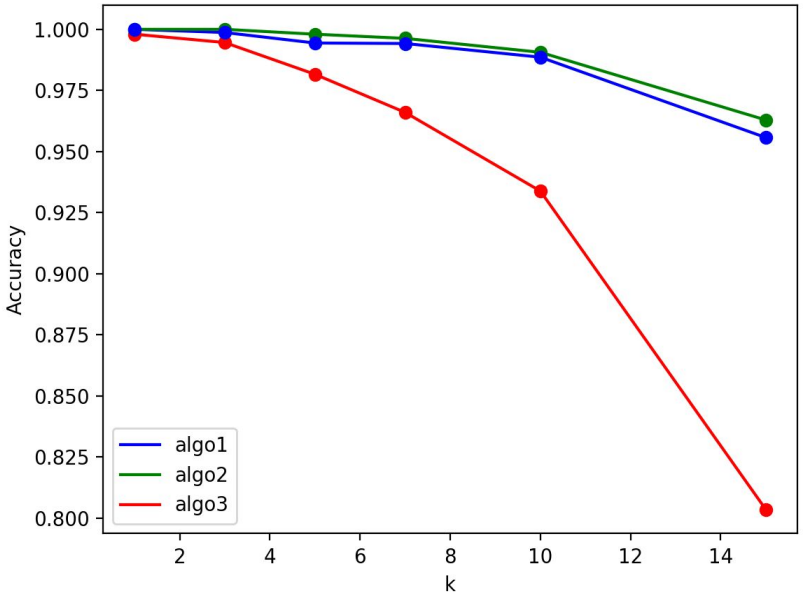
1. Consider all three faulty behaviors
2. k $\in$ [1,3,5,7,10,15] (that is, different length of output)
3. For each k, make 500 datasets with zipf(1.34/1.3/1.25, 50)
4. Run the algorithms in the 500 datasets
5. Calculate the average accuracy

# Results :

Approaching to 1: accuracy increases



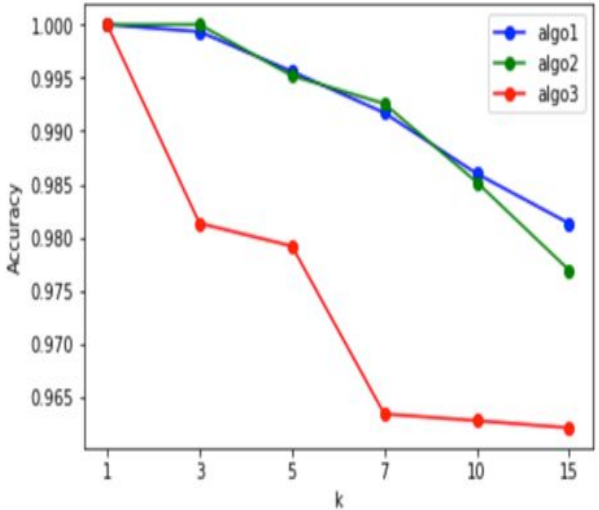

Zipf(1.34, 50) with zero faulty nodes

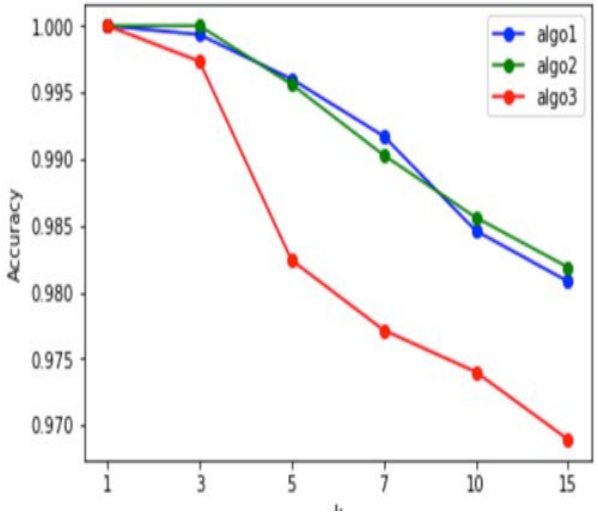


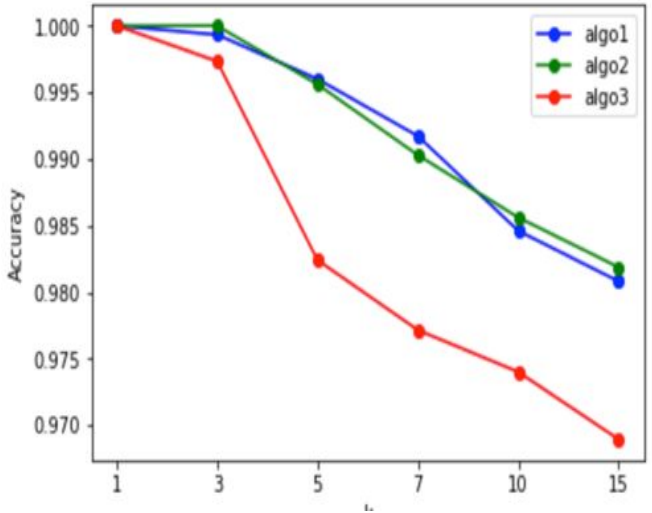

Zipf(1.25, 50) with zero faulty nodes

# Results :

For zipfian: zero nodes decrease the accuracy the most




Zero nodes

# Results
# For Uniformly distributed datasets