

 **mpi** *Network-Centric Systems Information* **NetCInS**

## **KLEE: Approximate Distributed Top-k Query Algorithms**

**Sebastian Michel**   **Peter Triantafillou**   **Gerhard Weikum**  
Max-Planck Institute for Informatics   RACTI / Univ. of Patras   Max-Planck Institute for Informatics  
Saarbrücken, Germany   Rio, Greece   Saarbrücken, Germany  
smichel@mpi-inf.mpg.de   peter@ceid.upatras.gr   weikum@mpi-inf.mpg.de

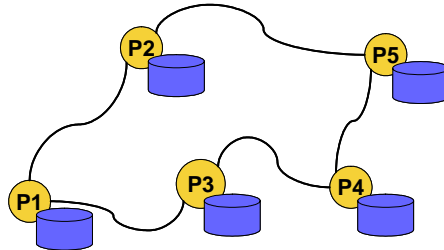
*Max-Planck-Institut Informatik*   *KLEE: Approximate Distributed Top-k Query Algorithms*   *University of Patras NetCInS Lab*

## **Overview**

- **Problem Statement**
- **Related Work**
- **KLEE**
- **The Histogram Bloom Structure**
- **Candidate Filtering**
- **Evaluation**
- **Conclusion / Future Work**

## Computational Model

- distributed aggregation queries:  
*Query with  $m$  terms with index lists spread across  $m$  peers  $P1 \dots Pm$*

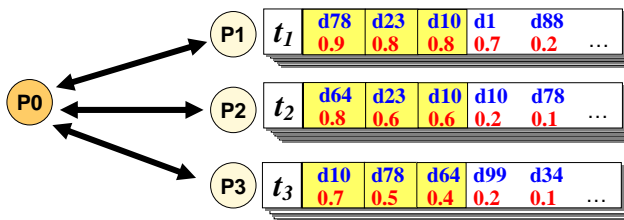


### Applications:

- Internet traffic monitoring
- Sensor networks
- P2P Web search

## Problem Statement

Query initiator  $P0$  serves as per-query coordinator



- Consider
  - network consumption
  - per peer load
  - latency (query response time)
    - network
    - I/O
    - processing

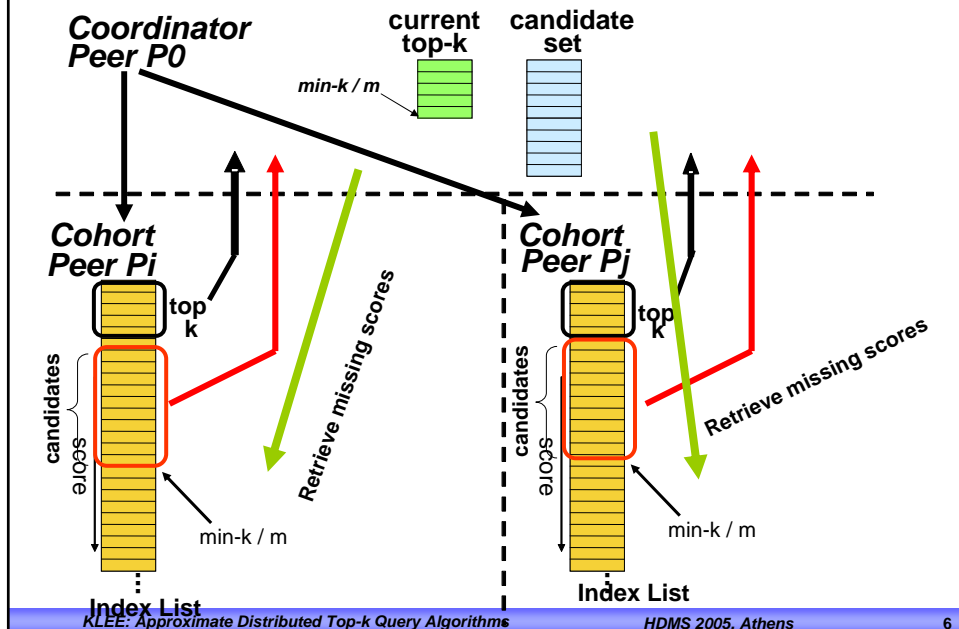
## Related Work

### Existing Methods:

- **Distributed NRA/TA:** NRA/TA (Fagin et al. '99/'03, Güntzer et al. '01, Nepal et al. '99) with batched access
- **TPUT (Cao/Wang 2004):**
  - 1) fetch  $k$  best entries ( $d, s_j$ ) from each of  $P_1 \dots P_m$  and aggregate ( $\sum_{j=1..m} s_j(d)$ ) at  $P_0$
  - 2) ask each of  $P_1 \dots P_m$  for **all entries with  $s_j > \min-k / m$**  and aggregate results at  $P_0$
  - 3) **fetch missing scores** for all candidates by random lookups at  $P_1 \dots P_m$

- + **DNRA aims to minimize per-peer work**
- **DTA/DNRA incur many messages**
- + **TPUT guarantees fixed number of message rounds**
- **TPUT incurs high per-peer load and net BW**

## TPUT



## KLEE: Key Ideas

- if  $\text{min}_k / m$  is small TPUT retrieves a lot of data in Phase 2
  - high network traffic
- random accesses
  - high per-peer load

### **KLEE:**

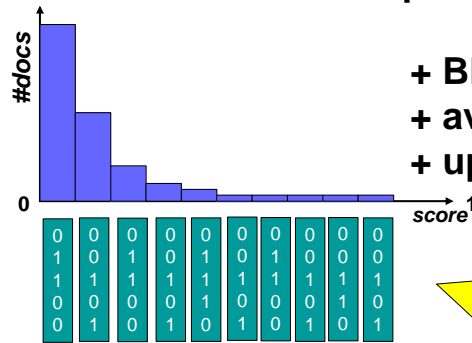
- Different philosophy: **approximate** answers!
- **Efficiency:**
  - Reduces (docId, score)-pair transfers
  - no random accesses
- Two pillars:
  - The **HistogramBlobs** structure
  - The **Candidate List Filter** structure

## The KLEE Algorithms

- **KLEE 3/4:**
  1. **Exploration Step:** ... to get a better approximation of min-k score threshold
  2. **Optimization Step:**
    - decide: 3 or 4 phases?
  3. **Candidate Filtering:** ... a docID is a good candidate if high-scored in many peers.
  4. **Candidate Retrieval:** get all good docID candidates.

## Histogram Bloom Structure

- Each peer pre-computes for each index list:  
an equi-width histogram



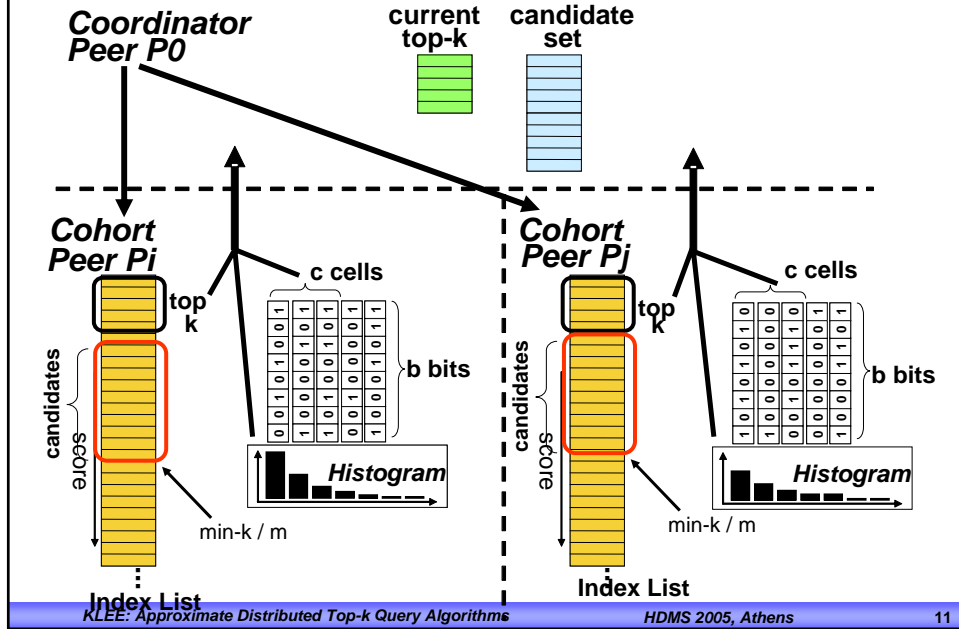
- + Bloom filter for each cell
- + average score per cell
- + upper/lower score

“increase” the mink / m threshold

## Bloom Filter

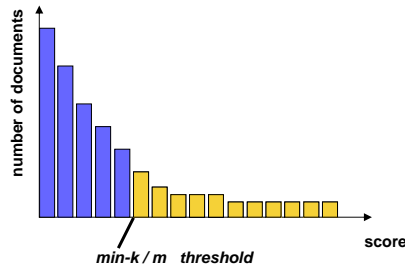
- bit array of size  $m$
- $k$  hash functions
 
$$h_i: docId\_space \rightarrow \{1, \dots, m\}$$
- insert  $n$  docs by hashing the ids and setting the corresponding bits
- Membership Queries:**
  - document is in the Bloom Filter if the corresponding bits are set
- probability of false positives ( $pfp$ )  $pfp = (1 - e^{-kn/m})^k$
- tradeoff accuracy vs. efficiency

## Exploration and Candidate Retrieval



## Candidate List Filter Matrix

- **Goal:** filter out unpromising candidate documents in step 2
- estimate the max number of docs that are above the  $\text{min-k} / m$  threshold



- send this number and the threshold to the cohort peers

## Candidate List Filter Matrix (2)

- Each cohort returns a Bloom Filter that “contains” all docs above the  $\text{min-k} / m$  threshold

### → Candidate List Filter Matrix (CLFM)

0101010010111110101001001010101001

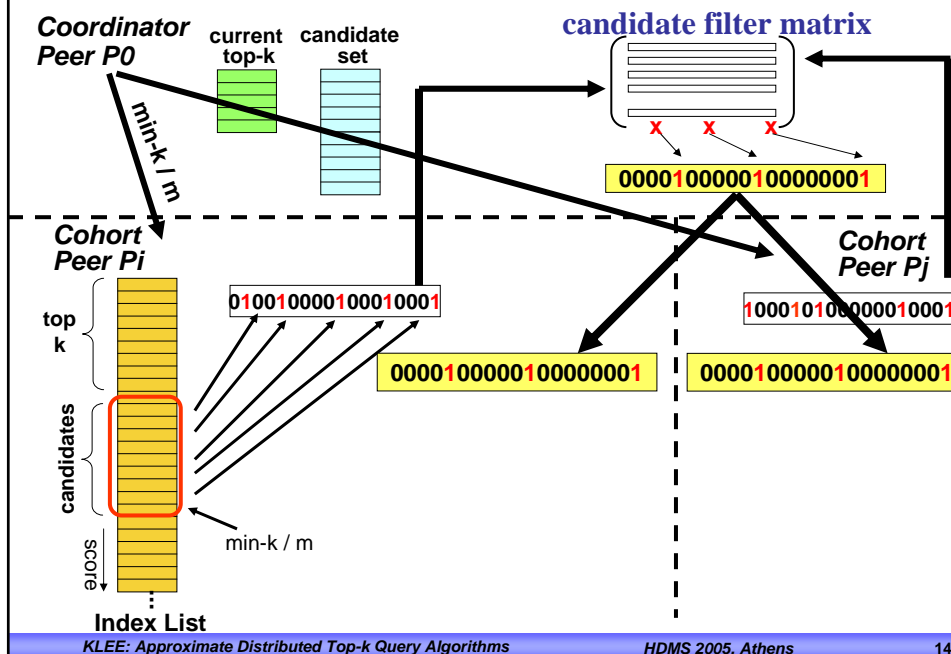
0100100110010111110010010101111110

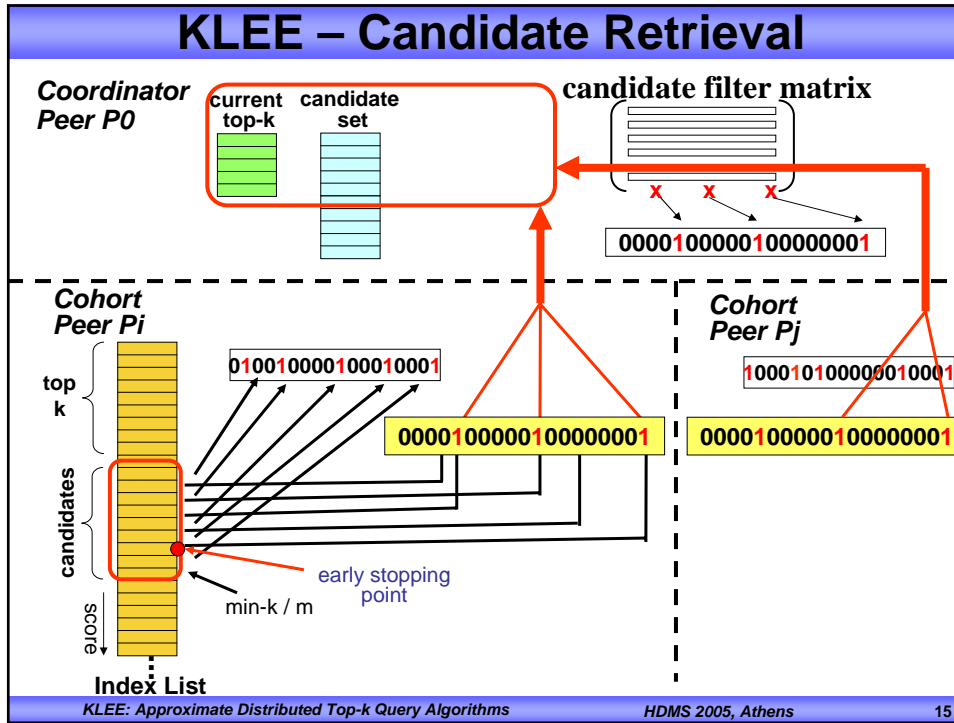
10101010101010011001001001110000

000000001000000100000000000100000

Select all columns with at least R bits set

## KLEE– Candidate Set Reduction





## Enhanced Filtering

- BF representation can be improved ...

|   |        |  |              |        |              |        |              |        |
|---|--------|--|--------------|--------|--------------|--------|--------------|--------|
| <ul style="list-style-type: none"> <li>(d1, 0.9)</li> <li>(d2, 0.6)</li> <li>(d5, 0.5)</li> <li>(d3, 0.3)</li> <li>(d4, 0.25)</li> <li>(d17, 0.08)</li> <li>(d9, 0.07)</li> </ul> |        | <table border="1" style="border-collapse: collapse;"> <tr><td style="padding: 2px;">010100000000</td><td style="padding: 2px;">Cell 1</td></tr> <tr><td style="padding: 2px;">010001000100</td><td style="padding: 2px;">Cell 2</td></tr> <tr><td style="padding: 2px;">010000000100</td><td style="padding: 2px;">Cell 3</td></tr> </table> | 010100000000 | Cell 1 | 010001000100 | Cell 2 | 010000000100 | Cell 3 |
| 010100000000  | Cell 1 |  |              |        |              |        |              |        |
| 010001000100  | Cell 2 |  |              |        |              |        |              |        |
| 010000000100  | Cell 3 |  |              |        |              |        |              |        |

$d1 \in \text{cell}_1$  and  $d2 \in \text{cell}_1$  but  $s1 - s2 = 0.3!$

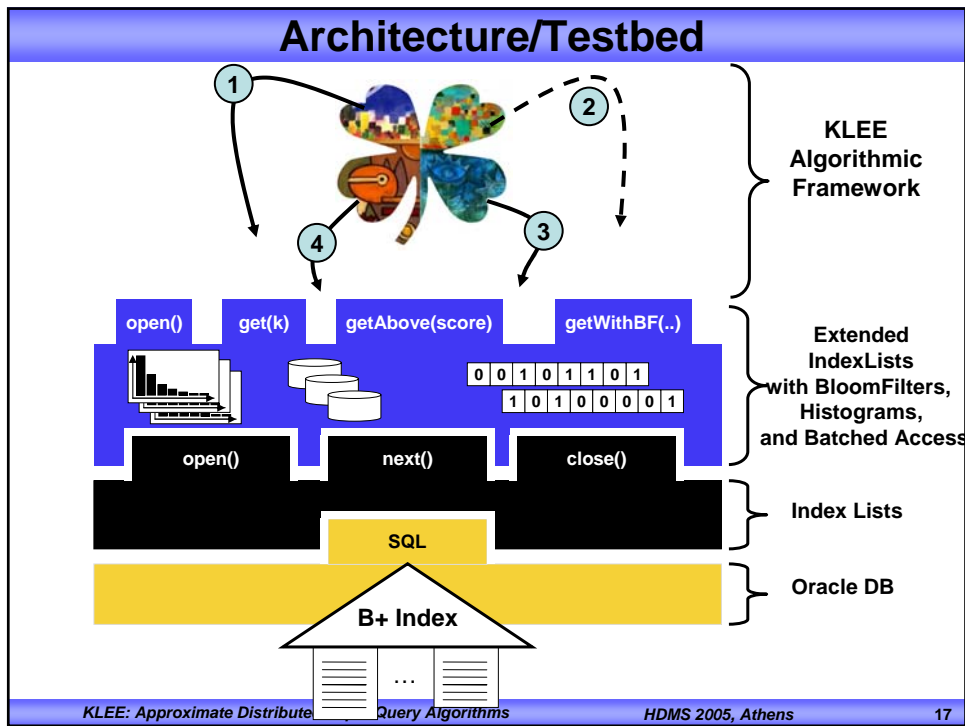
- Send byte-array with cell-numbers instead of bits

|    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|-----|
| 28 | 23 | 14 | 85 | 26 | 76 | 27 | ... |
|----|----|----|----|----|----|----|-----|

- Select „columns“ with

Sum over upper-bounds > min-k

*KLEE: Approximate Distributed Top-k Query Algorithms* *HDMS 2005, Athens* 16



- ### Evaluation: Benchmarks
- **GOV:** TREC .GOV collection + 50 TREC-2003 Web queries, e.g. *juvenile delinquency*
  - **XGOV:** TREC .GOV collection + 50 manually expanded queries, e.g. *juvenile delinquency youth minor crime law jurisdiction offense prevention*
  - **IMDB:** Movie Database, queries like
    - actor = John Wayne; genre =western
  - **Synthetic Distribution** (Zipf, different skewness): GOV collection but with synthetic scores
  - **Synthetic Distribution + Synthetic Correlation:** 10 index lists
- KLEE: Approximate Distributed Top-k Query Algorithms      HDMS 2005, Athens      18

## Evaluation: Metrics

- Relative recall w.r.t. to the actual results
- Score error
- Bandwidth consumption
- Rank distance
- Number of RA and number of SA
- Query response time
  - network cost (150ms RTT,  
800Kb/s data transfer rate)
  - local I/O cost (8ms rotation latency  
+ 8MB/s transfer delay)
  - processing cost

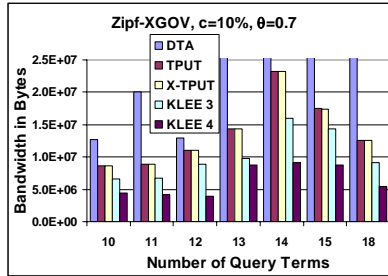
## Evaluated Algorithms

- **DTA:**
    - batched distributed threshold algorithm,  
batch size k.
  - **TPUT**
  - **X-TPUT:**
    - approximate TPUT. No random  
accesses.
  - **KLEE-3**
  - **KLEE-4**
- } C = 10% of the score mass

# Synthetic Score Benchmarks

| Zipf-GOV<br>c=10% | Total<br># of Bytes | Total<br>Time in ms | Average<br>Recall |
|-------------------|---------------------|---------------------|-------------------|
| DTA               | 17,752,769          | 3,532,180           | 1                 |
| TPUT              | 53,494,903          | 576,713             | 1                 |
| X-TPUT            | 53,011,252          | 404,991             | 0.99              |
| KLEE 3            | 49,861,342          | 367,931             | 0.97              |
| KLEE 4            | 25,057,920          | 160,585             | 0.94              |

| Zipf-XGOV<br>c=10% | Total<br># of Bytes | Total<br>Time in ms | Average<br>Recall |
|--------------------|---------------------|---------------------|-------------------|
| DTA                | 617,009,260         | 39,582,682          | 1                 |
| TPUT               | 377,928,880         | 1,599,581           | 1                 |
| X-TPUT             | 377,097,644         | 1,521,220           | 0.98              |
| KLEE 3             | 287,294,812         | 1,189,891           | 0.91              |
| KLEE 4             | 165,077,807         | 375,077             | 0.92              |

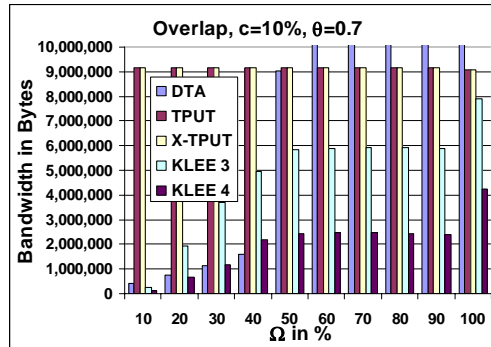
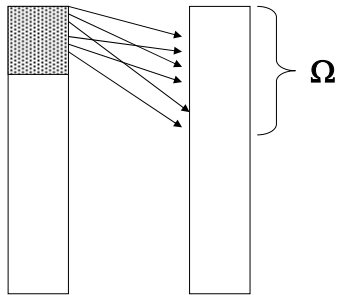


$\theta = 0.7$

# Synthetic Correlation Benchmark

| Overlap+Zipf<br>c=10% $\Omega = 30\%$ | Total<br># of Bytes | Total<br>Time in ms | Average<br>Recall |
|---------------------------------------|---------------------|---------------------|-------------------|
| DTA                                   | 1,146,32            | 157,420             | 1                 |
| TPUT                                  | 9,150,904           | 29,270              | 1                 |
| X-TPUT                                | 9,150,904           | 28,335              | 1                 |
| KLEE 3                                | 3,678,780           | 12,971              | 0.92              |
| KLEE 4                                | 1,192,704           | 6,546               | 0.91              |

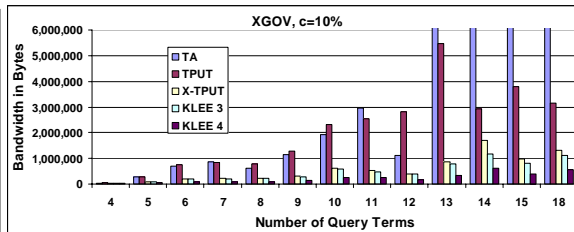
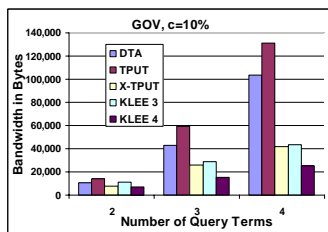
randomly insert top k documents from list i  
in the top  $\Omega$  documents of list j



## GOV / XGOV

| GOV<br>c=10% | Total<br># of Bytes | Total<br>Time in ms | Average<br>Recall |
|--------------|---------------------|---------------------|-------------------|
| DTA          | 1,172,446           | 190,259             | 1                 |
| TPUT         | 1,505,290           | 185,049             | 1                 |
| X-TPUT       | 597,991             | 31,432              | 0.89              |
| KLEE 3       | 722,664             | 28,319              | 0.9               |
| KLEE 4       | 440,868             | 39,564              | 0.9               |

| XGOV<br>c=10% | Total<br># of Bytes | Total<br>Time in ms | Average<br>Recall |
|---------------|---------------------|---------------------|-------------------|
| DTA           | 92,587,264          | 3,740,677           | 1                 |
| TPUT          | 70,044,884          | 2,346,882           | 1                 |
| X-TPUT        | 19,236,084          | 96,153              | 0.91              |
| KLEE 3        | 16,690,912          | 88,271              | 0.83              |
| KLEE 4        | 7,920,774           | 56,609              | 0.79              |



## Conclusion / Future Work

- **Conclusion**
  - KLEE: approximate top-k algorithms for wide-area networks
  - significant performance benefits can be enjoyed, at only small penalties in result quality
  - flexible framework for top-k algorithms, allowing for trading-off
    - efficiency versus result quality and
    - bandwidth savings versus the number of communication phases.
  - various fine-tuning parameters
- **Future Work**
  - Reasoning about parameter values
  - Consider “moving” coordinator

**Thanks for your attention!**

**Questions?**

**Comments?**

